KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK–ESAT
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee

Anonymity and Privacy in Electronic Services

Promotors:
Prof. Dr. ir. Bart Preneel
Prof. Dr. ir. Joos Vandewalle

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Claudia DIAZ

December 2005

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK–ESAT
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee

Anonymity and Privacy in Electronic Services

Jury:
Prof. Dr. ir. Guido De Roeck, voorzitter
Prof. Dr. ir. Bart Preneel, promotor
Prof. Dr. ir. Joos Vandewalle, promotor
Prof. Dr. ir. Bart De Decker
Prof. Dr. ir. Geert Deconinck
Dr. Dogan Kesdogan (RWTH Aachen University)
Prof. Dr. ir. Patrick Wambacq

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Claudia DIAZ

U.D.C. 681.3*D46 December 2005

D/2005/7515/99

ISBN 90-5682-671-9

Al abuelo

Acknowledgements

No man is an island, entire of itself...
– John Donne

This thesis would not have been possible without the help and support of many people I would like to acknowledge here.

First, I want to thank Prof. Bart Preneel for giving me the opportunity to do this Ph.D., guiding my research and finding the money to fund me; and Prof. Joos Vandewalle for offering me the opportunity to work in his research group.

I am also very grateful to Prof. Bart De Decker and Prof. Geert Deconinck for their reviews and suggestions to improve this dissertation. I want to thank Prof. Patrick Wambacq and Dr. Dogan Kesdogan for serving as jury members and Prof. Guido De Roeck for chairing the jury.

I would also like to thank all my co-authors, and in particular Joris Claessens, Andrei Serjantov, Len Sassaman and Evelyne Dewitte. Their collaboration has been essential in many of the research results presented here. Special thanks to Stefaan Seys for translating the abstract of this dissertation from English to Dutch. I would like to take this opportunity to also thank George Danezis, Roger Dingledine, Andreas Pfitzmann, Paul Syverson, Ben Laurie, Joss Wright and the other regular PET attendees for the fruitful and motivating discussions on anonymity research.

A big thanks goes to my (current and past) COSIC colleagues. I am very lucky to work with people who are always willing to help out when you need it. They have also been great company at work, for dinner, over a coffee or a beer. This includes in particular Danny De Cock, Klaus Kursawe, Svetla Nikova, Jasper Scholten and Berna Örs. Péla Noë, Marleen Somers and Elvira Wouters deserve a big thank you for their patience and valuable help with all sorts of administrative issues.

I also want to thank my family and friends back in Spain for making it so

easy to *disconnect* from the Ph.D. during holidays. And the last and biggest thank you goes to my partner Diego Juiz for his patience, understanding and encouragement over the time it has taken me to complete this Ph.D.

Finally, I would like to mention my two cats, Ambar and Borges, who have amused my writing days.

Claudia Díaz
Gent, December 2005

Abstract

This thesis presents information theoretic anonymity metrics and various analysis of anonymous communication nodes. Our contributions are a step towards the understanding of anonymity properties and the development of robust anonymous communications. Anonymous communications are an essential building block for privacy-enhanced applications, as the data available at the communication layer may leak critical private information.

One of the main contributions of our work is the *degree of anonymity*, a practical information theoretic anonymity metric. Entropy-based anonymity metrics can be applied to measure the degree of anonymity provided by an anonymous service to its users. In particular, these metrics can be applied to systems which leak probabilistic relationships between the anonymous subjects and their transactions.

We present a taxonomy of the two main building blocks used to implement anonymous communication networks, which are anonymous communication nodes (called *mixes*) and cover traffic policies (called *dummy traffic*). We propose a model for describing anonymous communication nodes which extends design possibilities and facilitates the analysis of anonymity properties. We identify the parameters which must be taken into account in the design and analysis of mix-based anonymous communication networks.

In order to show the practical applications of information theoretic anonymity metrics, we have applied the metrics to evaluate the anonymity properties of various nodes for anonymous communication which have been proposed in the literature. We analyze the anonymity provided by these nodes when subject to passive and active attacks, while considering scenarios with and without cover traffic techniques.

We have analyzed two working implementations of anonymous email in real traffic conditions. The tools used for the analysis are information theoretic metrics and our model for anonymous communication nodes. We show that anonymous email traffic patterns are hard to predict and no assumptions on them should be made. We find that the two studied designs offer very different trade-

offs for anonymity and performance.

All in all, we believe that information theoretic metrics are a useful tool to characterize anonymity properties. Our work is one step towards a better understanding of anonymity and our results can be used for the design of robust anonymity technologies.

Samenvatting

In deze thesis worden informatie-theoretisch metrieken voor het meten van anonimiteit en verscheidene analyses van anonieme communicatiesystemen voorgesteld. Onze bijdragen helpen om de verschillende aspecten van anonimiteit beter te begrijpen en bij het ontwerpen van robuuste anonieme communicatiesystemen. Anonieme communicatiekanalen zijn een essentiële bouwblok van privacy-ondersteunende toepassingen, aangezien de gegevens die beschikbaar zijn op de communicatielaag, persoonlijke gegevens kunnen bevatten.

Een van de hoofdbijdragen van ons werk is de *graad van anonimiteit*, een praktische informatie-theoretisch metriek voor het meten van anonimiteit. Metrieken gebaseerd op entropie kunnen gebruikt worden om de graad van anonimiteit te meten die aangeboden wordt aan gebruikers. In het bijzonder kunnen deze metrieken toegepast worden op systemen die probabilistische informatie lekken over relaties tussen anonieme gebruikers en hun transacties.

Wij stellen een taxonomie voor van de twee belangrijkste bouwblokken die gebruikt worden om anonieme communicatienetwerken te implementeren: netwerkknopen die anonimiteit voorzien (*mixes* genaamd) en richtlijnen voor *dummy* verkeer (hierdoor wordt het echte verkeer verstopt in een groter geheel van nepberichten). Wij stellen een model voor dat kan aangewend worden voor het beschrijven van deze mixes. Dit model breidt de ontwerpmogelijkheden voor mixes uit en vereenvoudigt de analyse van anonimiteitskenmerken. Wij identificeren de parameters die in acht genomen moeten worden bij het ontwerp en analyse van anonieme communicatienetwerken.

Om de praktische toepassingen van informatie-theoretisch metrieken voor het meten van anonimiteit aan te tonen, hebben we aan de hand van onze metrieken de anonimiteitskenmerken geëvalueerd van verscheidene mixes die in de literatuur beschreven worden. Wij analyseren de anonimiteit voorzien door mixes wanneer deze onderworpen zijn aan passieve en actieve aanvallen, in scenario's met en zonder verkeer van nepberichten.

We hebben twee werkende implementaties van anonieme elektronische post geanalyseerd. In deze analyse werd gebruik gemaakt van informatie-theoretische

metrieken en ons model voor het beschrijven van mixes; beiden toegepast op echte gegevens. We tonen aan dat patronen in anonieme elektronische post moeilijk te voorspellen zijn en dat het niet aangewezen is hier veronderstellingen over te maken. Uit onze bevindingen blijkt dat de twee bestudeerde ontwerpen zeer verschillende afwegingen bieden aangaande anonimiteit en performantie.

Tot slot geloven wij dat informatie-theoretische modellen een nuttig middel zijn, dat kan aangewend worden voor het karakteriseren van anonimiteit. Ons werk kan helpen om de verschillende aspecten van anonimiteit beter te begrijpen en onze resultaten kunnen gebruikt worden bij het ontwerp van robuuste anonieme communicatiesystemen.

Contents

List of Figures

List of Tables

List of Abbreviations

CDF	Cumulative Distribution Function
DES	Data Encryption Standard
GMM	Generalized Mix Model
IOI	Item Of Interest
MD	Message Digest
MTA	Mail Transport Agent
OS	Operating System
PDF	Probability Density Function
PETs	Privacy Enhancing Technologies
POSIX	Portable Operating System Interface
PRNG	Pseudo Random Number Generator
RGB	Red-Green-Black
RSA	Rivest-Shamir-Adleman
S-G	Stop-and-Go
SSL	Secure Sockets Layer

Chapter 1

Introduction

Information is power.

1.1 Motivation

People increasingly use the Internet for an ever wider range of activities: reading the newspaper, shopping, staying in contact with family and friends, finding a partner, booking holidays, expressing their opinion, keeping an online diary, etc.

While performing an online activity, even if the confidentiality of the information being transmitted is protected through encryption, the source and destination of the communication are easily traceable. The information on who communicates with whom may reveal critical information that could be used against the Internet user. For example, someone accessing a web site with information on a life-threatening disease may not obtain a health insurance or lose his job if this information gets to the insurance company or the employer.

The linkability of all traffic information generated by an Internet user (e.g., through the IP address, national ID number or social security number), allows for sophisticated profiling of each user. Some of the data that could be gathered and stored directly or indirectly, just by monitoring the user's communication are: email address, age, gender, location, religious preferences, sexual orientation, bank, job, type of products bought on the Internet, period of holidays, political orientation, lifestyle, or social network.

In the current communication infrastructure, traffic data is available at moderate cost to anyone willing to harvest it, without the data subject being aware of it. There is already an emerging market of personal data that criminals use to impersonate their victims. In some cases, the damage inflicted to identity theft victims is huge. With the development of data collection technologies, data stor-

age capacity and profiling techniques, these data become easier to get, cheaper to store and more profitable to use (either for legal or illegal practices).

Software tools which can be used for privacy violations are increasingly available. These include spyware such as key loggers (e.g., KeyLog Pro [Pro]) or search engines like Google [Goo].

In this scenario, large amounts of information about large numbers of people are under the control of a few data holders. Users effectively lose control over their own personal data, and at the same time all their online activity can be collected, aggregated and stored. These data can be used to take decisions that impact the data subjects. This asymmetric distribution of information creates dangerous imbalances, as those in control of the information may use their acquired power for many different purposes.

This possibility is particularly disturbing in contexts where human rights are not respected. Totalitarian regimes may monitor electronic communication in order to identify (and punish) dissidents or journalists. Mass profiling may also be used to identify members of minorities (such as homosexuals) and take actions against them.

The European Directive on Data Protection [PC95] is an attempt to protect citizens from these threats. However, it is very difficult to enforce, and its practical effectiveness is yet to be proven. Technology can be designed to keep personal data under the control of the user. The user could disclose the minimal amount of information to the entities with which he interacts.

Anonymity technologies serve as tools for the protection of privacy in electronic applications, and they are a key component of Privacy Enhancing Technologies (PETs). Anonymous communication networks protect the privacy of Internet users towards the other end of the communication and towards observers in the network. This is achieved by hiding the link between the initiator of the communication and the responder. For applications such as electronic voting and electronic payments, anonymity and privacy are strictly necessary. In a democratic society, public elections will be held anonymously and citizens have a fundamental right to privacy, for example when buying goods or subscribing to services.

1.2 Related Work

The field of research of anonymity technologies started in the early 80's with David Chaum's paper on untraceable electronic mail [Cha81]. However, it was not before year 2000 that anonymity and privacy enhancing technologies started to get the attention of a large research community. In Fig. 1.1 we show the number of yearly publications referenced in the Freehaven Anonymity Bibliography [Din] for the period $1981 - 2004$.

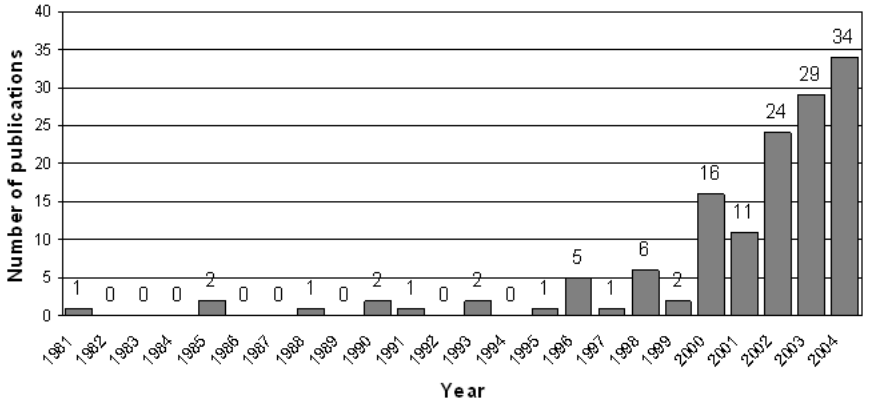


Figure 1.1: Number of Yearly Publications (Freehaven Anonymity Bibliography)

We can distinguish several sub-topics within the field of Privacy Enhancing Technologies. The numbers of publications per sub-topic referenced in the Freehaven Anonymity Bibliography are shown in Fig. 1.2. Here, we summarize the main research contributions to each of these sub-topics:

- **Anonymous Communication.** The problem on which more work has been performed is the study of technologies that can anonymize the communication layer. These technologies include mix-based general purpose anonymous communication networks such as TOR [DMS04], Onion Routing [STRL00, GRS96] or ISDN mixes [PPW91]; anonymous electronic mail such as Chaum's original proposal [Cha81], Babel [GT96] or Mixminion [DDM03]; peer-to-peer anonymizing network layers such as Crowds [RR98], Tarzan [FM02], MorphMix [RP04], P^5 [SBS02], Cebolla [Bro02] or Herbivore [GRPS03]; DC-nets [Cha88, WP90, GJ04]; proposals and analysis of mixes [KEB98, JMP+98, Dan02, DS03b, DP04b]; and proposals to improve the resistance of anonymous communication systems towards attacks [DS03a, DFHM01, BFTS04, SM05, DC05].
- **Traffic Analysis.** A substantial number of publications of the recent years have focussed on the various traffic analysis attacks that can be deployed against anonymous communication systems. Among these, we point out the Sybil attack [Dou02], fingerprinting attacks [Hin02], long-term intersection attacks [BL02], blending attacks [SDS02], statistical disclosure

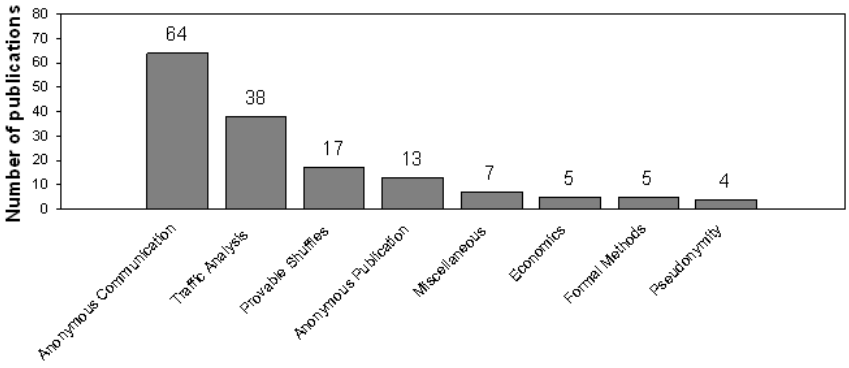


Figure 1.2: Number of Publications per Sub-Topic (Freehaven Anonymity Bibliography)

attacks [DS04], timing attacks [LRWW04] and hitting set attacks [KP04]. Anonymity metrics [SD02, DSCP03] are a tool to measure the success of traffic analysis attacks.

- **Provable Shuffles.** For critical applications such as electronic voting, it is important to prove that the shuffle (permutation of inputs into outputs) has been correctly and securely performed. We can find in the literature several proposals for provable shuffles, such as Flash mixing [Jak99, MK00], universally verifiable mixing [Abe98] and hybrid mixes [OA00, JJ01].

- **Anonymous Publication.** Censorship-resistant systems aim at providing individuals with the possibility of anonymously publishing information in such a way that it cannot be removed. The most relevant proposals for anonymous publication systems are the Eternity Service [And96, Ben01], TAZ servers [GW98], The Free Haven Project [DFM00], Freenet [CSWH00], Publius [WRC00] and Tangler [WM01].

- **Economics.** Incentives may be critical for the success of the implementation of Privacy Enhancing Technologies. The research that has been done in this direction focusses on the understanding of the underlying economic aspects of anonymity [ADS03, FR01, DA04]; and on the design of reputation systems that stimulate users to behave honestly [DMS03, Gro03].

- **Formal Methods.** There have been attempts to formalize anonymity properties. Garcia *et al.* [GHPvR05] propose a formal framework for the analysis of information hiding properties. Syverson and Stubblebine [SS99]

introduce group principals for the formalization of anonymity. A probabilistic model checker to analyze anonymity systems is proposed by Shmatikov in [Shmng].

- **Pseudonymity.** Pseudonyms are used in systems where users need to maintain a permanent identity. Pseudonyms have been proposed for a variety of systems, such as electronic mail systems [MK98, SCM05], or communication infrastructures [Gol00].

- **Miscellaneous.** Two surveys on the state of Privacy Enhancing Technologies were presented by Goldberg [GWB97, Gol02] in 1997 and 2002. A terminology for pseudonymous and anonymous systems was proposed by Pfitzmann and Hansen in [PH04]. Clayton and Danezis studied the real world patterns of failure in anonymity systems in [CDK01].

Some other relevant publications which are not listed in the Freehaven Anonymity Bibliography include several privacy enhanced **payment systems** [Bra93, CFN88, CPS94, Bra95, PS00, CPS96, STS99a, CMS96, Cam98, BBC+94], which allow users to keep private the information on their online purchases. There has been as well substantial research on **pseudonymous credentials** [CE87, Cha90, Bra99, Che95, LRSW99, CL01, CH02, JC02], which are a privacy-enhanced alternative to public key certificates.

Several escrow mechanisms have been proposed for preventing fraud in anonymous electronic cash [DFTY97, FO96, PS00, STS99b]. However, the issue of identity escrow in anonymous communications remains controversial. We have proposed two theoretical architectures [CDG+03, Dó5] that use fair blind signatures and pseudonymous credentials, respectively, as mechanisms to recover the identity of misbehaving users.

1.3 This Thesis

The main focus of this thesis is on the development of anonymity metrics and on the analysis of anonymous communication nodes. Anonymous communications are an essential building block of anonymity systems. Our work is one step towards the understanding of anonymity techniques and the evaluation of their effectiveness.

Anonymity metrics are an essential tool to evaluate the anonymity properties of a given system. They also enable objective comparison of different approaches to provide anonymity services. The anonymity metrics presented here can be applied to a wide range of anonymity systems. We present an abstract model suited for the quantification of anonymity and various examples on how it can be applied to evaluate anonymous communication nodes.

This thesis introduces a model for anonymous communication nodes which establishes a framework for the analysis of anonymity strategies and opens new design possibilities. We also propose randomization techniques which improve the robustness of these nodes.

The models for anonymity quantification and anonymous communication nodes are applied to the evaluation of various theoretical node proposals. We analyze, discuss and compare the anonymity properties of existing node designs. We also present an evaluation of two working implementations of anonymous email.

This thesis summarizes the most relevant research results we have published in the last four years. We have worked in other research publications that have not been included here in order to keep this thesis more focussed. In [DP04a], we provide an overview of anonymous communications and its building blocks. We have studied the possibility of anonymity revocation for anonymous communication infrastructures in [CDG+03]. An extension of the information theoretic metrics presented in this thesis has been published in [DCSP02] to model the anonymity of subjects classified in groups. Finally, we have studied the impact of network topology on the anonymity provided by anonymous communication networks in [BDD+05].

1.4 Outline and Main Contributions

The outline of this doctoral dissertation is the following:

- **Chapter** 1 presents the motivation and context of the work performed for this thesis.
- **Chapter** 2 introduces information theoretic anonymity metrics. We provide a general model for anonymity systems and present two flavors of entropy based anonymity metrics: the *effective anonymity set size* and the *degree of anonymity*, which is one of our original contributions to the field, and was published in [DSCP03]. We apply the metrics to an anonymity system and discuss the results obtained.
- **Chapter** 3 presents an analysis of mixes and dummy traffic policies. The different issues related to the analysis and design of mix based anonymous communications are brought together, and the parameters of mix strategies as well as dummy traffic policies are identified and discussed. Most of the work presented in this chapter was published in [DP04c]. We introduce in this chapter a model to express pool mixes, called the *Generalized Mix Model*, and a new variant of pool mixes called *binomial mixes* which were published in [DS03b].

- **Chapter** 4 studies the impact of passive attacks (traffic analysis attacks) on the nodes of an anonymous communication network. We indicate how to compute the recipient and sender anonymity and we point out some problems that may arise from the intuitive extension of the metric to take into account dummies. Two possible ways of inserting dummy traffic are discussed and compared. The work presented in this chapter was published in [DS03b, DP04b].

- **Chapter** 5 analyzes the deployment of an active attack (the *blending* or $n-1$ attack) on (timed and threshold) deterministic pool mixes, binomial mixes, continuous mixes, and pool mixes that generate dummy traffic. We define a set of parameters to measure the effort of the attacker, and we study how those parameters can be measured for these mix configurations. We analyze the remaining anonymity of the message under attack, and show how the use of dummies impacts it. These results were published in [DS03b, DP04b].

- **Chapter** 6 presents quantitative results on the anonymity actually provided by two mix software implementations in wide deployment, to test the actual anonymity provided to the users of the remailer service, and to compare the two different designs. We evaluate anonymity in a single-node context. As individual nodes are the basic component to the network of mixes, we aim to provide information to be considered when choosing this component. We have used as input real-life data gathered from a popular remailer, and simulated the behavior of the mix. This research was published in [DSD04].

- **Chapter** 7 summarizes the conclusions of this thesis and presents lines for future research.

Chapter 2

Anonymity Metrics

For most of history, Anonymous was a woman.
– Virginia Woolf

2.1 Introduction

The need for a metric to measure the performance of anonymity implementations appeared with the development of applications that enabled anonymous electronic transactions, such as untraceable email, electronic voting, anonymous e-coins or privacy-enhanced web browsing.

The research questions that arose were: how can anonymity be measured? How can two different anonymity systems be compared? Is there a general anonymity metric which can be applied to any anonymity system? How can we evaluate the effectiveness of different attacks on the anonymity system? How can we quantify losses and gains in anonymity? How can anonymity metrics reflect the partial or statistic information often obtained by an adversary? What is a sufficient level of anonymity? The metrics described here provide answers to these questions.

The work presented in this chapter is an original contribution to the field of privacy enhancing technologies, and was published in the *2nd Workshop on Privacy Enhancing Technologies* 2002 [DSCP03]. The presented anonymity metrics provide answers to the research questions formulated above. However, the question on which is the sufficient level of anonymity remains open, as it depends on the context and specific needs of the application, which go beyond the technical dimension.

The anonymity metrics presented here can be applied to concrete systems, adversaries and conditions. These metrics give a measure of the size and distin-

guishability of the set of subjects potentially linked to a particular transaction, and attacked by a concrete adversary. In order to get an idea on the performance of an anonymity implementation under different conditions, multiple anonymity measurements must be made and analyzed.

Information theoretic metrics can be applied to a broad range of anonymity systems. It is thus important to understand the concepts behind entropy-based anonymity metrics in order to apply and interpret them correctly in concrete scenarios. The metrics must be adapted to the anonymity system under study, and the computation of probability distributions that lead to meaningful metric values is not always obvious. The intention of this chapter is to explain the basics of information theoretic anonymity metrics, which are then applied to concrete systems in Chapters 4, 5 and 6.

We put this work into context by describing the related work in Sect. 2.2. The model for anonymity systems is described in Sect 2.3, and the attack model in Sect. 2.4. Section 2.5 describes information theoretic anonymity metrics, which are then applied to a practical example in Sect 2.6. Finally, Sect. 2.7 presents the conclusions of this chapter.

2.1.1 Defining Anonymity

Prior to the quantification of anonymity, a working definition for the term *anonymity* was needed. Pfitzmann and Hansen [PH04] defined *anonymity* as *the state of being not identifiable within a set of subjects, the anonymity set.* This definition, first proposed in year 2000, has been adopted in most of the anonymity literature.

According to the Pfitzmann-Hansen definition of anonymity, the subjects who may be related to an anonymous transaction constitute the *anonymity set* for that particular transaction. A subject carries on the transaction *anonymously* if he cannot be distinguished (by an adversary) from other subjects. This definition of anonymity captures the probabilistic information obtained by adversaries trying to identify anonymous subjects, as we explain in Sect. 2.5.

2.2 Related Work

Before information theoretic anonymity metrics were proposed, there had been some attempts to quantify anonymity in communication networks.

Reiter and Rubin [RR98] define the *degree of anonymity* as a probability $1-p$, where p is the probability assigned by an attacker to potential senders. In this model, users are more anonymous as they appear (towards a certain adversary) to be less likely of having sent a message. This metric considers users separately, and therefore does not capture anonymity properties very well. Consider a first

system with 2 users which appear to be the sender of a message with probability 1/2. Now consider a second system with 1000 users. User u_1 appears as the sender with probability 1/2, while all the other users are assigned probabilities of having sent the message below 0.001. According to the definition of Reiter and Rubin, the *degree of anonymity* of u_1 and of the two users of the first system would be the same (50%). However, in the second system, u_1 looks much more likely to be the sender than any other user, while the two users of the first system are indistinguishable to the adversary.

Berthold *et al.* [BPS00] define the *degree of anonymity* as $A = log_2(N)$, where N is the number of users of the system. This metric only depends on the number of users of the system, and therefore does not express the anonymity properties of different systems. The total number N of users may not be known. Moreover, adversaries may be able to obtain probabilistic information on the set of potential senders, which is not taken into account in this metric.

Information theoretic anonymity metrics were independently proposed in two papers presented at the *2nd Workshop on Privacy Enhancing Technologies*. The basic principle of both metrics is the same. The metric proposed by Serjantov and Danezis [SD02] uses entropy as measure of the *effective anonymity set size*. Our metric [DSCP03] goes one step further, normalizing the entropy to obtain a *degree of anonymity* in the scale 0..1. The details of the two flavors of anonymity metrics are explained in Sect. 2.5.

2.3 Model

In this chapter, we present a general model for anonymity systems. In Sect. 2.6 we apply the model to Crowds and, in subsequent chapters, we use it to evaluate the anonymity of anonymous communication nodes.

Many anonymity systems can be modeled in terms of unlinkability. Unlinkability is defined by Pfitzmann and Hansen [PH04] as follows: *unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, events, actions, ...) means that within the system (comprising these and possibly other items), from the attacker's perspective, these items of interest are no more and no less related after his observation than they are related concerning his a-priori knowledge.*

Our model can be applied both to *sender* and *recipient anonymity*. If we consider the sending and receiving of messages as *Items Of Interest* (IOIs), anonymity may be defined as unlinkability of an IOI and a subject. More specifically, we can describe the anonymity of an IOI such that it is not linkable to any subject, and the anonymity of a subject as not being linkable to any IOI. In this context, *unlinkability* is achieved with high entropy values.

Figure 2.1 presents a simplified anonymity model. The goal of anonymity systems is to hide the relationship between subjects and IOIs. Hiding these links

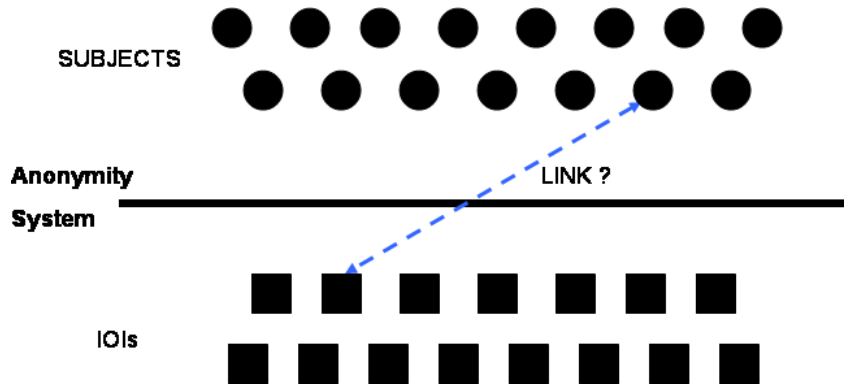


Figure 2.1: Model for Anonymity Systems

is the basic mechanism behind anonymous transactions.

An observer of the system sees that a set of subjects are accessing the anonymity system. At the output of the system, they see IOIs which are hard to link to a particular subject. The set of subjects who might be linked to an IOI is called the *anonymity set*. The larger the *anonymity set*, the more anonymity a subject is enjoying. The notion of *anonymity set* is the key to define anonymity metrics, as we show in Sect. 2.5.2.

2.4 Attack Model

We can distinguish two types of attacks on anonymity systems: *attacks on anonymity* and *attacks on the availability* of the anonymity service (also called *denial of service attacks*). Denial of service attacks may only be deployed by *active* attackers (see description below). These attacks are aimed at reducing the availability of the system, which may be a goal in itself, or part of an attack on anonymity (e.g., the adversary may block several entities from accessing the system in order to reduce the anonymity set). In this chapter, we are interested in the effects of the attacks on anonymity. More specifically, in measuring the certainty of the adversary on the existence of a link between a subject and an IOI.

The quantification of anonymity is dependent on the *adversary* or *attacker* considered. The adversary has certain capabilities and deploys attacks in order to gain information and find links between subjects and IOIs. Most of these attacks

lead to a distribution of probabilities that assign subjects a certain probability of being linked to IOIs.

The metrics we propose here take into account the probabilities assigned by the adversary to users potentially linked to an IOI. Note that the metrics measure anonymity *with respect to* a particular attack; results are no longer valid if the attack model changes. Therefore, concrete assumptions about the attacker have to be clearly specified when measuring anonymity. Some of the adversary's properties we should make explicit are:

- *Passive-Active:* A passive attacker listens to the communication and/or reads internal information of entities participating in the protocols; passive attackers typically perform traffic analysis of the communication. Active attackers can add, delay, alter or remove messages and modify internal information of participating entities.

- *Internal-External:* An internal attacker controls one or several entities that are part of the system (e.g., the attacker controls some communication nodes). External attackers only control communication links.

- *Partial-Global:* A global attacker has access to the entire communication system (e.g., all communication links), while a partial attacker (also called *local attacker* in the literature) only sees part of the resources (e.g., a limited number of peers in a peer-to-peer network).

- *Static-Adaptive:* Static attackers control a predefined set of resources and are unable to alter their behavior once a transaction is in progress. Adaptive attackers gain control on new resources or modify their behavior, depending on intermediate results of the attack.

2.5 Information Theoretic Anonymity Metrics

In this section, we first introduce the concept of entropy, on which information theoretic anonymity metrics are based. Then, we explain how the *effective anonymity set size* and the *degree of anonymity* can be computed. The metrics presented in this section are applied to a practical system in Sect. 2.6.

2.5.1 Entropy

The information theoretic concept of entropy [Sha48] provides a measure of the uncertainty of a random variable. Let X be the discrete random variable with probability mass function $p_i = Pr(X = i)$, where i represents each possible value that X may take with probability $p_i > 0$. In this case, each i corresponds to a

subject of the anonymity set; i.e., p_i is the probability of subject i being linked to the IOI.

We denote by $H(X)$ the entropy of a random variable, and by N the number of subjects in the anonymity set. $H(X)$ can be calculated as:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i) \ .$$

2.5.2 Effective Anonymity Set Size

The *effective anonymity set size* is an intermediate step to compute the *degree of anonymity*. Serjantov and Danezis proposed in [SD02] the use of the effective anonymity set size as metric. Our contribution, which was published in [DSCP03], proposes as anonymity metric a normalized entropy.

As mentioned in Sect. 2.1.1, *anonymity* was defined by Pfitzmann and Hansen in [PH04] as *the state of being not identifiable within a set of subjects, the anonymity set.* Anonymity metrics aim at giving a meaningful measure of the anonymity set size.

After deploying an attack on an anonymity system, the adversary typically obtains a distribution of probabilities that link subjects to the particular IOI of the attack. The probabilities are shown in Fig. 2.2 with the arrows that connect the IOI to the subjects of the anonymity set. Different subjects may appear as having a higher or lower probability p_i of having a link with the IOI, depending of the information obtained by the adversary using the attack.

Let N denote the total number of subjects which are linked to the IOI with a non-zero probability ($p_i > 0$, $i = 1..N$). The effective anonymity set size is defined as the entropy $H(X)$ of the distribution X of probabilities that link the subjects of the anonymity set to the IOI.

Entropy-based anonymity metrics give a measure of the uncertainty of the adversary on the subject who is related to the IOI. The *effective anonymity set size* takes into account the *number* of potential subjects linked to the IOI, and the *probabilities* assigned to the subjects.

The interpretation of the metric is as follows: when the effective anonymity set size has a value h, the anonymity subjects are enjoying is as if they were perfectly undistinguishable among 2^h subjects.

The metric (and thus anonymity) increases its value with two factors. First, with the number of subjects potentially linked to the IOI; and second, with the uniformity of the probability distribution. The more equally distributed the probabilities assigned to the subjects of the anonymity set, the higher the entropy (i.e., the higher the effective anonymity set size).

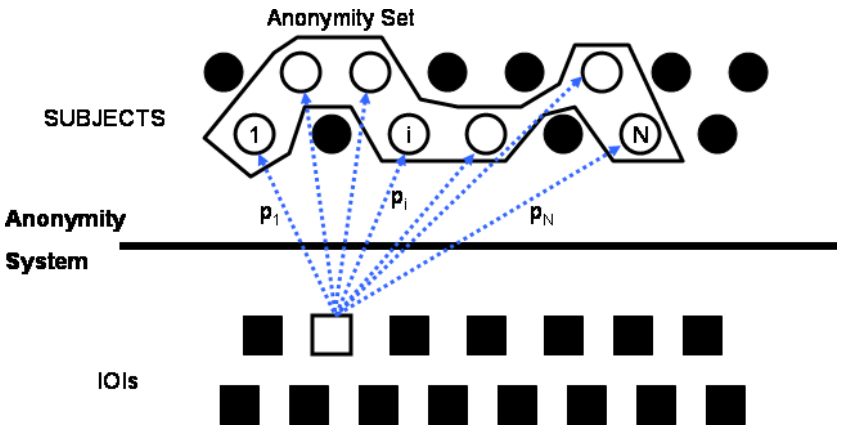


Figure 2.2: Anonymity Set

2.5.3 Degree of Anonymity

The *degree of anonymity* is a normalized version of the *effective anonymity set size*, which tells tells how good the system is performing on a $0-1$ scale. This metric is an original contribution of this thesis (note that both metrics were proposed independently at the same time).

The maximum effective anonymity set size for N subjects is reached when all subjects are linked to the IOI with equal probability (i.e., $p_i = 1/N$). In this case, all subjects are indistinguishable towards the adversary with respect to the IOI. For a given number N of users, the maximum achievable anonymity corresponds to the entropy of a uniform distribution. We denote the maximum entropy by H_M:

$$H_M = \log_2(N) \ .$$

If we assume that the adversary has no *a priori* information on the system (i.e., the *a priori* anonymity of an IOI is H_M), the amount of information gained by the adversary with an attack is the difference in the entropy before and after the attack, that is: $H_M - H(X)$.

The *degree of anonymity* is defined as the normalized value of this difference in knowledge of the adversary:

$$d = 1 - \frac{H_M - H(X)}{H_M} = \frac{H(X)}{H_M} \ .$$

As we can observe in the formula, the *degree of anonymity* is obtained dividing the *effective anonymity set size* by the maximum entropy for a given number of

subjects. This *degree* evaluates how much anonymity is provided by a system independently from the number of users. Given a certain number of subjects, the computation of the *degree of anonymity* gives an idea on how close the anonymity is to the maximum achievable.

Both metrics are computed using the same information, and one can trivially be computed from the other. The difference is, however, that the *effective anonymity set size* ties the anonymity to the actual number of users in the system; while the *degree of anonymity* makes abstraction on the number of users and focusses on the performance of the system (i.e., how close it is to the maximum achievable anonymity).

2.6 Example: Crowds

In this section, we apply the anonymity metrics described in the previous section to Crowds [RR98], a system designed to provide anonymous access to web pages. To achieve this goal, the designers of Crowds introduced the notion of *blending into a crowd*: users are grouped into a set, and they forward requests within this set before the request is sent to the web server. The web server cannot know from which member the request originated, since it gets the request from a random member of the crowd, who is forwarding the message on behalf of the real originator. The users (members of the crowd) are called *jondos*.

The system works as follows: when a *jondo* wants to request a web page it sends the request to a second (randomly chosen) *jondo*. This *jondo* will, with probability p_f, forward the request to a third *jondo* (again, randomly chosen), and will, with probability $(1-p_f)$ submit it to the server. Each *jondo* in the path (except for the first one) chooses to forward or submit the request *independently* from the decisions of the predecessors in the path.

Communication between *jondos* is encrypted, and the final request to the server is sent in cleartext. Every *jondo* can observe the contents of the message (and thus the address of the target server), but it cannot know whether the predecessor is the originator of the message or whether he is just forwarding a message received by another member.

2.6.1 Attack Model

We calculate the degree of anonymity provided by Crowds to its users with respect to colluding crowd members, that is, a *set of corrupted jondos that collaborate in order to disclose the identity of the jondo that originated the request.* The assumptions made on the attacker are:

- *Internal*: The attacker controls some of the entities which are part of the

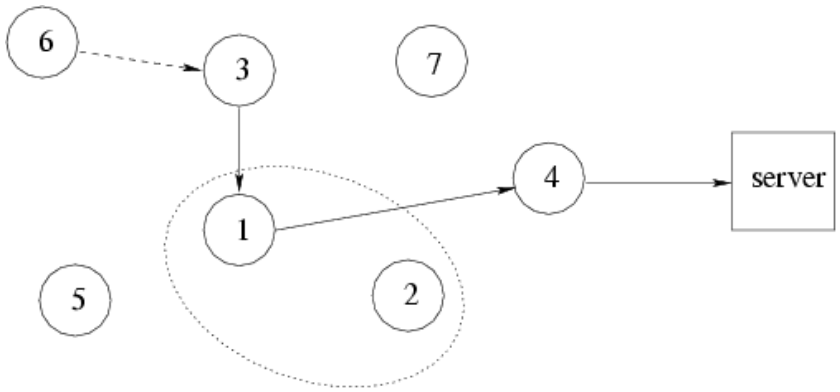


Figure 2.3: Example of a Crowds System with 7 *jondos*

system.

- *Passive*: The corrupted *jondos* can listen to communication. Although they have the ability to add or delete messages, they do not gain extra information on the identity of the originator by doing so.
- *Partial*: We assume the attacker controls a limited set C of *jondos*, and cannot perform any traffic analysis on the rest of the system.
- *Static*: The set of jondos controlled by the adversary is fixed.

2.6.2 Effective Anonymity Set Size

Figure 2.3 shows an example of a Crowds system. In this example the *jondos* 1 and 2 are controlled by the attacker, i.e., they are *colluding crowd members*. An honest *jondo* creates a path that includes at least one corrupted *jondo*.[1] The adversary wants to know which of the *jondos* is the real originator of the message.

In a general Crowds network, let N denote the number of members of the crowd, C the number of malicious collaborators, p_f the probability of forwarding and p_i the probability of being originator of a request assigned by the attacker to *jondo* i. From [RR98] we know that, under the described attack model, the probability assigned to the predecessor of the first malicious *jondo* in the path (for simplicity, let this *jondo* be number *C+1*) equals:

[1] If the path does not go through a corrupted *jondo*, the attacker cannot get any information.

$$p_{C+1} = \frac{N - p_f(N - C - 1)}{N} = 1 - p_f \frac{N - C - 1}{N} \ .$$

The probabilities assigned to the colluding *jondos* remain zero, and assuming that the adversary does not have any extra information about honest nodes, the probabilities assigned to those members are:

$$p_i = \frac{1 - p_{C+1}}{N - C - 1} = \frac{p_f}{N} \ , \quad C + 2 \leq i \leq N \ .$$

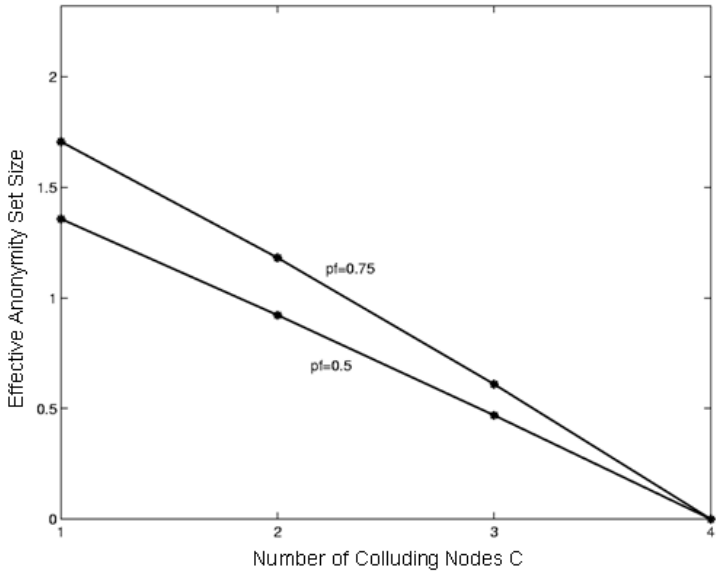
Applying the formula of the entropy presented in Sect. 2.5.1, the effective anonymity set size under these attack conditions can be computed as:

$$H(X) = \frac{N - p_f(N - C - 1)}{N} \log_2 \left[\frac{N}{N - p_f(N - C - 1)} \right] + p_f \frac{N - C - 1}{N} \log_2 \left[\frac{N}{p_f} \right] \ .$$

As we can see, the *effective anonymity set size* for Crowds is a function of N, C and p_f. In order to show the variation of $H(X)$ with respect to these parameters we chose $p_f = 0.5$ and $p_f = 0.75$, and $N = 5$, $N = 20$ and $N = 100$. The effective anonymity set sizes for these values are shown in Figs. 2.4 and 2.5 as a function of the number C of colluding *jondos*, which takes values between 1 and $N - 1$. Note that if $C = 0$ there is no adversary, and the effective anonymity set size is maximum ($\log_2(N)$); if $C = N$ the adversary controls all *jondos*, leaving none to attack.

As we can see in Figs. 2.4 and 2.5, the effective anonymity set size decreases almost linearly with the number of colluding *jondos* (controlled by the adversary), down to zero when the adversary controls $N - 1$ *jondos* (and is thus able to uniquely identify messages sent by the remaining *jondo*). We can also see in the figures that the effective anonymity set size is bigger for higher values of p_f. This indicates a tradeoff between anonymity and performance, as higher p_f implies more intermediate *jondos* in the communication path, and therefore more delay. Regarding the number of members of the crowd, it is clear that the larger the crowd, the higher the value of the effective anonymity set size.

Note that the figures presented correspond to a particular type of attack (namely, the "collaborating *jondos* attack" as described in [RR98]). The variation of anonymity towards adversaries capable of deploying other attacks may be very different from the results presented in this example. The same applies to the results shown in the next section for the degree of anonymity.



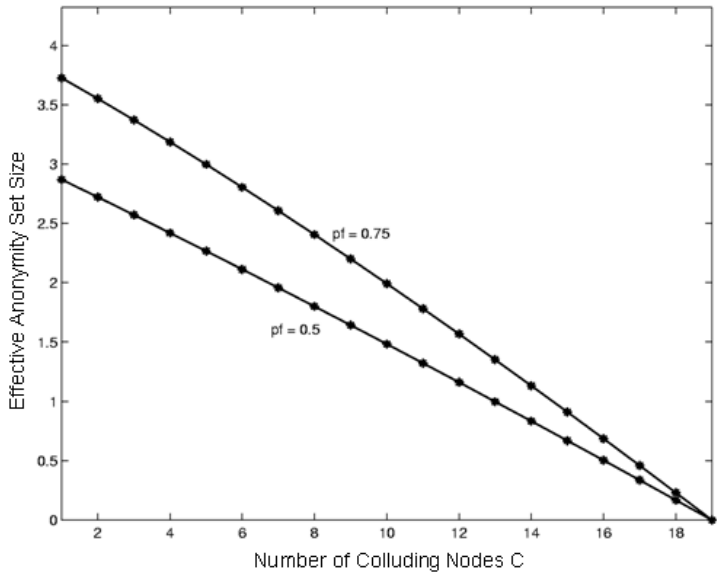


Figure 2.4: Effective Anonymity Set Size for Crowds for $N = 5$ and $N = 20$

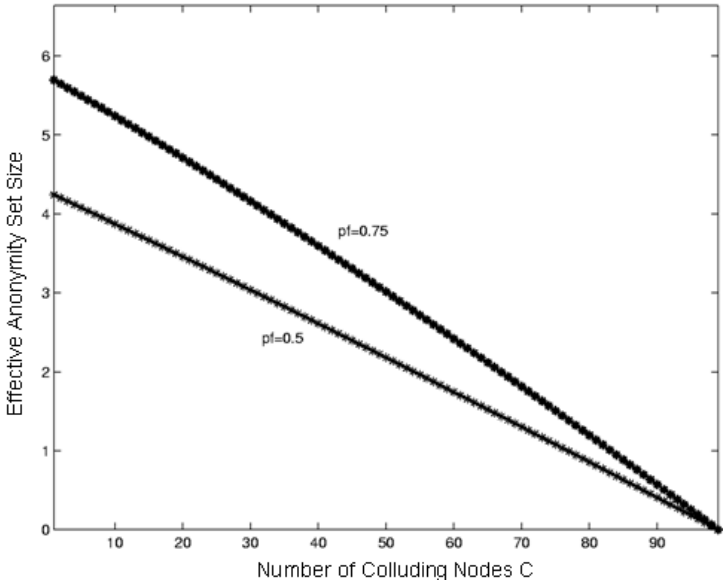


Figure 2.5: Effective Anonymity Set Size for Crowds for $N = 100$

2.6.3 Degree of Anonymity

The *degree of anonymity* is obtained normalizing the effective anonymity set size with respect to the maximum entropy, H_M. Taking into account that the size of the anonymity set is $N-C$ (the C colluding jondos are not part of the anonymity set), H_M equals:

$$H_M = \log_2 (N - C) \ .$$

According to the formulas presented in Sect. 2.5.3, we compute the *degree of anonymity*, d. Figures 2.6 and 2.7 represent the degree of anonymity for 5, 20 and 100 crowd members, respectively. As in the figures of the *effective anonymity set size*, the probability of forwarding p_f has been set to 0.5 and 0.75, and the variable in the x axis is the number C of corrupted *jondos*.

We can see in the figures that d decreases with the number of collaborating *jondos* and increases with p_f. The variation of d is very similar for systems with different number of users.

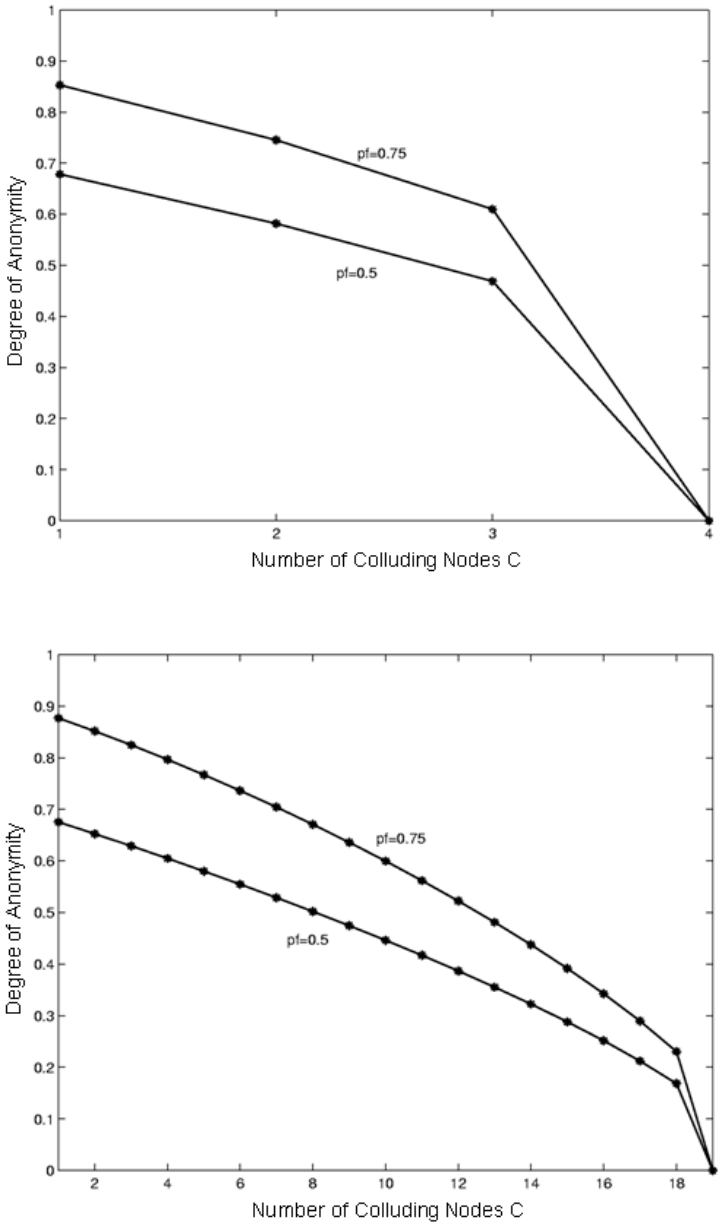


Figure 2.6: Degree of Anonymity for Crowds for $N = 5$ and $N = 20$

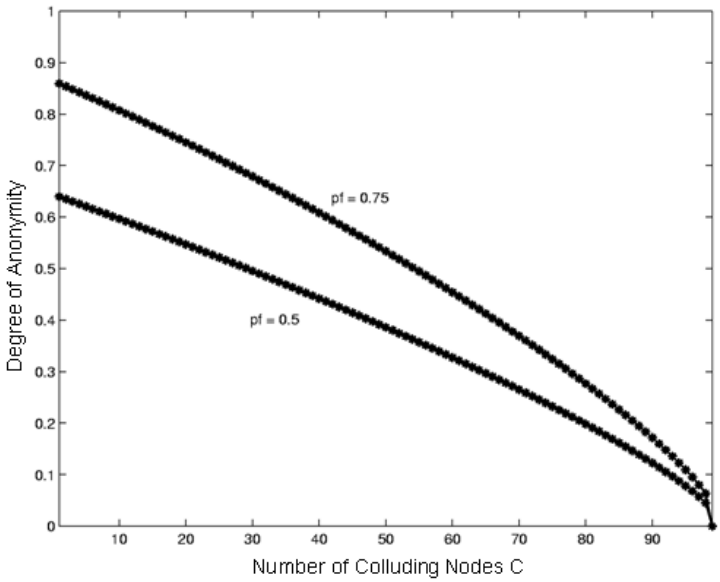


Figure 2.7: Degree of Anonymity for Crowds for $N = 100$

If we compare the results of the two proposed metrics, we can see that while the *effective anonymity set size* presents large variations in $C = 1$ for different values of N, the *degree of anonymity* for all three crowds systems ($N = 5, 20, 100$) takes values between 0.8 and 0.9 for $p_f = 0.75$, and between 0.6 and 0.7 for $p_f = 0.5$ (evaluated in $C = 1$). Also, the *effective anonymity set size* decreases almost linearly, while the *degree of anonymity* is concave.

The information provided by these metrics can be combined to give a better estimation of the anonymity offered to users. The *effective anonymity set size* gives a quantitative measure of the (un)certainty of the attacker with respect to the identity of a subject, while the *degree of anonymity* indicates the performance of the anonymity system relative to the best it can do for the given number of users. Note that, while the values of the *effective anonymity set size* significantly increase for $N = 100$ with respect to $N = 20$ or $N = 5$, the *degree of anonymity* slightly decreases for $N = 100$ in comparison with $N = 20$ and $N = 5$. This can be explained as follows: although the *effective anonymity set size* increases due to the increase of potential originators of a communication, the adversary is able to get more information (i.e., reduce his uncertainty with respect to his *a priori* knowledge) from the network with more nodes, because the distribution of probabilities is less uniform.

2.7 Conclusions

Several solutions for anonymity services have been proposed and implemented in the past. However, the problem of how to quantify the actual anonymity they provide had not yet been studied thoroughly. We propose a general measurement model from which we present two flavors of information theoretic metrics. These metrics provide answers to the research questions formulated in the introduction of the chapter: they provide a general method to measure anonymity, to compare different systems, to evaluate the effectiveness of attacks on anonymity, to quantify gains and losses in anonymity which take into account the partial or statistical information obtained by an adversary.

With these metrics we can quantify the *effective anonymity set size* and the *degree of anonymity* provided by a system in particular attack circumstances. We applied the metrics to Crowds, an existing solution for anonymous communication, and discussed the results obtained. The metrics showed to be useful for evaluating a system, and could be used in the design and comparison of anonymity systems and their robustness towards attacks.

The metrics proposed can be adapted to systems where anonymity can be defined in terms of unlinkability. Anonymous transactions are abstracted as IOIs (*Items Of Interest*); the anonymity of the subjects who are either originators or responders of a transaction can be computed applying the proposed formulas.

Anonymity metrics provide relevant information on the anonymity of concrete subjects in concrete attack scenarios. In order to know more about the robustness of an anonymity system, we need to make multiple measurements in different scenarios.

The model is based on the probabilities adversaries assign to subjects; finding these probability distributions in real situations is however not always easy.

One of the questions that remains open is the sufficient level of anonymity a system should provide to be privacy enabled. The answer to this question is different for each system, as it depends on the (legal and social) consequences of the breach of privacy in particular scenarios.

A subject of future work on anonymity metrics would be to explore the possibility of using other information theoretic metrics such as mutual information or min-entropy. The min-entropy (instead of Shannon's entropy) may be useful in scenarios in which the priority is to avoid that any subject appears linked to an IOI with a relatively high probability.

Chapter 3

Taxonomy of Mixes and Dummy Traffic

Nothing is built on stone; all is built on sand,
but we must build as if the sand were stone.
– Jorge Luis Borges

3.1 Introduction

In this chapter, we present an analysis of two building blocks of anonymous services. In particular, we study the taxonomy of mixes and dummy traffic policies (which consist, as we will see in Sect. 3.6, in inserting fake messages in the network in order to thwart traffic analysis). The goal of the chapter is to bring together the different issues related to the analysis and design of mix based anonymous communications.

The research questions addressed in this chapter are:

- What sorts of mixes have been proposed in the literature? How can we classify them? Is there a framework or a set of parameters to describe mixes?
- Are there mix designs we have not yet thought about which may provide good anonymity properties? Can we design mixes which are more robust against well known attacks?
- Which are the relevant parameters we must take into account when designing a dummy traffic policy? What is the impact of these parameters in the effectiveness of the dummy policy?

We summarize in this chapter the content of two research papers published in international workshops: [DP04c] and [DS03b]. The original contributions presented here are:

- Identification of the set of parameters of mixes and dummy policies which have an impact on anonymity.
- Analysis and classification of the different mix techniques and dummy traffic strategies.
- Proposal of a mathematical model (Generalized Mix Model) for pool mixes which eases the design, analysis and comparison of different mix implementations by representing mixes as a function.
- The Generalized Mix Model allows for easy computation of the anonymity provided to messages routed through the mix.
- The Generalized Mix Model allows for the design of mixes with arbitrary mix functions.
- As a result of the possibilities opened by this the Generalized Mix Model, a randomization of the mixing strategy is proposed. These mixes are called *binomial mixes* and they are more robust towards certain attacks than previous designs at no additional cost.

We first introduce the basic concept of a mix and the related work done on mixes in Sect. 3.2. Pool mixes are analyzed in the next two sections. Section 3.5 discusses the issues related to continuous mixes, and dummy traffic is presented in Sect. 3.6. Finally, Sect. 3.7 presents the conclusions of the chapter.

3.2 What is a Mix?

3.2.1 The Chaumian Mix

Chaum proposed the first mix design in 1981 as a technique to implement untraceable electronic mail [Cha81]. Its purpose was to hide the correspondences between the items in its input and those in its output. The order of arrival was hidden by outputting the uniformly sized items in lexicographically ordered batches. A public key cryptosystem was proposed to provide semantic secrecy.

The mix generates a pair of keys K and K^{-1} from a suitable randomly generated seed. The public key K is made known to the other users, while the private key K^{-1} is never divulged. The encryption of X with key K will be denoted $K(X)$, and is just the image of X under the mapping implemented by the cryptographic algorithm using key K. The increased utility of these algorithms over

conventional algorithms is explained because the two keys are inverses of each other, in the sense that $K^{-1}(K(X)) = K(K^{-1}(X)) = X$.

A message X is *sealed* with a public key K so that only the holder of the private key K^{-1} can discover its content. If X is simply encrypted with K, then anyone could verify a guess that $Y = X$ by checking whether $K(Y) = K(X)$. This threat can be eliminated by attaching a large string of random bits R to X before encrypting. The sealing of X with K is then denoted $K(R, X)$. Chaum's approach is based on two assumptions:

1. No one can determine anything about the correspondences between a set of sealed items and the corresponding set of unsealed items, or create forgeries without the appropriate random string or private key.

2. Anyone may learn the origin, destination(s), and representation of all messages in the underlying telecommunication system and anyone may inject, remove, or modify messages.

A participant prepares a message M for delivery to a participant at address A by sealing it with the addressee's public key K_a, appending the address A, and then sealing the result with the mix's public key K_1. The left-hand side of the following expression denotes this item which is input to the mix:

$$K_1(R_1, K_a(R_0, M), A) >>> K_a(R_0, M), A.$$

The $>>>$ denotes the transformation of the input by the mix into the output shown on the right-hand side. The mix decrypts its input with its private key K^{-1}, throws away the random string R_1, and outputs the remainder.

3.2.2 Related Work on Mixes

Since Chaum proposed the first mix design for untraceable email in 1981 [Cha81], several schemes have been suggested in the literature:

- **Low latency mixing:** In 1991, Pfitzmann *et al.* [PPW91] proposed a system to anonymize ISDN telephone connections. Later, the design was generalized to implement *Real-Time MIXes*, a low-latency mix network, in [JMP+98]. ISDN and Real-Time MIXes evolved into an anonymous web browsing design called Web Mixes [BFK01]. This design has been implemented as a web anonymizing proxy called JAP [JAP].

- **Continuous mixes:** Continuous, or *Stop-and-Go* (S-G) mixes were proposed by Kesdogan *et al.* [KEB98]. These mixes delay each message independently according to an exponential distribution. Continuous mixes are explained in detail in Sect. 3.5.

- **Remailers:** Mixes can be used to implement anonymous email applications. The most popular mix-based anonymous email designs are Babel [GT96] and Mixmaster [UMS03], which has been widely deployed. More recently, Danezis *et al.* proposed a remailer design called Mixminion [DDM03] which improves the security and functionality of its predecessors.

- **Anonymous communication infrastructures:** Extensive research has also been done in the field of general purpose, low-latency, circuit-based mix networks. In 1996, Goldschlag *et al.* proposed Onion Routing [GRS96, RSG98, PSG00, STRL00, RSG98, STRL00], a general purpose anonymous infrastructure which routed traffic on top of TCP/IP. The Freedom network, designed by Ian Goldberg *et al.* [Gol00, BSG00, BGS01], is a commercial implementation of Onion Routing, which was operated by Zero Knowledge [Zer]. Dingledine *et al.* [DMS04] proposed in 2004 an improved anonymous communication network called TOR.

- **Verifiable mix constructions:** Much effort has been put into designing mixing algorithms with universal verifiability properties. More information on these schemes can be found in [Abe98, Jak99, JJR02, JM98, Jak98, FS01, Nef01].

3.2.3 Mix Functionality

Mixes are the nodes of anonymous communication networks. The mix takes a number of input messages, and outputs them in such a way that it is hard to link an output to the corresponding input (or an input to the corresponding output) with certainty. In order to achieve this goal, the mix changes the *appearance* (by encrypting and padding messages) and the *flow* of messages (by delaying and reordering).

Bitwise Unlinkability

In order to provide *bitwise unlinkability*, messages should *look* different when entering and leaving the mix. This is achieved using a public key cryptosystem as introduced in Sect. 3.2.1.

Messages are padded to a fixed size before being encrypted under the public key of the recipient. The generation of this padding differs for the existing mix implementations. For example, Mixmaster version one appends random noise to the message, while in Mixmaster version three the noise is generated using a secret included in the header of the (encrypted) message, and thus shared by the sender and the recipient of the message. In Mixmaster version three, the appended noise

is predictable (as it depends on the message and the shared secret) and it is used by the recipient to check the integrity of the message.

When messages are routed through several mix nodes, *layered* or *nested encryption* is used: the message is encrypted recursively with the public keys of the mixes in its path to the recipient in inverse order (starting by the exit mix); every node in the path strips a layer of encryption using its private key to find the next node and a new encrypted payload, as it is shown in Fig. 3.1.

Optionally, the message that leaves the last mix may be encrypted with the recipient's public encryption key. This encryption is used to preserve the confidentiality (besides the anonymity) of the message.

Modifying the Flow of Messages

We need to change the flow of messages in order to make the linking of an input and an output difficult for an attacker. Modifying the flow of messages is not an easy problem, especially when the delay of the messages is constrained by real-time requirements, as anonymity has to be traded with delay. Typically, any practical mix reveals a probabilistic relationship between inputs and outputs. In the following sections of this chapter, we give an overview on the options that have been explored in order to anonymize the flow of messages.

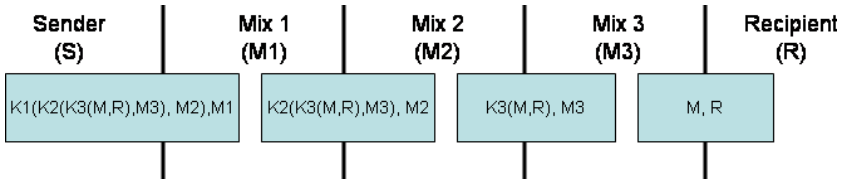


Figure 3.1: Nested Encryptions

3.2.4 Mix Networks

Using a single mix for providing an anonymity service presents some limitations. The single point of failure makes the system very vulnerable to denial of service attacks (if the mix can be taken out of service the whole anonymity system would be shut down). Moreover, the mix knows the link between sender and recipient, so that it has to be trusted not to reveal it.

In order to enhance the robustness and trust distribution of a mix-based anonymity system, messages are routed through several mixes. We can distinguish three network topologies:

- Cascades: In his original mix paper [Cha81], Chaum already proposes the connection of several mixes in a cascade (i.e., one after the other) as way to improve the robustness of the anonymity system. In this topology, the route that a message must follow is fixed.

- Free route networks: In free route networks all mixes are interconnected. The sender may choose a random path, selecting a subset of mixes in a random order to route his messages.

- Restricted route networks: Restricted route networks were proposed by Danezis in [Dan03a]. In these networks mixes are connected to a number of other nodes, but not every path is possible in the network. Note that both cascades and free route networks are particular cases of restricted route networks.

The advantages and disadvantages of the different network topologies are discussed in our paper [BDD+05], which was published at the *Workshop on Privacy Enhancing Technologies* 2004.

3.3 Pool Mixes

Pool mixes are based on the original Chaumian mix [Cha81], so they process messages in batches (i.e., groups). They collect messages for some time, place them in the pool (internal memory of the mix), and forward them (in random order) when the flushing condition is fulfilled. Each cycle of collecting inputs, placing them in the pool and flushing part of the messages in the pool is called a *round*. We could describe pool mixes as a buffer:

- At the beginning of the round, the pool of the mix contains P messages that were left from the previous round $x_p[0]..x_p[P-1]$.

- A elements arrive to the mix $x_a[0]..x_a[A-1]$, and are added to the pool $x_p[0]..x_p[P+A-1]$.

- The mix chooses uniformly at random B elements from the pool, which are forwarded at the end of the round, $x_b[0]..x_b[B-1]$. The pool stays with $P+A-B$ messages $x_p[0]..x_p[P+A-B-1]$.

The aspects that we should take into account when designing and analyzing a pool mix are the *flushing condition* and the *pool selection algorithm*.

3.3.1 Flushing Condition

We can distinguish two types of pool mixes according to the flushing (or forwarding) condition: *timed mixes* send messages every fixed internal time, called *timeout*. *Threshold mixes* send messages when they have collected a certain amount of messages, called the *threshold*. Some mix designs, such as Mixmaster [UMS03], combine the two mechanisms: they flush when the *timeout* expires only if the *threshold* has been reached. The cycle of collecting and flushing messages is called a *round*.

So far, the mixes that have been implemented have a *fixed* timeout or threshold. It would be interesting to study the properties of mixes that choose the threshold or the timeout according to a *random* distribution.

3.3.2 Pool Selection Algorithm

The performance of a pool mix (in terms of delay and anonymity) is mainly determined by the pool selection algorithm. In Chaum's design, the mix flushes all the messages it contains (i.e., it keeps zero messages in the pool). Later, the concept of *pool* was added to the mix, extending the original mix to keep a number of messages (instead of flushing all of them). In the first stage, the proposals of mixes kept a fixed number of messages in the pool (*static pool mixes*). Later on, mixes that kept a variable number of messages were designed (*dynamic pool mixes*; e.g., Mixmaster).

Pool algorithms enhance the anonymity (compared to Chaum's mix) by extending the anonymity set size to, potentially, an infinite number of users. Nevertheless, it should be noted that the probability distributions obtained by an attacker trying to trace a message will not be uniform for all possible senders (or recipients) of messages.

The parameters that should be taken into account when designing a pool selection algorithm are the number of messages kept in the pool (which can be fixed or variable, e.g., percentage of the total number of messages at the time of flushing); and the number of messages sent (which can also be fixed or variable). Table 3.1 in Sect. 3.7 summarizes these parameters.

3.4 Generalized Mix Model

As we have emphasized, one of the most important parameters of a pool mix is its *pool selection algorithm*. Intuitively, the pool selection of a mix is the algorithm for collecting the messages to be mixed together and forwarding them to the next hop. Naturally, this influences both the anonymity and message delay properties of the mix.

In the past, batching strategies of mixes were often described by giving the algorithm which determines when to flush the mix and how many (and which) messages to forward during the flush. In this section, we present a formal model for describing mixes, which represents the pool selection algorithm of the mix at the time of flushing. This model was proposed by us and published at the *Workshop on Privacy Enhancing Technologies* 2003 [DS03b].

In this section, we first show how to represent existing pool mixes in our model. Then, we point out the possibility of designing new mix functions. Finally, we introduce *binomial mixes*. A detailed analysis of anonymity metrics for pool mixes under passive and active attacks is presented in Chapter 4 and Chapter 5, respectively.

3.4.1 Representation of Classical Pool Mixes in the Model

Let us examine some existing mixes. Several mixes have been described in the literature (see survey in [SDS02]): threshold mix, timed mix, timed pool mix and the timed dynamic pool (Cottrell) mix. We now seek to express mixes, just as an implementer would, as functions $P : \mathbb{N} \to [0, 1]$ from the number of messages inside the mix to the fraction of messages to be flushed. We note that just this function expresses the pool selection algorithm of a mix, so we also need to specify the flushing condition. The random variable n represents the number of messages contained in the mix at the time of flushing.

Figures 3.2, 3.3, 3.4 and 3.5 present the function $P(n)$ for the following classical mixes:

- Timed mix: This mix is represented in Fig. 3.2. It flushes all the messages it contains at the time of flushing. Therefore, the percentage of sent messages is always 100%, i.e., $P(n) = 1$.
- Timed pool mix: This mix (in Fig. 3.3) keeps a constant number of messages, N_p, in the pool ($N_p = 20$ in this example), and flushes periodically. If the mix contains less than N_p messages at the time of flushing, it will not output any message. When it contains more, it outputs $n - N_p$ messages, that means that the percentage of sent messages can be expressed as: $P(n) = 1 - N_p/n$.
- Timed dynamic pool mix (Cottrell mix): This mix outputs messages at the timeout only when the number of messages is greater than a threshold N_p, as shown in Fig. 3.4. The number of output messages is a fraction, f, of the difference between the number of messages inside the mix and the value of the threshold of the pool, $f(n - N_p)$ ($f = 0.7$ and $N_p = 20$ in the example). In the figure, the function that represents the percentage of sent messages is $P(n) = f(1 - N_p/n)$.

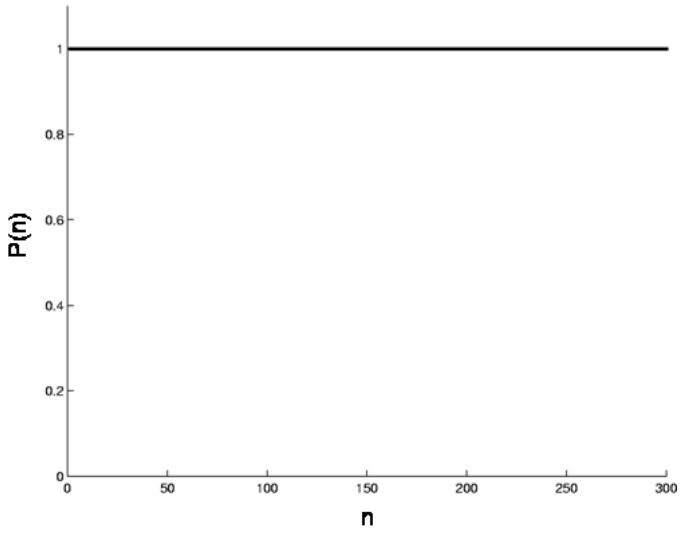


Figure 3.2: Representation of a Timed Mix

- Threshold pool mix: We have noted above that each mix is a function, together with a time period (T) which specifies how often we flush the mix. If we set $T = 0$ and let the function $P(n) = 0$ everywhere apart from the threshold, we can express threshold mixes as well as timed mixes. Thus, such a mix is represented by a single dot (at $(N, 1)$ for a threshold mix, or $(N,1-\frac{N_p}{N})$ for a threshold pool mix that keeps N_p messages in the pool) as it is shown in Figure 3.5. The mix shown in the figure is a threshold pool mix with threshold $N = 100$ and pool size $N_p = 50$.

Note that the reason why we have been able to express all the above mixes in this framework is because they are stateless, i.e. the fraction (and therefore the number) of messages to be flushed depends only on the number of messages in the mix, but not, say, on the number of messages flushed during the previous round.

3.4.2 New Functions

The natural way to proceed is to say that a mix is an arbitrary function from the number of messages inside the mix to the percentage of messages to be flushed. What does this buy us?

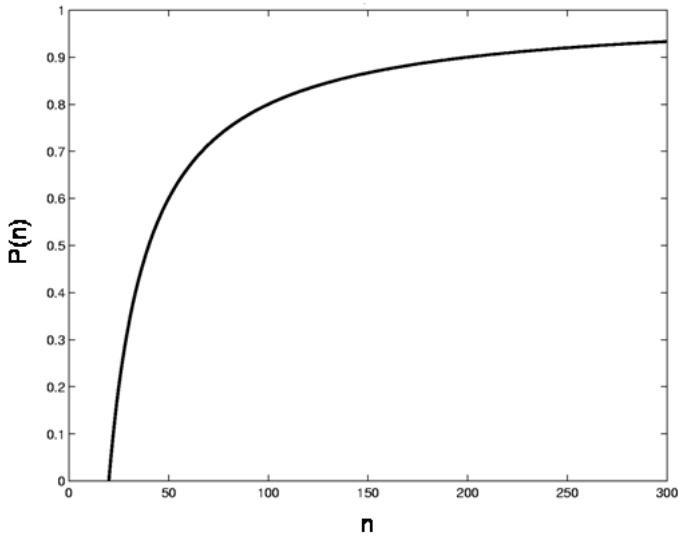


Figure 3.3: Representation of a Timed Pool Mix

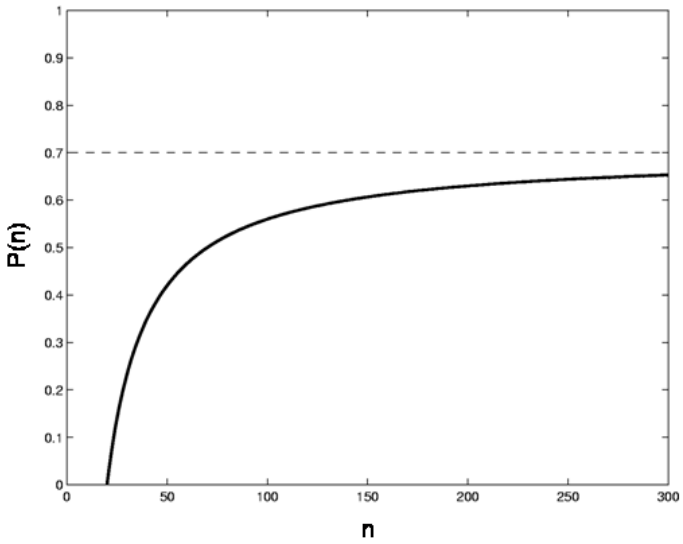


Figure 3.4: Representation of a Timed Dynamic Pool Mix

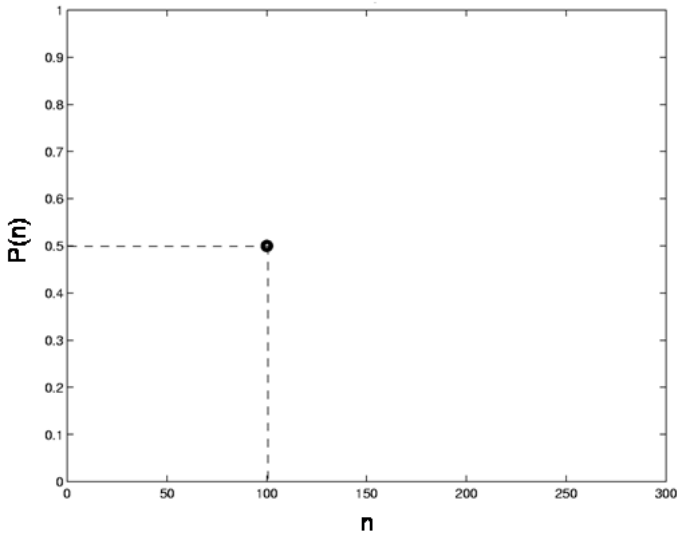


Figure 3.5: Representation of a Threshold Pool Mix

Throughout the mix literature, a tradeoff between message delay and anonymity can clearly be seen. Indeed, as Serjantov and Danezis showed in [SD02], the pool mix gains more anonymity from higher average delay as compared to the Chaumian mix. As we will show in Chapter 4, high values of the function $P(n)$ favor lower delays (and lower anonymity), while low values of the function provide higher anonymity (and longer delays).

Expressing the mix pool selection algorithm as a function allows us to define an arbitrary tradeoff between anonymity and message delay. As example, we present in Fig. 3.6 a mix pool selection algorithm based on the normal cumulative distribution function (note that such a pool selection could not be expressed in the algorithmic way of describing classical pool mixes).

This function presents some nice features. The low values of the function for small n increase the delay under low traffic conditions in order to keep a good level of anonymity. As traffic increases and anonymity becomes higher, the function grows smoothly in order to reduce the latency of the messages going through the mix.

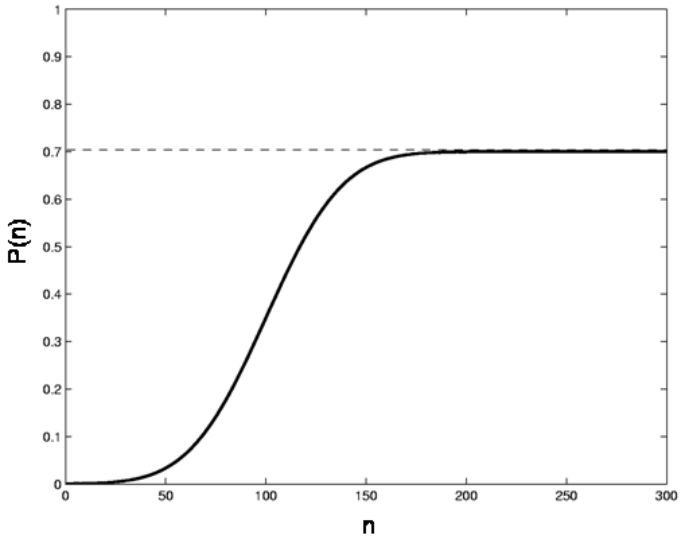


Figure 3.6: Example of New Function for a Pool Mix

3.4.3 Binomial Mixes

Here, we add randomness to the mix. Suppose we treat the result of the function $P(n)$ not as a fraction, but as a probability. We can then use it as the bias of a coin, which we toss for each message inside the mix. A head indicates that this message should be sent out during this round, and a tail that it should remain in the mix.

Let s denote the variable that represents the number of messages sent by the mix when it flushes. On average, $s = nP(n)$; but s follows a binomial distribution, with variance equal to $np(1-p)$, where p is the result of the function $P(n)$. Due to this property, we call this proposed mix *binomial mix*.

The added randomness of the binomial mix can be used to increase the uncertainty of the attacker, as we show in Chapter 4.

3.5 Continuous Mixes

The idea of continuous mixes (also called *Stop-and-Go* mixes) was first proposed by Kesdogan *et al.* [KEB98]. In this design, the users generate a random delay from an exponential distribution, and add this delay to the headers of the message. The mix holds the message for the specified delay and then forwards it.

The messages are reordered by the randomness of the delay distribution. This mix sends messages continuously: every time a message has been kept for the delay time, it is forwarded by the mix.

3.5.1 Reordering Technique

In Kesdogan's original idea, the delay is chosen by the user from an exponential distribution. The exponential distribution has the advantage of being memoryless, but other distributions, such as the uniform distribution (in which the variance of the delay can be larger), may also be taken into account. A thorough study must be carried out in order to find out which distribution provides the best anonymity properties for the expected working context of the mix (traffic load, traffic pattern, and delay constraints).

3.5.2 Analysis of Continuous Mixes

The main advantage of these mixes is that the delay does not depend on the traffic that arrives to the mix. This means that tight delay constraints can be implemented by this mix, regardless of the current load of the mix (which may be useful for applications in which a small delay is more important than providing a high level of anonymity).

Moreover, when the message is routed through a mix network, the user can know in advance the amount of time it will take to the message to arrive to every mix on the path (and to the recipient), since he is the one who generates the amount of time his message will be delayed at each mix.

On the other hand, the anonymity provided to the users may go to low levels if the number of users decreases during a certain period of time (as we can see in Chapter 6). We must not forget that there is always a tradeoff anonymity/delay, and if we bound the delay we may drop to low levels of anonymity under certain conditions (in this case, low traffic conditions).

This design may be appropriate for systems with stable incoming traffic patterns, in which the anonymity is guaranteed by a (more or less) constant traffic rate. Systems with variable number of users and with changing traffic conditions risk to result in low levels of anonymity during quiet traffic periods, as shown in [DSD04].

These mixes are also vulnerable to *blending* or $n-1$ attacks [SDS02]. This active attack is deployed by an attacker who is able to delay the messages before entering the mix. The attacker selects a *target* message he wants to trace, and delays all the other messages. In a continuous mix, this would result in the attacker being able to trace the target message, given that (with an arbitrarily high probability) the attacker can succeed in making the message going through the mix when it does not contain any other messages (the message is not *mixed*).

This attack can be prevented, or at least detected, using additional mechanisms. Kesdogan proposes to add a timestamp to the messages (note that the user knows the expected time of arrival of the message to every mix); the mixes discard all messages that contain an old timestamp.

Dummy traffic (Section 3.6) can also be used both to prevent and to detect *blending* attacks. See Section 3.6.3 to find a description on how dummy traffic can be used to detect and react when a mix is subject to active attacks.

3.6 Dummy Traffic

A dummy message is a *fake* message introduced in a mix network in order to make it more difficult for an attacker to deploy passive and active attacks. Dummy messages are normally generated by the mixes (although users may also generate dummies, which increases the anonymity level of the mix network and prevents end-to-end intersection attacks [BL02]); they have as destination another mix, instead of a real recipient. Dai proposed the Pipenet network [Dai96], a system in which the traffic is constant: the links between mixes are padded with dummy messages whenever the real traffic is not sufficient to fill them. This system provides not only anonymity, but also unobservability, since an observer of the network cannot tell whether there are real messages traveling in the network or not. Unfortunately, the system is not practical due to the enormous amount of resources it needs.

The generation and transmission of dummy traffic has a cost, and it is therefore very important to find the right balance on the number of dummies that should be created in a mix network. The rest of this section studies the possible choices we can make when designing a dummy policy.

3.6.1 Generation of Dummies

The first question that arises when designing a dummy traffic policy is whether the dummies generated should depend on the traffic or not. Generating dummies depending on the traffic may make a more efficient use of resources, and may also help to prevent active attacks. On the other hand, this dependency can be exploited by active attackers to maximize the effectiveness of their attacks by generating their own messages in such a way that they minimize (or control) the number of dummies generated by mixes.

One of the issues that needs to be decided is the average number of dummies we want to generate (for pool mixes we will choose an average number of dummies per round, while in continuous mixes we will generate dummies per fixed time unit). These dummies can be generated following a deterministic or random distribution. Random distributions increase the uncertainty of the attacker,

especially when combined with binomial mixes, as pointed out in [DSD04].

Continuous Mixes

These mixes may generate a certain number of dummies every period of time, selecting their delay (amount of time they are kept in the mix from their generation until the moment in which they are sent) from a random distribution (typically an exponential distribution). This is the approach followed by *Reliable*, one of the mixes that composes the Mixmaster network. The probability density function of the exponential distribution is expressed as:

$$f(t) = \lambda e^{-\lambda t} \ .$$

Other dummy policies may be explored. For example, the mix could keep always one dummy inside, and generate a new one (with its corresponding delay) when the dummy is sent. Another policy would be that the mix decides every certain amount of time on whether to generate a dummy or not. Traffic dependent dummy policies can be used to compensate for traffic fluctuations.

Pool Mixes

The design of dummy policies for pool mixes implies making decisions on the following issues:

- the dependency on the traffic load;
- the average number of dummies generated per round;
- the distribution followed to select the number of dummies in a particular round (binomial, uniform, geometric, etc.);
- whether the dummies are inserted in the pool or at the output;
- route length and selection of path for the dummies.

Insertion in the Pool: With this technique, the mix inserts the dummies it generates each a round in the pool. These dummies are treated as real messages by the mix after being placed in the pool (note that all other mixes until the last in the path of the dummy cannot distinguish it from a real message).

Insertion at the Output: If the mix is to insert the dummies at the output, then it adds the dummies to the batch of real messages taken from the pool. The dummies are never stored in the pool of the mix.

The advantages and disadvantages of these two dummy insertion options have been discussed in [DSD04], and are presented in Chapter 4. Here, we summarize the conclusions:

- Inserting the dummies in the pool provides less anonymity and less delay than inserting them at the output.
- When dummies are inserted at the output, binomial mixes offer more protection against the $n-1$ attack than deterministic mixes.
- Inserting dummies in the pool protects deterministic mixes better than inserting them at the output when an $n-1$ attack is deployed.

3.6.2 Route Length and Selection of Path

Dummy messages, just like real messages, travel in the mix network via a number of mixes. The route length of the dummy determines the number of mixes a dummy is going through. Regarding this issue, we should decide on the average number of mixes in the path of the dummy and on the distribution of this route length. Random distributions increase the uncertainty of the attacker with respect to a deterministic distribution (i.e., fixed number of mixes in the path) when the attacker wants to find out whether a message is a dummy or not.

Normally, a dummy is routed through several mixes randomly selected. The last mix in the path of a dummy discards it. Note that intermediate mixes (i.e., except for the first and last in the path of the dummy) cannot distinguish dummy messages from real messages.

3.6.3 RGB Dummy Policies

This dummy policy was proposed by Danezis and Sassaman in [DS03a]. The goal is to detect and counter active attacks (such as the $n-1$ attack). The basic idea of this dummy policy is that the mix generates dummies that – after being routed through the network – end in the mix that generated them. If the mix receives less dummy messages than expected, it assumes that it is being subject to an attack, and it reacts by stopping its functioning until the attack is no longer being deployed.

3.7 Conclusions

In this chapter, we have introduced a taxonomy for mixes and dummy traffic, in order to make explicit the issues we must take into account when designing mixes and dummy traffic policies. These issues are summarized in Table 3.1 and Table 3.2. In Table 3.3 we indicate the parameters that should be taken into account for each type of mix. Note that the function $P(n)$ already expresses the number N_p of messages kept in the pool (for pool mixes) and the threshold N (for threshold mixes).

Table 3.1: Analysis and Design of Mixes

Change appearance of messages	- Select encryption and padding primitives
Alter the flow of messages	- Continuous or pool mix? - Real-time constraints?
Pool mixes	- Flushing condition: timed, threshold or both - Pool selection algorithm? ($P(n)$ in the GMM) - Deterministic or binomial?
Continuous mixes	- Delay distribution?
Anonymity provided by the mix	- Stable and unstable traffic patterns - High and low traffic loads - Different attacks and adversaries
Delay introduced by the mix	- Stable and unstable traffic patterns - High and low traffic loads
Attacks	Analyze the robustness of the mix against: - Passive attacks (e.g., traffic analysis attacks) - Active attacks (e.g., $n-1$ attacks)
Mix network	Topology: - Cascade - Free route network - Restricted route network

We have proposed a framework with which we can generalize classical pool mixes. This model is a powerful tool that gives us a new understanding of the batching strategies implemented by existing mixes. Also, new strategies that improve existing designs arise from the framework, which can implement a tailored anonymity/delay tradeoff that adapts to the fluctuations in the traffic load. We have proposed as example the cumulative distribution function.

Table 3.2: Analysis and Design of Dummy Traffic Policies

Dependent on incoming traffic	Yes / No
Dummy generation for continuous mixes	- Average number of dummies - Distribution in time of dummies
Dummy generation for pool mixes	- Average number of dummies - Distribution of the number of dummies - Insertion: pool or output
Root length of dummy messages	- Average number of intermediate mixes - Distribution of the route length
Selection of the path	- Algorithm to select intermediate mixes - Generating mix is last mix in the path?
Attacks	Dummy policy prevents active/passive attacks?

Table 3.3: Parameters of Mixes

Mix Type	$P(n)$	N_p	N	T	Delay function
Timed Mix	X			X	
Timed Pool Mix	X	X		X	
Timed Dynamic Pool Mix	X	X		X	
Threshold Mix	X		X		
Threshold Pool Mix	X	X	X		
Binomial Mix	X			X	
Continuous Mix					X

We have added randomness to the flushing algorithm, in order increase the uncertainty of adversaries trying to trace a message going though the mix. We will analyze the effects of this randomness in the next chapter.

The main questions that remain open are:

- If we add randomness to the threshold or timeout of a pool mix, what is the impact of this randomness?
- Can we find an optimal mix function? Here optimal being the one that maximizes anonymity for a given delay; or one that minimizes the delay while guaranteeing a minimal given anonymity.

Chapter 4

Anonymity Metrics for Mixes: Passive Attacks

I didn't do it, nobody saw me do it,
there's no way you can prove anything!
– Bart Simpson

4.1 Introduction

In this chapter, we analyze how anonymity metrics can be applied to generalized mixes when they generate dummy traffic. We indicate how to compute the recipient and sender anonymity and we point out some problems that may arise from the intuitive extension of the metric to take into account dummies. Two possible ways of inserting dummy traffic are discussed and compared.

The results presented here have been extracted from our original work *Reasoning about the anonymity of pool mixes that generate dummy traffic*, published at the *6th Information Hiding Workshop 2004* [DP04b].

The research questions that motivated this work are: how can we apply anonymity metrics to mix systems? How can we measure anonymity when dummies are being created or destroyed at a mix? How do different dummy strategies affect anonymity?

In Sect. 4.3 we detail the adversary considered in the anonymity measurements. Anonymity metrics are introduced in Sect. 4.4 and applied to deterministic mixes in Sect. 4.5, to binomial mixes in Sect. 4.6 and to continuous mixes in Sect.4.7. Section 4.8 explains how anonymity metrics must be applied to pool mixes when they generate or discard dummy traffic. Finally, Sect. 4.9 presents

the conclusions of the chapter.

4.2 Related Work on Passive Attacks

Passive attacks on anonymity networks are based on traffic analysis techniques and very difficult to detect. Therefore, it is important for these kind of systems to offer a high protection of anonymity against passive attacks. In order to put into perspective the traffic analysis attack for which anonymity computations are presented in this chapter, we briefly recall here the related work on passive attacks which is available in the literature. For an overview on traffic analysis attacks, details can be found in the work of Danezis [Dan04]. Some of the most relevant traffic analysis attacks are:

- **Brute force:** The adversary generates the set of potential communication partners of a user by following the message through the network and adding to the anonymity set all the possible mixing combinations. This attack is described in detail by Raymond in [Ray00].
- **Timing:** The adversary uses information on the delay applied by the mixes in every link in order to find a probabilistic relationship between incoming and outgoing messages.
- **Communication patterns:** The patterns of activity of the users (e.g., by relating it to the time zone) may reveal information on their communications.
- **Packet counting:** These attacks can be deployed on anonymity infrastructures such as Onion Routing [GRS96], given that they route connections over the same path. By counting packets, the adversary may be able to correlate incoming and outgoing streams.
- **Intersection:** This attack was presented by Berthold *et al.* in [BPS00], and consists of intersecting anonymity sets of consecutive messages sent or received by a user (assuming that users repeatedly communicate with a restricted set of participants). This attack model is also considered in [AKP03, KAP02] and [WALS03].
- **Statistical disclosure:** This attack is a variant of the intersection attack which improves the efficiency by requiring less effort from the attacker. It was proposed by Danezis in [Dan03b].

The attacks considered in this chapter correspond to timing attacks, given that they exploit the delay characteristic introduced by the mix. Our metrics can

also be applied to all the passive attacks described above whenever the adversary obtains probabilistic information on the sender or recipient of a message as a result of the attack.

4.3 Attack Model

The adversary model considered in this chapter corresponds to a *passive external static global attacker*, as defined in Chapter 2. The goal of the attacker is to get information on the link between the inputs and the outputs of the mix under attack, hoping to deanonymize messages. The adversary is *passive*, so he cannot generate, delete or modify messages. By *external*, we indicate that the adversary does not have access to the internal memory of the mix under attack. In this context, *global* means that the attacker can observe all input and output links of the mix (i.e., he can observe everything except the internal actions of the mix), and record all actions for future computations. The set of resources controlled by the attacker (the communication links) is fixed, so we characterize our adversary as *static*.

The adversary knows the internal algorithms of the mix (both for pool selection and dummy generation). He will use the information extracted from the observation of the mix to match inputs and outputs. We assume that the attacker has no *a priori* or contextual information on the messages going through the mix.

We assume our attacker cannot break secure implementations of public key encryption systems, and that random padding is generated so that all messages going though the mix system have the same length.

4.4 Anonymity Metrics for Mixes

The anonymity metric used to measure the degree of success of the adversary is the *effective anonymity set size*, which was explained in Chapter 2.

It is important to recall that the anonymity metric must be computed for each individual message going through the mix; i.e., either a *target input* or a *target output*. In order to get an idea of the anonymity performance of a mix, we must make multiple measurements in different traffic load circumstances, as shown in Chapter 6.

The anonymity provided by a mix can be computed for the incoming or for the outgoing messages. We call this *sender anonymity* and *recipient anonymity*.

Sender anonymity. In order to compute the sender anonymity, we want to know the effective size of the anonymity set of senders for a message forwarded

by the mix. Therefore, we compute the entropy of the probability distribution that relates an outgoing message of the mix (the one for which we want to know the anonymity set size) with all the possible inputs. As we will show later, the sender anonymity of all the outputs of a round is the same.

Recipient anonymity. If we want to compute the effective recipient anonymity set size of an incoming message that goes through the mix, we have to compute the entropy of the probability distribution that relates the chosen input with all possible outputs. Analogously to sender anonymity, the recipient anonymity of all the inputs of a round is the same.

4.4.1 Notation

The notation used for the computation of anonymity metrics for pool mixes is as follows:

- a_i**:** number of messages arrived to the mix in round i.
- n_i**:** number of messages contained in the pool of the mix at round i (just before flushing).
- m_i**:** number of real messages sent (forwarded) by the mix at round i.
- d_i**:** number of dummies which have been created by the mix and are sent (forwarded) by the mix at round i.
- s_i**:** number of total messages sent (forwarded) by the mix at round i ($s_i = m_i + d_i$).
- $P(n)$: characteristic function of the mix in the Generalized Mix Model.
- H_S**:** effective sender anonymity set size for an output message (sender anonymity).
- H_R**:** effective recipient anonymity set size of an input message (recipient anonymity).
- $p_a(i)$**:** probability of a target output message having arrived to the mix in round i.
- $p_s(i)$**:** probability of a target input message having been forwarded by the mix in round i.
- $I_{i,k}$**:** input k of round i.
- $O_{r,q}$**:** output q of round r.

- $\Pr(I_{i,k})$**:** probability of a target output message of matching input $I_{i,k}$.
- $\Pr(O_{r,q})$**:** probability of a target input message of matching output $O_{r,q}$.

4.5 Metrics for Deterministic Mixes

In this chapter, we represent pool mixes using the Generalized Mix Model introduced in the previous chapter. Let us recall that, in the model, pool mixes are represented by the *pool selection algorithm*; i.e., the function $P(n)$.

If a mix is deterministic then the number of messages sent is determined by the number of messages contained in the pool; the mix sends $s = nP(n)$ messages. The only randomness present in the flushing algorithm is the one used to select *which* messages will be sent, but not *how many*. Classical pool mixes fall into this category. Note that, for these mixes, once the number n of messages in the pool is known, the number s of messages sent is determined, and vice versa.

4.5.1 Recipient Anonymity

In order to compute the recipient anonymity of an input message, we have to find the probability distribution that relates that input with all the outputs that could match it.

The function $P(n)$ gives us the probability of a message leaving in the current round as a function of the number n of messages contained in the mix. The probability of an input message $I_{i,k}$ that arrived at round i leaving at round r is given by:

$$p_s(r) = P(n_r), \quad r = i \ ;$$

$$p_s(r) = P(n_r) \prod_{j=i}^{r-1} (1 - P(n_j)), \quad r > i \ .$$

That is, the probability of an input message $I_{i,k}$ being forwarded in the same round it has arrived, is given by the function $P(n)$. For later rounds, we compute it as the probability of the message staying in the pool the rounds $i..r-1$ and then leaving in round r.

In the absence of *a priori* or contextual information (as we have assumed in the description of the adversary model), the attacker cannot distinguish between messages belonging to the same input or output round. Therefore, the probability of each individual output $O_{r,q}$ matching the target input is the same for all outputs of round r:

$$\Pr(O_{r,q}) = \frac{P(n_r)}{s_r},\ \ r = i,\ \ q = 1..s_r\ \ ;$$
$$\Pr(O_{r,q}) = \frac{P(n_r)}{s_r}\prod_{j=i}^{r-1}(1 - P(n_j)),\ \ r > i,\ \ q = 1..s_r\ \ .$$

This result only makes sense if $s_r > 0$ (some message has been sent by the mix in round r). Otherwise, $\Pr(O_{r,q}) = 0$, and this term should not count in the computation of the entropy. The recipient anonymity of the target input is:

$$H_R = -\sum_{r=i}^{\infty} s_r \cdot \Pr(O_{r,q}) \log_2(\Pr(O_{r,q}))\ \ . \tag{4.1}$$

In theory, the adversary should wait infinitely in order to compute the recipient anonymity of a target input message; in practice, the adversary may get a good estimation just by taking into account a few rounds after the target input got to the mix.

4.5.2 Sender Anonymity

In order to compute the sender anonymity, we want to obtain the effective size of the anonymity set of senders for a message output by the mix in round r. Therefore, we compute the entropy of the probability distribution that relates a target outgoing message of the mix (the one for which we want to know the anonymity set size) with all the possible inputs.

Given that the mix treats all messages in the same way, the probability for an input to correspond to the target output depends on the round in which the input arrived to the mix. If the input arrived in the current round r, it is certain that it is in the pool, and the probability of matching the target output is uniformly distributed among the messages in the pool:

$$\Pr(I_{i,k}) = \frac{1}{n_r},\ \ i = r,\ \ k = 1..a_r\ \ .$$

For the messages that have arrived in previous rounds, we need to take into account that they might have already been sent by the mix. Therefore, we need to multiply the previous result by the probability of input $I_{i,k}$ still being inside the mix. Taking into account that the decisions of different rounds are independent, the probability of the target output of round r corresponding to input $I_{i,k}$ of round i is:

$$\Pr(I_{i,k}) = \frac{1}{n_r}\prod_{j=i}^{r-1}(1 - P(n_j)),\ \ i < r,\ \ k = 1..a_i\ \ .$$

Note that the result only makes sense if the number of inputs of the round we are considering is greater that zero ($a_i > 0$), otherwise the term should not be taken into account when computing the entropy. The measure of the effective sender anonymity set size of our target output is given by the entropy of the probability distribution:

$$H_S = -\sum_{i=0}^{r} a_i \cdot \Pr(I_{i,k}) \log_2(\Pr(I_{i,k})) \ . \tag{4.2}$$

Note that, in theory, the attacker needs to monitor the mix since it started operating in order to compute the anonymity; in practice, the probabilities associated to rounds which are far away in time become negligible and good approximations can be made taking into account a few rounds.

4.6 Metrics for Binomial Mixes

Binomial mixes [DS03b] differ from deterministic mixes in the fact that the number s_i of messages forwarded in round i follows a binomial distribution $B(n_i, P(n_i))$. In these mixes, an independent decision is taken for every message in the pool: a biased coin (being the bias the value of $P(n_i)$) is thrown for each message, so it is sent with probability $P(n_i)$.

In the case of binomial mixes, $P(n_i)$ does not determine the relationship between n_i and s_i. For a fixed incoming traffic pattern, deterministic mixes generate the same output traffic in repeated experiments. Binomial mixes, due to the added randomness, generate different output traffic patterns in each experiment. In order to measure the anonymity of a message going through a binomial mix, we must substitute $P(n_i)$ by the actual probability in the concrete realization of the mix, which is s_i/n_i.

4.6.1 Recipient Anonymity

For a given target input arriving to the mix in round i, the probability of being forwarded in round r is:

$$p_s(r) = \frac{s_r}{n_r}, \quad r = i \ ;$$

$$p_s(r) = \frac{s_r}{n_r} \prod_{j=i}^{r-1} (1 - \frac{s_j}{n_j}), \quad r > i \ .$$

Following the similar steps to the ones explained for the case of deterministic mixes, we obtain the probabilities linking the target input with the messages $O_{r,q}$ at the output of the mix:

$$\Pr(O_{r,q}) = \frac{1}{n_r}, \;\; r = i, \;\; q = 1..s_r \;\; ;$$

$$\Pr(O_{r,q}) = \frac{1}{n_r} \prod_{j=i}^{r-1} (1 - \frac{s_j}{n_j}), \;\; r > i, \;\; q = 1..s_r \;\; .$$

These probabilities are then used for the computation of the effective recipient anonymity set size H_R, as shown in equation 4.1.

4.6.2 Sender Anonymity

The sender anonymity for binomial mixes can be computed analogously by substituting $P(n_i)$ by the fraction of messages that have been taken from the pool and forwarded in round i, that is, s_i/n_i. The resulting distribution of probabilities linking the target output message to all possible inputs is:

$$\Pr(I_{i,k}) = \frac{1}{n_r}, \;\; i = r, \;\; k = 1..a_i \;\; ;$$

$$\Pr(I_{i,k}) = \frac{1}{n_r} \prod_{j=i}^{r-1} (1 - \frac{s_j}{n_j}), \;\; i < r, \;\; k = 1..a_i \;\; .$$

4.7 Metrics for Continuous Mixes

Continuous mixes were introduced in Chapter 3. In this section, we analyze how to apply anonymity metrics to this kind of mixes. In the original continuous mix proposal by Kesdogan [KEB98], also called *Stop-and-Go* mixes, the authors give an estimate of the anonymity offered by those mixes assuming that incoming traffic follows a fixed rate Poisson distribution.

In this section, we provide anonymity metrics for continuous mixes when traffic cannot be characterized as Poisson. We first provide a method for exponential delays at the mix, and then for uniformly distributed delays.

4.7.1 Exponential Delays

To formalize the behavior of the mixes, we define:

- X_s : an incoming message arriving at time s;
- Y_t : an outgoing message leaving at time t;
- D : the amount of time a message has been delayed.

We consider here that the messages are delayed according to an exponential distribution of the form $D \sim \exp(d)$. The Probability Density Function (PDF) and the Cumulative Distribution Function (CDF) of an Exponential distribution are:

$$\begin{aligned} \text{PDF} : f(d) &= e^{-d} \qquad \text{for all } d \geq 0 \ ; \\ &= 0 \qquad \text{elsewhere} \ ; \\ \text{CDF} : F(d) &= \Pr(D \leq d) = 1 - e^{-d} \qquad \text{for all } d \geq 0 \ ; \\ &= 0 \qquad \text{elsewhere} \ . \end{aligned}$$

All delay times are independent.

Crucial to note in this setup is that the sequence of outgoing messages is not a Poisson process. This would only be true if all inputs would arrive at the same time, hence belong to the mix when the delaying starts or if the sequence of arrivals are a Poisson process. But in our case, messages arrive at distinct moments in time, each being exponentially delayed upon their arrival times.

Mixes flush at fixed time moments which are observed by the attacker:

$$t \in \{\text{out}_1, \ \text{out}_2, \ \ldots, \ \text{out}_M\}.$$

He also observes the arrival times:

$$s \in \{\text{in}_1, \ \text{in}_2, \ \ldots, \ \text{in}_N\}.$$

If a message leaves the mix at time t, what are then the probabilities for the arrival times? Suppose the departure time $t = out$ is fixed. We then look for the probability that the message that left at time out is the same message as the one that entered the mix at time s:

$$\Pr(Y_{\text{out}} = X_s) = \Pr(D = \text{out} - s) \ .$$

We can hence rephrase the problem in terms of the delay: which values for the delay times are the most probable? Clearly, negative delay is impossible so only arrival times prior to out are probable. These arrival times form a set $\{\text{in}_1, \ \text{in}_2, \ \ldots, \ \text{in}_k\}$ with $\text{in}_k < out$. The matching delay times are then $\{$ out-in_1, out-in_2,..., out-in_k $\}$ to which we will refer to as $\{d_1, d_2, \ldots, d_k\}$. Note that $d_1 > d_2 > \ldots > d_k$. The density function of the delay times is known. Caution has to be taken however as the exponential function is a continuous function which means that the probability of the delay taking a single value is

zero: $\Pr(D = d_1) = \ldots = \Pr(D = d_k) = 0$.

How can we now calculate the probabilities of the delay times? To make this clear, let us look at Figure 4.1 and suppose that we only have three arrival times prior to *out*. We have thus three possible delays $d_1 > d_2 > d_3$. Let us now assume for simplicity reasons that $d_1 = 3$ hours, $d_2 = 2$ hours and $d_3 = 1$ hour. The variable delay is continuous and can theoretically take every value in the interval $[0, 3]$. However, we know that we only flush at three particular times and that hence only three particular delays can occur. We can exploit this knowledge in the following way:

$$\begin{aligned}
\Pr(D = d_1) &\approx \Pr(d_2 < D \leq d_1) = \text{ light area ;} \\
\Pr(D = d_2) &\approx \Pr(d_3 < D \leq d_2) = \text{ medium area ;} \\
\Pr(D = d_3) &\approx \Pr(0 < D \leq d_3) = \text{ dark area .}
\end{aligned}$$

In this way one can clearly see that the biggest area corresponds to the most probable delay. This is straightforward for more than three delays. For computation we make use of the cumulative distribution function (CDF) which is graphed in Figure 4.2. Cumulative probabilities are listed in tables and known in statistical software. For reasons of simplicity we put the mean of the exponential to be 1 hour (easy parametrization):

$$\begin{aligned}
\Pr(D = d_1) &\approx F(d_1) - F(d_2) = 0.9502 - 0.8647 = 0.0855 ; \\
\Pr(D = d_2) &\approx F(d_2) - F(d_1) = 0.8647 - 0.6321 = 0.2326 ; \\
\Pr(D = d_3) &\approx F(d_3) = 0.6321 .
\end{aligned}$$

In our little example, the message corresponds most likely with the one that entered the mix 1 hour before *out*. You can also clearly see this on Figure 4.1. In practical applications however, many possible delays will occur so that visual inspections will not be efficient and calculations have to made and compared.

4.7.2 Uniform Delays

Some practical implementations of continuous mixes allow for mix-chosen uniform delays. We have found a method to compute the anonymity provided by a mix that delays inputs uniformly from a distribution $U[a, b]$. The method consists in creating a table with all inputs and outputs. Then we search for all possible combinations input-output that are possible from an external observer's point of view (i.e., those that assign to every input that arrives at time T an output that

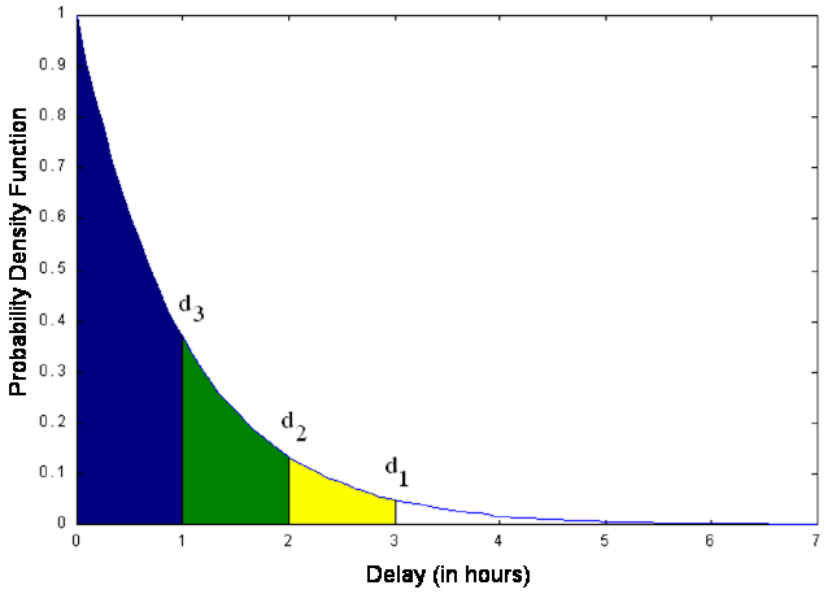


Figure 4.1: Example of an Exponential Probability Density Function

leaves between $T + a$ and $T + b$). Let us call the total number of combinations C.

Then, to compute the recipient (sender) anonymity of message m_i, we need to find the distribution of probabilities that link this input (output) to all outputs (inputs).

If input m_i appears matching output s_j in P combinations, then the probability assigned to s_j is P/C.

The probability of an input matching an output is computed as possible cases divided by total cases. From this distribution, the sender and recipient anonymity can be computed for every message.

4.8 Metrics with Dummy Traffic

As explained in Chapter 3, dummies are *fake* messages used to hide the traffic patterns inside a mix network. Although users can generate dummies (as proposed by Berthold and Langos in [BL02]), we focus here on dummies generated and discarded by mixes, and their impact on the anonymity metrics computation.

We assume that the mix generates dummies following a probability distribu-

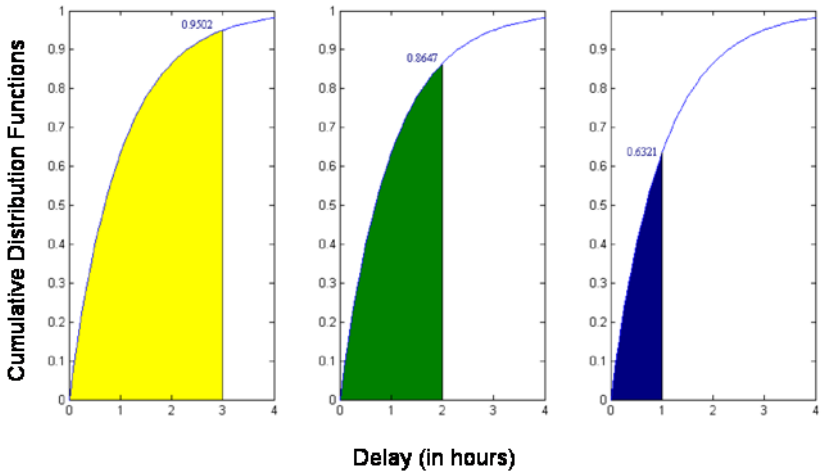


Figure 4.2: Matching Exponential Cumulative Distribution Function

tion which is independent from the traffic flow (in order to prevent an attacker from exploiting to his advantage potential dependencies). We also assume that the number of dummies inserted in a round is independent of the number of dummies inserted in previous rounds, in order to keep the mix design *stateless*, that is, that the decisions of one round are not constrained by the events of previous rounds. A setting in which the mix keeps, for instance, a constant number of dummies in the pool would need a different analysis. We consider two possible scenarios for the generation of dummies.

Insertion at the output. We assume that the mix inserts the dummy messages into the output link at the time of flushing (i.e., adds them to the batch of real messages withdrawn from the pool). If the mix flushes after a timeout (timed mix), it could add dummies even if no real messages are sent. In this scenario, the pool contains only real messages (note that dummies at intermediate hops are indistinguishable from real messages, and thus considered real messages).

Insertion in the pool. The dummies are added to the pool and mixed there with the real messages being routed by the mix. In this case, the number of dummies present at the output of a particular round depends on the random selection of messages from the pool. The function $P(n)$ must take into account the dummies present in the pool (i.e., n is the sum of the real messages and the dummies inserted by the mix itself). Otherwise, in the case of low traffic, the

mix would accumulate dummies that are flushed at a very low rate.

In addition to the notation presented previously, we denote:

- d_k**:** number of dummies inserted in round k;
- p_d**:** probability of a target message being a dummy.

4.8.1 Sender Anonymity with Dummies Inserted at the Output

In this section we point that the intuitive way of including dummies in the anonymity metrics leads to a misleading result that does not reflect the actual increase in the anonymity of the users.

In order to compute the sender anonymity provided by a pool mix when dummy traffic is being inserted at the output link, we would first choose a target output, and then compute the probability of this message being either one of the inputs or a dummy. There is a conceptual difference between these two cases: if the output is a real message, we want to know *which one*; if it is a dummy, we do not really care whether it is "dummy number 1" or "dummy number 7": the fact of the message being a dummy contains only one bit of information (dummy/no dummy). We show that treating the two cases analogously would lead to a metric that is not meaningful in terms of anonymity.

Let us consider a distribution of probabilities $\Pr(I_{i,k})$ that relates the chosen target output with every possible input $I_{i,k}$ when no dummies are generated by the mix. The entropy of this distribution is H_S. If the mix adds d_k messages to every output round, then:

- Probability of target output being a dummy: $p_d = d_k/s_k$.
- Probability of target output being input $I_{i,k}$: $(1 - p_d) \cdot \Pr(I_{i,k})$.
- H_{DS}**:** Effective sender anonymity set size for the output message (sender anonymity) when dummies are generated by the mix.
- H_{DR}**:** Effective recipient anonymity set size of an input message (recipient anonymity) when incoming dummies are discarded by the mix.

The entropy of the new distribution is:

$$H_{DS} = -p_d \log_2(p_d) - \sum_i (1 - p_d) \cdot \Pr(I_{i,k}) \log_2((1 - p_d) \cdot \Pr(I_{i,k})) \ ;$$

$$H_{DS} = -p_d \log_2(p_d) - (1 - p_d) \log_2(1 - p_d) + (1 - p_d) \cdot H_S \ .$$

From the formula, we observe that for high values of H_S and p_d, the value of the new entropy H_{DS} (with dummies) may be lower than H_S (entropy with no dummies).

The decrease in the entropy is consistent with the concept associated with it: *uncertainty*. If $p_d >> 1 - p_d$, the attacker has little uncertainty about the output, he may guess that it is a dummy and he will be right with probability p_d. Nevertheless, the attacker is not gaining much with this guess because the uncertainty about the inputs that correspond to real outputs stays the same.

We should conclude that it is not straightforward to use the metric H_{DS} to compute the sender anonymity of a mix that generates dummy traffic. In order to get meaningful results, we should assume that the attacker chooses a real output message, and never a dummy. As complementary information about the chances of the attacker of choosing a real message at the output of a mix, we suggest to provide, together with the metric H_S, the probability of success choosing a real message, $1 - p_d$.

On the other hand, we should note that the incoming dummies that are discarded by the mix do contribute to the sender anonymity, and they must be taken into account just as if they were real messages (they have the effect of an increased traffic load going through the mix).

4.8.2 Sender Anonymity with Dummies Inserted in the Pool

The same problem pointed out in the previous section about the relevance of the metric applies to this scenario, hence the same solution is suggested. We propose as metric the entropy conditioned to the event that a real message is chosen, together with the probability of choosing a real message $1 - p_d$, as in the previous case. The average number of dummies contained at round r in the pool of a mix implementing this policy is:

$$D_r = d_r + \sum_{i=1}^{r-1} d_i \prod_{j=i}^{r-1} (1 - \frac{s_j}{n_j}) \ .$$

Note that for deterministic mixes s_j/n_j can be substituted by $P(n_j)$. The proportion of dummies at the output is, on average, the same as in the pool (the dummies are selected to be sent with the same probability as real messages). The probability of selecting a real message at the output is: $1 - p_d = 1 - D_r/n_r$.

Note that the entropy in this scenario must be computed taking into account that n includes the dummies present in the pool. The value of n is thus higher than in the case in which dummies are inserted at the output. Note that the value of the function $P(n)$ depends now not only on the number of real messages

contained in the pool, but also on the number of dummies. This implies that n_k will be bigger than in the case in which no dummies are generated. $P(n)$ is a function that either grows with n or stays constant (a function that decreases with n would not make sense: the mix would send less messages as the traffic increases). From the expression of the entropy, we can conclude that for the same traffic load, the anonymity and the delay decrease when this policy is used instead of inserting the dummies at the output (note that higher values of $P(n)$ provide less anonymity and less delay). Eventually, we could reach a situation in which a real message is only mixed with dummies. Note that if the function $P(n)$ does not increase its value ($P(n)$ may reach a maximum value), the anonymity would not be affected.

4.8.3 Recipient Anonymity with Dummies Inserted at the Output

In this section we discuss the impact of the dummy traffic created by the mix on the recipient anonymity. We show that a simple extension of the metric allows us to take into account dummy traffic generated by this mix (the input dummy traffic discarded by the mix cannot be considered for the same reasons as the dummies generated cannot be taken into account in the computation of sender anonymity). The number of dummies inserted at round k is d_k. The number of dummies inserted follows a distribution $\Pr(d_k = d)$. We make abstraction of this distribution.

A similar problem arises for the case of recipient anonymity as for sender anonymity. In this case, we must assume that the attacker chooses to trace a real input. This is a reasonable assumption when the message comes from the user. But in certain circumstances, the attacker may want to trace a message that comes from another mix (trying to find the path of the target message in the network). In this case, the attacker may choose a message that is actually a dummy that will be discarded by the mix. It does not seem easy to model the dummy traffic that arrives to a mix for being discarded, given that it depends on the topology of the network and the path length of the dummy.

In order to effectively apply the anonymity metric, we must assume that the attacker computes the recipient anonymity for a message that will not be discarded by the mix (that is, a message that matches an output). Analogously to the case of sender anonymity, we may provide as complementary information to the recipient anonymity, the probability of choosing an input message that is not discarded by the mix.

The mix inserts d_k messages at the output link in round k. The recipient anonymity when dummy traffic is being inserted at the output of the mix is computed using (4.1). The only difference in this case is that s_k has a component

of real messages, m_k, and another one of dummy messages, d_k ($s_k = m_k + d_k$). Therefore, the impact of the dummy traffic is equivalent to an increase in the traffic load.

This simple result is consistent with the fact that real messages which are not the one we want to trace act as cover traffic for the target message, just as dummy messages do. Whenever there is at least one real message in the output of a round, the probabilities of matching our target input message are distributed over the set of messages sent out by the mix in that round.

If the pool mix is deterministic, the adversary knows the number m_k of real messages forwarded by the mix, as it is deterministically derived from the contents of the mix. The same occurs when the dummy traffic policy is deterministic (i.e., a constant number of dummies is generated each round). In these cases, the rounds in which $m_k = 0$ (only dummy messages sent) can be identified and discarded by the attacker. These dummy messages do not increase the recipient anonymity provided by the mix. This is not the case when the attacker has uncertainty about d_k and m_k (binomial mix with random dummy policy). In this case, he has to take into account dummies sent in rounds in which no real message is flushed, as we show in Sect 5.6.2.

We can conclude that binomial mixes with random dummy policy offer more anonymity when the traffic is low (in particular, when $m_k = 0$), because the uncertainty of the attacker about the existence of real messages in the output increases the recipient anonymity: messages of rounds that would be discarded by the attacker when attacking a deterministic mix cannot be discarded if the mix is binomial.

4.8.4 Recipient Anonymity with Dummies Inserted in the Pool

The mix inserts in the pool d_k dummies in round k. The recipient anonymity provided by a mix implementing this dummy policy is computed using equation (4.1). The difference in this case is that the value of the function $P(n)$ depends not only on the number of real messages contained in the pool, but also on the number of dummies, with the same consequences on the anonymity as mentioned in Sect. 4.8.2.

Note that, as in the previous case, we must assume that the target input selected by the adversary for tracing must be a message which is not discarded by the mix (i.e., a dummy which reaches the final destination and is thus discarded).

4.8.5 Remarks on Binomial Mixes

When the mix does not generate dummy traffic, the attacker has all the information needed to compute the anonymity (a_k, s_k and $P(n_k)$), because he can determine the number of messages in the pool, n_k. When the mix generates dummies, we can find some differences between deterministic and binomial mixes, which are explained in detail in the next chapter. If the mix is deterministic, then the attacker can find out n_k, regardless of the dummy policy. If the mix is binomial, then for a deterministic dummy policy he will also be able to determine n_k. But for a random dummy policy the value n_k cannot be determined, and therefore $P(n_k)$ remains unknown. This means that the attacker cannot compute with certainty the anonymity of the messages. He may be able to estimate it; the estimation is more accurate when the number of dummies or the randomness of the dummy distribution decreases. It is important to note that this uncertainty is independent of the actual value of the anonymity provided by the mix.

4.9 Conclusions

In this chapter, we have computed the sender and recipient anonymity provided by generalized mixes and continuous mixes. The formulas provided are compact and easy to evaluate and implement. We have indicated how to measure the sender and recipient anonymity when pool mixes insert dummy traffic in the pool or at the output. Given that the intuitive extension of the metric for this scenario provides confusing results, we have clearly explained how it should be applied.

From the presented results, we can conclude that dummies generated by the mix contribute to recipient anonymity, but not to sender anonymity. Analogously, dummies discarded by the mix contribute to sender anonymity but not to recipient anonymity. Much attention must be paid when implementing this metric to nodes that generate or discard dummy traffic. Inserting the dummies in the pool provides less anonymity and less latency that inserting them at the output.

Some of the topics that are subject of future work are:

- Find a metric that expresses the sender and recipient anonymity provided by a mix network with dummy traffic.
- Compare the anonymity achieved with different distributions of dummy traffic. Obtain quantitative results.
- Compare the anonymity provided by pool mixes to the anonymity provided by *Stop-and-Go* mixes when dummy traffic is generated.

- Find how to measure anonymity if the dummy generation is dependent on the traffic flow. Do these dependencies make the system stronger or more vulnerable to passive attacks?

Chapter 5

Anonymity Metrics for Mixes: Active Attacks

I have nothing but confidence in you, and very little of that.
– Groucho Marx

5.1 Introduction

This chapter analyzes the deployment of an active attack (in particular, the *blending* or $n-1$ attack) on deterministic pool mixes (timed and threshold), binomial mixes, continuous mixes, and pool mixes that generate dummy traffic. We define a set of parameters to measure the effort of the attacker, and we study how those parameters can be measured for the considered mix configurations. We analyze the remaining anonymity of the message under attack, and show how the use of dummies impacts this parameter.

The research questions addressed in this chapter are: How can we characterize the effort of the $n-1$ attacker? What is the effort to deploy the attack on different mixes? Are some mixes more resistant than others? Does the adversary succeed with 100% probability in all cases? What is the impact of using dummy messages? How can we use dummy traffic in an optimal way to reduce as much as possible the success of $n-1$ attacks?

The results presented here build on those of Chapter 2 and Chapter 3. They have been extended from our original work *Reasoning about the anonymity of pool mixes that generate dummy traffic*, published at the *6th Information Hiding Workshop 2004* [DP04b] and *Generalizing mixes*, published at the *Workshop on Privacy Enhancing Technologies 2003* [DS03b].

In this chapter, we first describe the *blending* or $n-1$ *attack* in Sect. 5.2. In Sect. 5.3 we study this attack when it is deployed against timed and threshold deterministic pool mixes. Section 5.4 studies the effort required to deploy such an attack on binomial mixes and Sect. 5.5 on continuous mixes. The results of the attack when mixes generate dummy traffic are analyzed in Sec. 5.6. Finally, Sect. 5.7 presents the conclusions of this chapter.

5.2 The Blending or $n-1$ Attack

The $n-1$ or *blending* attack was introduced by Gülcü and Tsudik in [GT96], and later analyzed in detail for several deterministic mixes by Serjantov *et al.* in [SDS02, Ser04]. This attack is a method to trace a message going through a mix. Other active attacks that can be found in the literature are tagging attacks, described by Pfitzmann in [PP90, Pfi94]. Tagging attacks consist in altering the content of the message (e.g., by flipping one bit in the encrypted payload) and then analyzing the reaction of the anonymity system.

The goal of the $n-1$ attack is to identify the recipient of a message (the attack only affects recipient anonymity, not sender anonymity). We assume that the attacker can recognize his own messages at the output of a mix (e.g., by selecting easily recognizable destinations as next communication hop after the mix under attack). In this attack model, the adversary is able to delay messages from other users and to generate large numbers of messages from distributed sources (so that the flooding of the mix cannot be distinguished from a high traffic load).

The attacker we are considering in this chapter controls all communication lines (*global attacker*). He cannot only observe all incoming and outgoing messages, but also delay the messages of the users and insert messages (*active attacker*). The attacker does not have access to the contents of the mix, i.e., the mix is a black box for the attacker (*external attacker*). As the resources controlled by the attacker are fixed, we consider he is a *static* adversary.

The goal of the attacker is to trace a particular message (the target message) that is sent by a user to the mix. The actions of the attacker can be divided into two phases: the *emptying* phase and the *flushing* phase.

Emptying phase. As soon as the adversary selects a target message to trace, it starts delaying it and all the other inputs getting to the mix to which the target message is addressed. At the same time, the attacker generates a large number of messages which are sent to the mix under attack. The goal of the attacker in this phase is to *clean* the unknown messages sitting in the internal memory of the mix, and to fill it with his own messages (which he can recognize at the output). A graphical explanation of the emptying phase of the attack is presented

in Fig. 5.1. The left side of the figure shows the beginning of the emptying phase. The mix contains a certain number of unknown messages the adversary needs to expel from the mix. This phase finishes as the mix forwards the last unknown message. At this point, the mix only contains messages generated by the attacker.

Flushing phase. Once the mix only contains attacker messages, the adversary lets the target message into the mix, as shown in Fig. 5.2. In order to shorten the duration of the attack, the adversary keeps on sending attacker messages to the mix, until the mix flushes the target message. If the attacker is certain on his detection of the target message at the output, he has successfully broken the anonymity provided by the mix. Note that the adversary needs to block all unknown messages addressed to the mix from the moment the user sends the target message until it appears at the output of the mix.

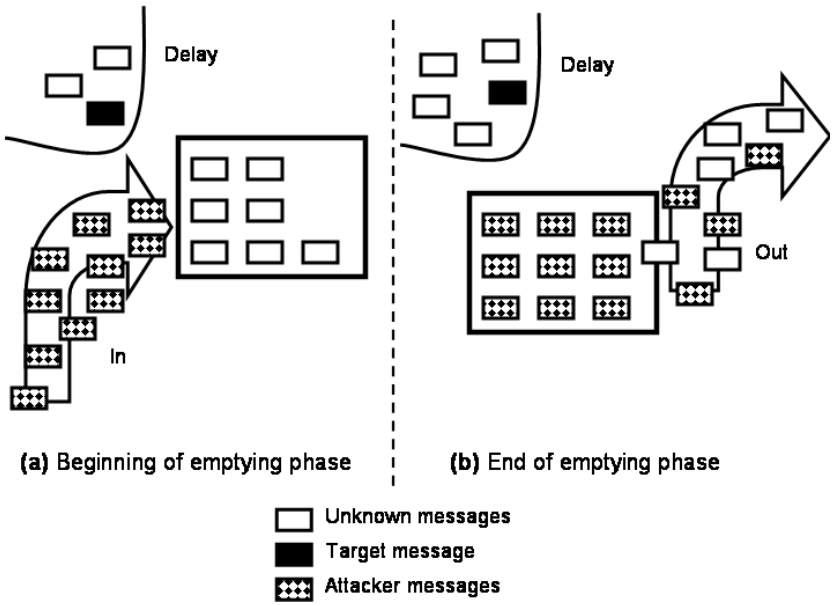


Figure 5.1: Emptying Phase of the $n-1$ Attack

We evaluate the effort of the attacker to deploy a *blending* taking into account the following parameters:

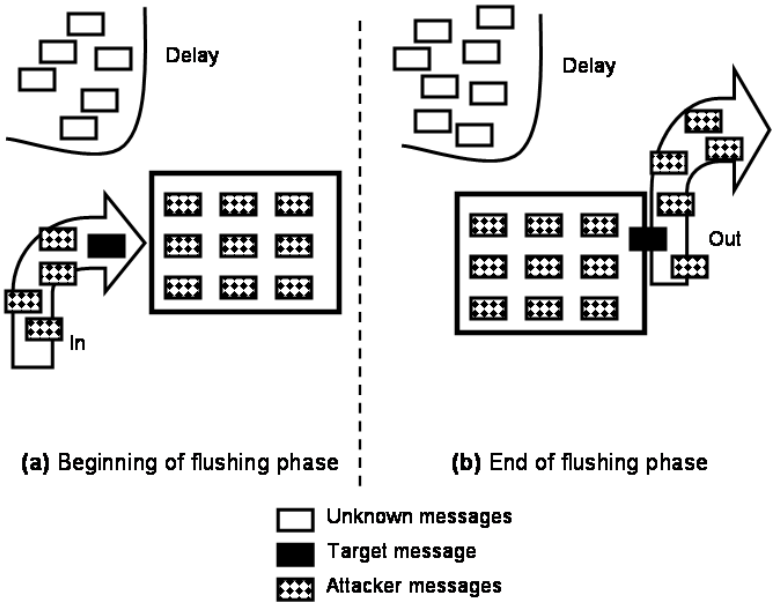


Figure 5.2: Flushing Phase of the $n-1$ Attack

- T_p**:** Observation time needed to establish the initial state of the mix.
- R_p**:** Average number of rounds needed to establish the initial state of the mix.
- T_a**:** Time required for the active phase of the attack.
- R_a**:** Average number of rounds of active attack.
- N_u**:** Average number of unknown messages the adversary needs to delay.
- N_a**:** Total amount of messages generated on average by the adversary to deploy the attack.
- $\Pr(O_{i,j})$**:** Probability of output message $O_{i,j}$ being the target message.
- H**:** Entropy of the probability distribution linking the target input with all possible outputs (remaining anonymity after the attack). This entropy defines the degree of success of the attacker. If $H = 0$, the attacker has succeeded with 100% confidence in tracing the target message.

We assume that the adversary has been monitoring the mix for enough time T_p prior to the detection of the target message to accurately estimate the initial state of the mix. The attacker is capable of generating m_a messages per second. During the attack, m_u unknown messages on average are sent to the mix per second.

5.3 The Blending Attack on Deterministic Pool Mixes

As introduced in Chapter 3, deterministic pool mixes can be characterized in the Generalized Mix Model by the function $P(n)$, where n is the number of messages stored in the pool. When these mixes forward a set of messages, they send $s = n \cdot P(n)$ messages.

5.3.1 Analysis of Deterministic Timed Pool Mixes

Timed pool mixes forward messages every T seconds, where T is the *timeout* of the mix. We denote by $P(n)$ the characteristic function of the mix in the Generalized Mix Model (GMM). We assume $P(n)$ reaches a maximum value P_M for $n = N_M$, as it is the case for most pool mixes, such as Mixmaster [UMS03]. We recall that $P(n)$ determines the fraction of messages withdrawn from the pool and forwarded.

Due to the deterministic relationship between the number of messages sent and the number of them contained in the pool, when the adversary observes s messages at the output, he knows that $s = nP(n)$. As the adversary knows the function $P(n)$, he can compute the number n of messages that were present in the pool before the flushing. Trivially, the number of messages left in the pool after the flushing is $n - s$. Therefore, the adversary only needs to observe one flushing round in order to guess the number of unknown messages the mix keeps in its pool; i.e., $R_p = 1$.

We assume that, at the time of detection of the target, the mix contains U unknown messages. As we have indicated, the adversary only needs one round of observation in order to determine with 100% certainty the number of unknown messages contained in the pool of the mix (in our case, this number is U). For a timed pool mix of timeout T seconds, the adversary must wait, on average, $T_p = \frac{T}{2}$ seconds to determine the amount of messages contained in the pool.

Emptying phase. When the adversary first detects the target message being sent to the mix at time T_0, it blocks it and starts delaying any other unknown input getting to the mix. Assuming that there are U initial unknown messages

in the pool, the adversary sends $N_M - U$ attacker messages to the mix. In the following rounds of attack, the adversary refills the mix with $P_M \cdot N_M$ attacker messages (the same number of messages sent per round), until he detects that the last unknown message has left the mix.

The probability of an unknown message being forwarded at round i is $\Pr(i) = P_M(1-P_M)^{i-1}$. This probability follows a geometric distribution of the form $X \rightarrow Ge(1-P_M)$. The average number of rounds needed to expel one unknown message from the pool is $\frac{1}{P_M}$. Assuming that there are initially U messages in the pool, we know from [SDS02] that the probability of the mix being empty at round R_e of active attack is given by:

$$\Pr(\mathit{empty} \leq R_e) = (1-(1-P_M)^{R_e})^U \ .$$

The probability of the last unknown message leaving the mix at exactly round R_e is:

$$\Pr(\mathit{empty} = R_e) = (1-(1-P_M)^{R_e})^U - (1-(1-P_M)^{R_e-1})^U \ .$$

The expected number of rounds needed to empty the mix on average can be computed as:

$$R_a(\mathit{empty}) = \sum_{i=1}^{\infty} i \cdot ((1-(1-P_M)^i)^U - (1-(1-P_M)^{i-1})^U) \ .$$

Which, after some mathematical transformations, can also be expressed as:

$$R_a(\mathit{empty}) = -\sum_{k=1}^{U} \frac{(-1)^k \cdot U!}{k! \cdot (U-k)! \cdot (1-(1-P_M)^k)} \ .$$

Flushing rounds. Once the pool of the mix is clean from unknown messages, the attacker may let the target message into the mix. He keeps on sending $P_M \cdot N_M$ messages per round. Taking into account that the probability of the target being flushed is P_M at every round, the average number of rounds required to flush the message through the mix is:

$$R_a(\mathit{flush}) = \frac{1}{P_M} \ .$$

The total number of rounds needed to deploy the active attack is $R_a = R_a(\mathit{empty}) + R_a(\mathit{flush})$. Given that the timeout of the mix is set to T seconds, the average amount of time needed by the attacker is:

$$T_a = R_a \cdot T \ \ \mathit{seconds}.$$

Assuming that the mix would receive during the attack m_u messages per second, the total amount of messages the adversary needs to delay (or remove) from the links is $N_u = m_u \cdot T_a$. The average number of messages generated by the attacker is $N_M - U$ for the first round and $N_M \cdot P_M$ for subsequent rounds. The total number of messages the adversary generates on average is:

$$N_a = N_M - U + N_M \cdot P_M \cdot (R_a - 1) - 1 \ .$$

In this scenario, the adversary identifies the target message at the output with probability $\Pr(O_{i,j}) = 1$. The attacker has total certainty on his identification of the output. Consequently, the remaining anonymity of the message H is zero.

5.3.2 Analysis of Deterministic Threshold Pool Mixes

Threshold mixes forward messages once they have accumulated a certain amount of them in the internal memory. As shown in Chapter 3, threshold pool mixes are represented in the GMM as a point with coordinates (N_M,P_M). This indicates that the mix forwards $P_M \cdot N_M$ messages every time it receives the N_M-th message.

These mixes have the advantage over timed mixes of providing better response times in high traffic load conditions. However, this feature can be exploited by the adversary in order to reduce significantly the duration of the attack.

In this case, the adversary does not need to determine the number U of messages in the pool in advance. He may just start sending his messages, and the mix will flush once the threshold N_M has been reached, indicating the number of messages that were present in the pool (which is N_M minus the number of messages sent by the attacker until the flushing). Therefore, in this case $R_p = 0$ and $T_p = 0$.

We assume that m_u unknown messages per second are addressed to the mix, and that the attacker can generate messages at the rate of m_a messages per second. The attack develops as follows:

Emptying phase. The attack begins as the adversary detects the target being sent to the mix. From that moment until the end of the attack, it will delay all unknown messages getting to the mix. Assuming that there are U messages in the mix when the attack starts, the mix will flush once the attacker has sent $N_M - U$ messages.

As in the previous case, the adversary counts the number of unknown messages at the output of each flush, and checks if all U unknown messages have left the mix. If there are remaining unknown messages, the adversary sends $P_M \cdot N_M$ messages to the mix in a new emptying round. As in the case of timed mixes, the average number of rounds needed to empty of the mix is:

$$R_a(empty) = -\sum_{k=1}^{U} \frac{(-1)^k \cdot U!}{k! \cdot (U-k)! \cdot (1-(1-P_M)^k)} \ .$$

Flushing rounds. Once the mix is empty of unknown messages, the adversary lets the target into the mix. It refills it with $P_M \cdot N_M - 1$ messages and observes if the target appears at the output. The adversary repeats this process until the target is sent by the mix. On average, the number of rounds needed to flush the target is:

$$R_a(flush) = \frac{1}{P_M} \ .$$

As we can see, the total number R_a of rounds needed to deploy the active attack is similar to the timed pool mix. On the contrary, the total amount of time needed to complete the attack can be greatly reduced for an attacker capable of generating messages at a high rate m_a. The first round of attack, the adversary needs to send $N_M - U$ messages to the mix, and in subsequent rounds he sends $P_M \cdot N_M$ messages (on average), just as in the case of the timed mix. The total amount of time needed by the attacker to trace a message is:

$$T_a = \frac{N_M - U}{m_a} + (R_a - 1)\frac{P_M \cdot N_M}{m_a} \ .$$

Note that, if the attacker is able to generate messages at a high rate, he may substantially reduce the duration of the attack compared to the case of timed mixes. As we show below, this reduction in the time needed to deploy the attack has an impact on the number N_u of unknown messages the adversary has to delay, which is reduced. Assuming that the mix would receive during the attack m_u messages per second, the total amount of messages the adversary need to delay (or remove) from the links is $N_u = m_u \cdot T_a$:

$$N_u = ((N_M - U) + (R_a - 1) \cdot P_M \cdot N_M)\frac{m_u}{m_a} \ .$$

As we can see, the reduction in time of attack provided by threshold mixes may greatly reduce the effort of the attacker spent on delaying unknown messages. The total number of messages the adversary generates on average is, as in the case of timed mixes:

$$N_a = N_M - U + (R_a - 1) \cdot P_M \cdot N_M - 1 \ .$$

As in the previous case, the adversary identifies the target message at the output with probability $\Pr(O_{i,j}) = 1$. Consequently, the remaining anonymity of the message, H is zero.

5.4 Binomial Pool Mixes

In this section we study the deployment of a blending attack on binomial pool mixes. We show that the probabilistic relationship established by the binomial mix between the number n of messages in the pool and the number s of messages sent forces the adversary to monitor the mix much longer in advance than in the case of deterministic mixes. Alternatively, the attacker may choose a flooding strategy, that involves generating more attacker messages.

We first present the effort to guess the number of unknown messages initially contained in the pool of the binomial mix. Then, we present a faster strategy for the adversary and compute the effort required to deploy it. The binomial mix considered here has the same timeout T and characteristic function $P(n)$ than the timed mix of Sect. 5.3.

5.4.1 Guessing the Number of Unknown Messages in the Mix

In this section we estimate how long in advance the adversary must monitor the mix in order to know the amount of messages contained in the pool. At the output of the binomial mix in round i, the number s_i of forwarded messages follows a binomial distribution with respect to the number n_i of messages in the pool. The probability of sending s_i messages, given that the pool contains n_i messages is [Fel50]:

$$\Pr(s_i|n_i) = \frac{n_i!}{s_i!(n_i - s_i)!} \cdot P(n_i)^{s_i} \cdot (1 - P(n_i))^{n_i - s_i} \quad .$$

If the output of a batch containing s_i messages is observed, we can apply Bayes' Rule to reverse the formula and compute the probability of each value of n_i, given the observation of s_i:

$$\Pr(n_i|s_i) = \frac{\Pr(s_i|n_i)}{\sum_{j=s_i}^{N_{\max}} \Pr(j|n_i)} \quad .$$

This way we obtain a probability distribution for the value of n_i. Note that we cannot determine the number n_i of messages contained in the mix by one observation of the number s_i of messages present at the output. N_{max} represents the maximum capacity of the mix; once it has been reached, new messages will be dropped.

We explain now how to compute the number of rounds of observation needed to estimate the number of initial unknown messages. The algorithms presented have been implemented in a Java simulator.

Given that every round is independent from the others, we can multiply the results of every round, taking care of shifting the terms we are multiplying as many positions as the difference between the number n_0 of unknown messages in the first round of attack, and the number n_r of messages at observation round r. This difference is known to the attacker because he can count the incoming and outgoing messages.

The notation used in this section is:

- n_j is the number of messages contained in the mix at the j-th round of attack (being n_0 -the number of messages contained in the mix when the attack starts- the number the attacker is trying to guess).
- s_j is the number of messages sent by the mix in the j-th round of attack. This number is a function of n_j.
- a_j is the number of messages that arrive to the mix during the j-th round. We take into account a_j starting from $j = 1$.
- *shift* is the difference between n_j and n_0 ($shift = n_j - n_0$). The attacker knows this number because he observes the number of incoming and outgoing messages at each round; e.g., at round 1 $shift = n_1 - n_0 = a_1 - s_0$. This number can be either positive or negative.
- P is an array that contains the result of the algorithm in the present round, taking into account all the previous rounds. The array has $N_{max} + 1$ elements. $P[i]$ contains the probability of $n_0 = i$.
- A is an array that contains the probabilities of the values of n for this round. The array has $N_{max} + 1$ elements. $A[i]$ contains the probability of $n_j = i$, where j is the number of the round.

The algorithm starts at round 1, and at the j-th round is as follows:

shift > 0**.** In this case we know that $n_j > n_0$. In order to be able to multiply the result of this round to the previous ones (which have the maximum value close to n_0), we have to shift the values of A *shift* positions to the left. This way, the estimation of n_j can be used to improve our knowledge of n_0 ($n_0 = n_j -$ *shift*).

The values we lose at the left of the array are not important, because this corresponds to impossible values of n_j: given that $n_0 \geq 0$, this implies that $n_j \geq$ *shift*. On the other hand, at the right side of the array, we have to introduce numbers. The solution is to propagate the value of N_{max}. This makes sense because in case $n_0 \geq N_{max} -$ *shift* then $n_j = N_{max}$, given that once the capacity of the mix (N_{max}) has been exceeded messages are dropped.

After shifting the values of the A array, we have to rescale them in order to have a distribution of probabilities (the sum of all values must be 1). The code in Java is as follows:

```
if (shift > 0) {
    for (int i=0; i<=N_MAX-shift; i++)
        A[i] = A[i+shift];
    for (int i=N_MAX+1-shift; i<=N_MAX; i++)
        A[i] = A[N_MAX];
    // rescaling A
    double sum = 0.0;
    for (int i=0; i<=N_MAX; i++) sum = sum + A[i];
    for (int i=0; i<=N_MAX; i++) A[i] = A[i]/sum;
}
```

shift < 0**.** This is the case in which in the present round $n_j < n_0$. We have to shift the values of the A array to the right by *shift* positions. We lose the last *shift* values, which are, again, impossible values of n_j, because $n_0 \leq N_{max}$ implies $n_j \leq N_{max} - |shift|$. At the left of the array we have to introduce values from the positions 0 to $|shift| - 1$. In this case the value we introduce is 0: we know that $n_j \geq 0$, therefore $n_0 \geq |shift|$ (note that $n_0 = n_j + |shift|$). This implies that any value of n_0 smaller than $|shift|$ is impossible.

Again, as in the previous case, we must rescale the values of A in order to obtain the new distribution.

The code in Java is as follows:

```
if (shift < 0) {
    for (int i=N_MAX; i>=-shift; i--)
        A[i] = A[i+shift];
    for (int i=0; i<-shift; i++)
        A[i] = 0.0;
    // rescaling A
    double sum = 0.0;
    for (int i=0; i<=N_MAX; i++) sum = sum + A[i];
    for (int i=0; i<=N_MAX; i++) A[i] = A[i]/sum;
}
```

shift $= 0$ In this case $n_0 = n_j$, and we can multiply both arrays (P and A) without changing A.

Multiply P and A. After shifting and rescaling the elements of the array A, we can multiply both arrays element by element. After this multiplication we have to rescale the result and we obtain the distribution of probabilities of the value of n_0 including the j-th round.

The code in Java is:

```
// multiply probabilities
double sum = 0.0;
for (int i=0; i<=N_MAX; i++) {
    P[i] = P[i]*A[i];
    sum = sum + P[i];
}
// rescaling P
for (int i=0; i<=N_MAX; i++) P[i] = P[i]/sum;
```

At this point, the array P contains the current distribution of probabilities, being $P[i]$ the probability of $n_0 = i$, and taking into account the information obtained during all the rounds of attack.

We have implemented these algorithms in a simulator. The function $P(n)$ selected for the test was the one implemented in the working versions of the Mixmaster remailer [UMS03]. According to the results of our simulations, the adversary needs about 200 rounds of observation before he can estimate the amount of initial unknown messages in the mix with 95% confidence. Taking into account that the timeout of Mixmaster is set by default to 15 minutes, the attacker would need to monitor the mix more than two days in advance (before the target is detected) in order to carry out the attack with a good estimate of the number U of unknown messages present in the pool.

If U can be confidently determined when the target message is sent to the binomial mix, the effort required by the adversary to deploy the active part of the attack is, on average, the same as in the case of deterministic timed mixes. Regarding the certainty on the result of the attack, it depends on the certainty the adversary has on the fact that the mix is empty when he lets the target into it. This uncertainty becomes arbitrarily low when the adversary has been monitoring the mix for a long time. Therefore, we can say that in this case the remaining anonymity can be made arbitrarily small.

5.4.2 Flooding Strategy

If the adversary has not been monitoring the binomial mix several days before the target was sent to it, he may use a different attack strategy which can be completed in much less time, while it requires more active resources than the previous case.

Binomial mixes have the advantage over deterministic mixes of not revealing the number of messages they keep in the pool just by a few observations of its outputs. In order to make a fair comparison, the binomial mix considered here has the same $P(n)$ function (i.e., same N_M and P_M) and timeout T than the deterministic pool mix considered in Sect. 5.3.

As in the previous cases, we assume that m_u unknown messages per second are addressed to the mix, and that the attacker can generate messages at the rate of m_a messages per second.

The emptying phase. During this stage of the attack, the goal of the attacker is to remove all unknown messages contained in the pool, while preventing new unknown messages from going into the mix. In order to force the mix to send out as many unknown messages as possible in each round, the attacker makes sure that the mix contains at least N_M messages, where N_M is the minimum number of messages that guarantees that the $P(n)$ function takes its maximum value, P_M. If the attacker wants to empty the mix with probability $1 - \epsilon$, then he will have to flood the mix for $R_a(\mathit{empty})$ rounds.

The formula that can be used to estimate the number $R_a(\mathit{empty})$ of rounds needed to flush all unknown messages with probability $1 - \epsilon$ is:

$$(1 - (1 - P_M)^{R_a(\mathit{empty})})^U \geq 1 - \epsilon \ . \tag{5.1}$$

Where U is the number of initial unknown messages contained in the pool. Note that the adversary has to estimate U from the observation of the mix flushes. Initially, he sends N_M messages. In later rounds, the attacker refills the mix with $N_M \cdot P_M$ messages to make sure that the mix contains at least N_M messages in the pool.

Note that the attacker succeeds in emptying the pool with probability $1 - \epsilon$. With probability ϵ an unknown message (or more) remains in the pool. If this is the case, the adversary may trace the wrong message and not notice it.

The flushing phase. Once the mix has been emptied of unknown messages, the attacker sends the target message to the mix. Now, he has to keep on delaying other incoming unknown messages and also send messages to make the mix flush the target.

The average number of rounds needed to flush the message is, as in the previous cases:

$$R_a(\mathit{flush}) = \frac{1}{P_M} \ .$$

And the total number R_a of rounds of active attack is the sum of $R_a(\mathit{empty})$ and $R_a(\mathit{flush})$. Given that the timeout of the mix is set to T seconds, the average amount of time needed by the attacker is:

$$T_a = R_a \cdot T \ \ seconds.$$

Assuming that the mix would receive during the attack m_u messages per second, the total amount of messages the adversary needs to delay (or remove) from the links is $N_u = m_u \cdot T_a$. The average number of messages generated by the attacker is N_M for the first round and $N_M \cdot P_M$ messages in subsequent rounds. The total number of messages the adversary generates on average to deploy the attack is:

$$N_a = N_M + N_M \cdot P_M \cdot (R_a - 1) \ .$$

Due to the probabilistic nature of the binomial mix, the attacker only succeeds with probability $1-\epsilon$. Therefore, with probability ϵ there is at least one unknown message in the mix. In this particular case, the attacker can detect his failure if during the flushing phase more than one unknown message leaves the mix in the same round (and there is no dummy traffic policy), which happens with probability P_M^2 for the case of one unknown message (besides the target) being inside the mix. With probability $P_M(1-P_M)$ the target message leaves the mix alone, and the attack is successful. Also with probability $P_M(1-P_M)$, the other unknown message leaves the mix first, and the attacker follows a message that is not the target without noticing. Finally, with probability $(1-P_M)^2$, both messages stay in the pool and the situation is repeated in the next round.

Therefore the attacker succeeds tracing the right message in the following cases:

- The attacker succeeded emptying the mix. This occurs with probability $1-\epsilon$.

- One unknown message (or more) was left in the pool (happens with probability ϵ), but the target leaves first the mix and those unknown messages do not interfere with the attack. We assume that the probability of two or more unknown messages is negligible compared to the probability of one unknown message staying in the pool. The probability of the target leaving before the unknown message is $P_M(1-P_M)\sum_{i=0}^{\infty}(1-P_M)^{2i}$.

- The attacker detected two unknown outputs in the same round and realized that one unknown message had stayed in the pool. This happens with probability $P_M^2\sum_{i=0}^{\infty}(1-P_M)^{2i}$. The adversary may choose one of the two outputs at random, succeeding with probability 50%. Note that if the adversary has context information that indicate likely recipients, the adversary may succeed with a higher probability.

The adversary fails when there were remaining unknown messages in the mix *and* an unknown message left the mix *before* the target. This happens with probability $\epsilon \cdot P_M(1-P_M)\sum_{i=0}^{\infty}(1-P_M)^{2i}$. Also, the attacker fails if two messages leave at the same time and the adversary decides that the target is the wrong one. This happens at maximum with probability 50% in a scenario with probability of happening of $P_M^2\sum_{i=0}^{\infty}(1-P_M)^{2i}$.

When we make the computations, we find that the adversary identifies the right target with average probability $\Pr(O_{i,j}) = (1-\epsilon) + 0.5\epsilon$, and fails with probability 0.5ϵ. The remaining anonymity of the message is, on average:

$$H = -(1-0.5\epsilon)\log_2(1-0.5\epsilon) - 0.5\epsilon\log_2(0.5\epsilon) \ .$$

We have assumed that ϵ has a small value. For $\epsilon = 0.01$, the remaining anonymity is $H = 0.046$. Although the adversary needs to spend more effort in the emptying phase (i.e., more rounds of attack) in order to reduce ϵ as much as possible, the fact is that ϵ decreases very fast after a few rounds of attack.

5.5 The Blending Attack on Continuous Mixes

As explained in Chapter 3, continuous or *Stop-and-Go* mixes [KEB98] do not function in rounds, like pool mixes. Instead, they apply a delay to each message independently that is chosen from an exponential distribution $Exp(\lambda)$. In order to make a fair comparison between continuous and pool mixes, we set the average delay introduced by the continuous mix to the same value as the average delay experienced by a message going through a timed pool mix, $\frac{T}{P_M}$. The parameter of the exponential distribution from which the delay of the messages is chosen, is the inverse of the average delay, $\lambda = \frac{P_M}{T}$. The probability density function of the exponential distribution is expressed as:

$$f(t) = \lambda e^{-\lambda t} \ .$$

Figure 5.3 represents the shape of the exponential probability density function. For a given message which arrives at time $t = 0$ to the continuous mix, the probability of leaving before time t is given by the cumulative distribution function:

$$F(t) = 1 - e^{-\lambda t} \ .$$

Figure 5.4 shows the exponential cumulative density function. If the adversary started observing the continuous mix at time $t = 0$, and we assume the mix contained M_0 messages at that time, the probability that all those messages

have left at time t is given by:

$$(1 - e^{-\lambda t})^{M_0} \quad .$$

Note that from the moment the adversary starts observing the mix, he can count the number of inputs and outputs and update the internal state of the mix with each received or sent message. If the adversary has been observing the mix for T_p seconds, and $T_p >> \frac{1}{\lambda}$, the probability that a message, which arrived before the observation started, is still in the mix is:

$$\epsilon = e^{-\lambda T_p} \quad .$$

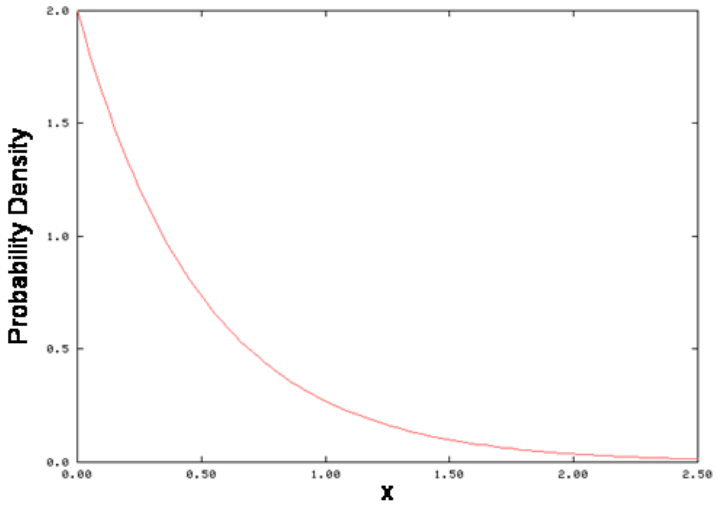


Figure 5.3: Exponential Probability Density Function ($\lambda = 2$)

If the adversary wants to be sure with probability $1 - \epsilon$ of the number of messages contained in the mix, he must observe it T_p seconds, where T_p can be computed from the formula presented above:

$$T_p = \frac{-\ln(\epsilon)}{\lambda} \quad .$$

Note that this is an approximation: as we are considering T_p to be large compared to the average delay, the probability of two or more undetected messages is negligible compared to the probability of just one undetected message staying inside the mix.

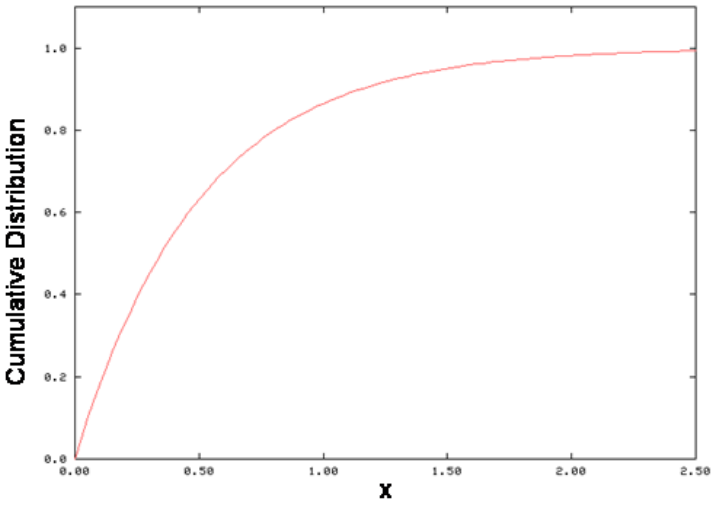


Figure 5.4: Exponential Cumulative Distribution Function ($\lambda = 2$)

The emptying phase. When the adversary detects the target message, he starts delaying all the inputs going to the continuous mix, including the target. In this scenario, however, the adversary does not need to generate any attacker message, as continuous mixes treat each message independently. The adversary then, only needs to count until the U initial unknown messages contained in the mix are sent out.

The flushing phase. Once the mix does not contain any unknown message, the adversary lets the target into the mix. On average, the message will be delayed $\frac{1}{\lambda}$, which we have set to the average delay of the timed pool mixes:

$$T_a(\mathit{flush}) = \frac{T}{P_M} \ .$$

The total time of active attack, T_a is the sum of $T_a(\mathit{empty})$ and $T_a(\mathit{flush})$. Note that with this type of mixes, the attacker does not need to generate any messages. Therefore we find that:

$$N_a = 0 \ .$$

Given that the rate of unknown messages getting to the mix is on average m_u messages per second, the total number of delayed messages is:

$$N_u = m_u \cdot T_a \ .$$

As in the case of binomial mixes, we can see that the adversary may arbitrarily reduce his uncertainty on the state of the mix (and therefore on the remaining anonymity) by observing it longer.

5.6 The Blending Attack on Pool Mixes with Dummy Traffic

In this section, we do a preliminary study on the deployment of blending attacks on pool mixes that generate dummy traffic. We leave for future work the computation of the effort needed to deploy the emptying phase of the blending attack in scenarios with dummies. This effort can be computed following the same reasoning as in the examples presented in the previous sections. Here, we focus on analyzing the effort required for the flushing phase and the remaining anonymity provided by dummies in the most favorable circumstances for the attacker. We assume the attacker has succeeded in removing all unknown messages from the pool of the mix in the emptying phase.

The mixes presented in this section generate and insert d_i messages at round i, where d_i is chosen from a random distribution of which we make abstraction.

5.6.1 Deterministic Mix with Dummy Traffic Inserted at the Output

In this section, we assume the attacker has successfully emptied the mix of unknown messages in the first phase of attack. We study the effort of the attacker to deploy the flushing phase of the attack and we proceed to compute the remaining anonymity of the target message.

Once the mix is emptied of unknown messages, the pool contains $N_M(1-P_M)$ messages. The adversary sends the target message to the mix, together with $N_M(1-P_M)-1$ of his messages. At the output, there are $s_i = N_M \cdot P_M + d_i$. If the adversary observes $N_M \cdot P_M$ of his messages, he knows that the target is still in the pool, and proceeds to another round of attack. When the attacker recognizes only $N_M \cdot P_M - 1$ at the output, he knows that one of the unknown messages is the target.

In this case, the attacker detects the round in which the target message is being sent by the mix. The average number of rounds needed to flush the target is, as in the case without dummies:

$$R_a(\mathit{flush}) = \frac{1}{P_M} \quad .$$

Consequently, the effort required to flush the message with 100% certainty

through a deterministic mix which generates dummy traffic is the same as for the case without dummy traffic.

Although the attacker can detect the round in which the target message is flushed, he still cannot distinguish between the target message and the dummies. Therefore, the remaining anonymity is provided by the d_k dummies inserted in the round k in which the target message has been flushed:

$$H = -\sum_{j=1}^{d_k+1} \frac{1}{d_k+1} \log_2(\frac{1}{d_k+1}) = \log_2(d_k+1) \ .$$

5.6.2 Binomial Mix with Random Dummy Traffic Inserted at the Output

As in the previous case, we assume here that the pool has been successfully emptied of unknown messages. At this point, the attacker knows the number of his messages contained in the pool, and that no unknown messages remain.

In the first flushing round, the adversary fills the pool of the mix with the target and as many of his messages as necessary to have $N_M - 1$ of them in the pool. At the output of round k, the adversary observes $s_k = m_k + u_k$, m_k of his messages and u_k unknown messages. As the number of messages selected from the pool cannot be predicted from the number of messages in the pool (N_M in our case), the u_k unknown messages of the output may or may not include the target. The adversary has no means to determine if the output contains the target or if, on the other hand, is nothing but a set of dummies.

Due to this uncertainty, the adversary must attack the mix for several rounds, until the probability of the target still being inside falls below ϵ.

The target is sent in each round with probability P_M, and kept in the pool with probability $1 - P_M$. If the adversary wants to make sure with probability $1 - \epsilon$ that the target has been sent out, he must wait $R_a(\mathit{flush})$ rounds:

$$(1 - P_M)^{R_a(\mathit{flush})} < \epsilon \ .$$

Note that the average number of rounds needed to flush the message increases considerably with respect to the deterministic mix case. For $P_M = 0.6$, the average number of rounds of observation is 1.6 rounds for deterministic mixes with dummy traffic inserted at the output. In the binomial case, in order to flush the message with 99% probability ($\epsilon = 0.01$) we need at least 5 rounds. This increase in the number of rounds of active attack has an impact in the number N_a of messages the adversary needs to generate, the number N_u the adversary has to delay, and the total time required to complete the attack.

We proceed now to compute the remaining anonymity H of the target message. We assume the adversary attacks the mix a sufficient number of rounds $R_a(\mathit{flush})$ to guarantee that the mix is clean of unknown messages (that arrived to the mix before the attack started) with probability $1-\epsilon$, where ϵ is arbitrarily small.

In each round i of the $R_a(\mathit{flush})$ rounds of attack, the adversary observes u_i unknown outputs. We denote the unknown output j (with $j = 1..u_i$ and $i = 1..R_a(\mathit{flush})$) of round i as $O_{i,j}$. The probability assigned to $O_{i,j}$ of being the target output is:

$$\Pr(O_{i,j}) = \frac{P_M(1-P_M)^{i-1}}{u_i} \ .$$

The remaining anonymity of the target output is given by the entropy H of the distribution of probabilities $\Pr(O_{i,j})$. On average, the remaining anonymity is:

$$H = -\sum_{i=1}^{R_a(\mathit{flush})} \sum_{j=1}^{u_i} \Pr(O_{i,j}) \log(\Pr(O_{i,j})) \ ;$$

$$H = -\sum_{i=1}^{R_a(\mathit{flush})} P_M(1-P_M)^{i-1} \log(\frac{P_M(1-P_M)^{i-1}}{u_i}) \ .$$

This means that all the dummies sent in the rounds in which there is a probability of sending the target (this includes the rounds before and/or after the actual sending of the target) contribute to the anonymity, in contrast with the previous case, in which the round that includes the target is observable, and only the dummies sent in that particular round contribute to the remaining anonymity of the message. Note that the value of the remaining anonymity may be considerably large in this case. For an average of 10 dummies inserted per round, and $P_M = 0.6$, we obtain that the anonymity provided in this case is $H = 5$ (i.e., perfect indistinguishability among 32 messages), while the remaining anonymity in the case of the deterministic mix would be $H = 3.4$ (i.e., perfect indistinguishability among 11 messages).

5.6.3 Dummies Inserted in the Pool

In this scenario the mix inserts d_k dummies per round in the pool. Once these dummy messages are in the pool, they add to the real messages contained in it. The mix treats all messages in the pool equally. As in the previous case, we assume here that the attacker has successfully emptied the pool of unknown real

messages, and we leave the computation of the effort required to achieve it for future work.

At the end of the emptying phase, the pool contains M_k attacker messages and U_k unknown messages (which in this case are all dummies). Assuming that this mix achieves the maximum value P_M of its $P(n)$ function at N_M (like the other pool mixes we have considered in the chapter), the adversary needs to send U_k less attacker messages to maximize the probability of forwarding for the messages in the pool.

Therefore, the negative implication of inserting the dummies in the pool is that the adversary needs to generate a lower number N_a of attacker messages. The larger the number of dummies inserted per round in the pool, the smaller the number of messages the adversary needs to generate. Note that the sum of the dummies and attacker messages in the pool must be at least N_M.

This case is analogous to the binomial mix with dummies inserted at the output in the sense that the adversary cannot detect the round in which the target is forwarded by the mix. Note that this applies both to binomial and deterministic mixes that insert the dummies in the pool.

The effort required by the attacker in terms of attacker flushing rounds is the same as in the binomial case with dummies at the output: if the adversary wants to have certainty $1 - \epsilon$ of the message having been forwarded by the mix, then he may have to consider $R_a(\mathit{flush})$ flushing rounds:

$$(1 - P_M)^{R_a(\mathit{flush})} < \epsilon \ .$$

The result in terms of remaining anonymity is also analogous to the case of binomial mixes. Given that the adversary observes u_i unknown outputs per round, the remaining anonymity of the target is:

$$H = - \sum_{i=1}^{R_a(\mathit{flush})} P_M(1 - P_M)^{i-1} \log(\frac{P_M(1 - P_M)^{i-1}}{u_i}) \ .$$

Hence, the combination of deterministic mixes with dummies inserted at the output provides significantly worse remaining anonymity than either using binomial mixes, inserting the dummies in the pool or combining both.

5.7 Conclusions

In this chapter, we have studied the *blending* or $n - 1$ attack. We have defined a set of parameters that indicate the effort required to deploy this active attack on deterministic pool mixes (timed and threshold), binomial pool mixes, continuous mixes and pool mixes with dummy traffic.

For all these mixes, we have computed the remaining anonymity of a target message going through the attacked mix. We summarize below the key conclusions of our work:

- When looking at deterministic pool mixes, the attack is much faster against threshold mixes than against timed mixes. Because of this, the number of messages delayed is lower for threshold mixes. The number of messages generated by the adversary to complete the attack is the same for timed and threshold pool mixes. The remaining anonymity in both cases is zero.

- There are two ways of deploying the blending attack on binomial mixes. The first method implies that the attacker must observe the mix long before the attack takes place, in order to accurately estimate the internal state of the mix (number of messages inside). If this is the case, the effort of the attacker spent on the active part of the attack is the same as in the case of deterministic timed pool mixes. Alternatively, the adversary may start deploying the attack without an estimation of the internal state. In this case, the number of rounds of active attacker increases (with respect to the deterministic cases), which provokes an increase in other effort parameters, such as number of messages generated, number of messages delayed, and amount of time needed to complete the attack. Regarding the remaining anonymity of the message, it tends to zero linearly up to a logarithmic factor $-\epsilon \log_2(\epsilon)$ as the adversary selects a higher accuracy parameter $1 - \epsilon$.

- The attack on continuous mixes requires a similar effort as the binomial or deterministic timed cases in terms of time required to complete the attack and number of messages delayed. However, in the continuous case the adversary does not need to generate messages in order to deploy the attack. The remaining anonymity provided by these mixes tends to zero as the adversary increments the time of attack.

- Making a deterministic pool mix insert the dummies at the output does not increase the effort required to the attacker to deploy the flushing phase of the attack with respect to the case of deterministic timed pool mixes without dummies. However, it does increase the remaining anonymity of the message up to $H = \log_2(d_k + 1)$, where d_k is the number of dummies inserted by the mix in the round in which the target is forwarded.

- If we combine binomial mixes with dummy traffic inserted at the output, we find that the effort required to deploy the flushing phase of the attack increases with respect to the case of no dummies. More specifically, there is an increase in the number of rounds needed in the flushing phase, which induces an increase in the number of messages generated and delayed, as

well as the total time required to complete the attack. Moreover, the remaining anonymity of the message is considerably higher than in the case of deterministic mixes with dummy traffic

- Regardless of whether the mix is deterministic or binomial, when dummies are inserted in the pool the remaining anonymity has similar (high) values to the case of binomial mix with dummies at the output. The shortcoming of this configuration is that the effort of the attacker in terms of number of messages generated is lower than when the dummies are inserted in the pool. The other effort parameters are similar to those of the binomial mix with dummy traffic inserted at the output.

The issues which are left for future work include:

- Compute the effort required to deploy the emptying phase of the attack in pool mixes with dummy traffic.
- Analyze the effort and results of the blending attack on continuous mixes with dummy traffic.
- Design dummy policies to provide a strong protection against blending attacks.

Chapter 6

Comparison between two practical mix designs

Things are as they are. Looking out into the universe at night,
we make no comparisons between right and wrong stars,
nor between well and badly arranged constellations.
– Alan Watts

6.1 Introduction

In this chapter, we apply the knowledge developed in the previous chapters to evaluate working implementations of mixes providing an anonymous email service. The mixes studied fit the taxonomy presented in Chapter 3, and we use the metrics introduced in Chapter 2 to measure anonymity. More specifically, we apply the traffic analysis attack analyzed in Chapter 4 to a real life case.

The objective of this work is to have quantitative results on the anonymity actually provided by two mix software implementations in wide deployment, to test the actual anonymity provided to the users of the remailer service, and to compare the two different designs. As individual nodes are the basic component to the network of mixes, we aim to provide information to be considered when choosing this component. We have used as input real-life data gathered from a popular remailer, and simulated the behavior of the mix.

The results presented in this chapter have been extracted from our original work *Comparison between two practical mix designs*, published in the proceedings of the *9th European Symposium On Research in Computer Security (ESORICS 2004)* [DSD04].

This chapter is organized as follows: Section 6.2 introduces Mixmaster, the pool mix under study, while Sect. 6.3 introduces Reliable, the continuous mix. Section 6.4 describes the attack model considered, the methods used to compute the anonymity, and other issues related to the implementation of the Java simulators used in this research. The results are exposed in detail in Sect. 6.5 (analysis of the input traffic), Sect. 6.6 (analysis of the anonymity provided by Mixmaster) and Sect. 6.7 (analysis of Reliable). Finally, Sect. 6.8 describes other factors that influence anonymity and their consequences for Mixmaster and Reliable.

6.2 Mixmaster

Mixmaster is a working implementation of the deterministic timed pool mix proposed by Cottrell in [UMS03, Cot], used for providing an anonymous email service. Deterministic timed pool mixes have been described in Chapter 3.

Mixmaster version 3.0, as well as Reliable, also optionally supports the older "Cypherpunk" remailer message format. For the purposes of this work, we are assuming that the remailers are being operated without this support. As anonymity sets for the two protocols generally do not overlap, this does not impact our results. The Cypherpunk remailer protocol is known to contain numerous flaws, and should not be used if strong anonymity is required [Cot, DDM03].

Mixmaster is a timed mix that has a *timeout* of 15 minutes. During this period of time, it collects messages that are placed in the pool of the mix. When the timeout expires, the mix takes a number of messages from the pool that are forwarded to their next destination, which may be another mix or a final recipient. The number s of messages sent in a *round* (one cycle of the mix) is a function of the number n of messages in the pool:

```
if (n<45) s=0;

else if (0.35*n < 45)s=n-45;

else s=0.65*n;
```

Pool mixes as Mixmaster are represented in the Generalized Mix Model (GMM) by their characteristic function $P(n)$, which represents the probability of a message from the pool of being forwarded as a function of the number of messages in the pool. The function $P(n)$ determines the number of messages to be withdrawn from the pool. The messages are chosen uniformly at random. At the expiration of the timeout, Mixmaster forwards $s = nP(n)$ messages out of the n messages in the pool. The $P(n)$ function of Mixmaster in the GMM is shown in Fig 6.1.

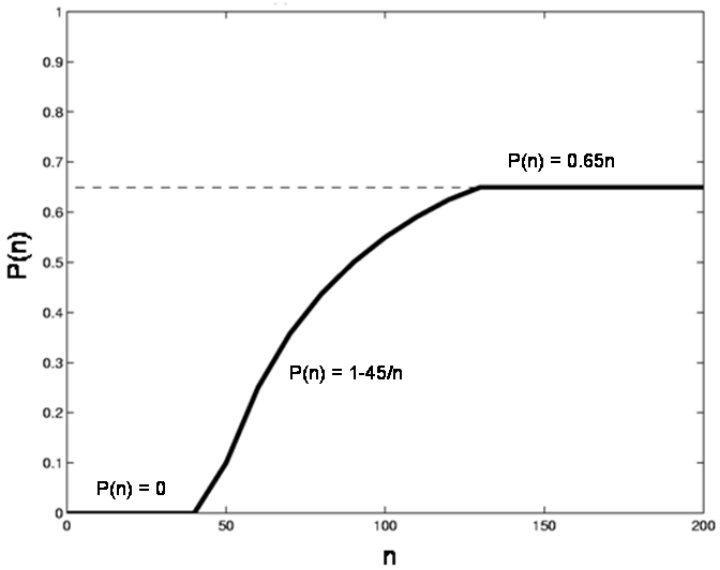


Figure 6.1: Mixmaster in the GMM

6.3 Reliable

Reliable is loosely based on the Stop-and-Go (*S-G Mix*) mix proposed by Kesdogan *et al.* in [KEB98]. In S-G mixes (also called *continuous mixes*), the users generate a random delay from an exponential distribution, as explained in the Chapter 3 of this thesis. The mix holds the message for the specified delay and then forwards it. The messages are reordered by the randomness of the delay distribution. This mix sends messages continuously: when it has been kept for the delay time it is sent out by the mix.

Reliable interoperates with Mixmaster on the protocol level by using the Mixmaster message format for packet transfer. Reliable uses a variant of the *S-G mix* design. The theoretical S-G mix design assumes that the delay distribution adapts to the traffic load; that is, the users should set the average delay according to the amount of input traffic the mix is receiving. This feature is not implemented in Reliable, which has a static delay. True S-G mixes also implement timestamps in order to prevent active attacks ($n-1$ attacks in particular), and would therefore require a service to provide such information. Regardless, as the message protocol was originally designed for a network of pool mixes, these timestamps are not used. Reliable thus does not provide any resistance to this kind of active attack.

In Reliable, the delay may be chosen by the sender from an exponential distribution of mean one hour. If the sender does not provide any delay to the mix, then the mix itself picks a delay from a *uniform* distribution that takes values between one and four hours. Note that these parameters of the delay distributions are configurable, and therefore many remailer operators may set them lower in order to provide a faster service.

6.4 Simulators

We have implemented Java simulators for Reliable and Mixmaster. We have fed the simulated mixes with real input, obtained by logging a timestamp each time a message arrived to a working Mixmaster node (note that the information we logged does not threaten the anonymity of the users of the mix). We have used four months of incoming traffic (July-November 2003) to obtain the results presented in Sect. 6.5, Sect. 6.6 and Sect. 6.7.

In order to make a fair comparison, we have set the mean of the exponential delay of Reliable (default 1 hour) to be the same as provided by Mixmaster for the given four months of input (43 minutes). We have also made some simulations for Reliable with mean 1 hour, and the results obtained do not differ significantly from the ones presented in this chapter. We have assumed users choose their delays from an exponential distribution. The mix-chosen uniform delay option has not been taken into account, due to the infeasibility of implementing the algorithm presented for uniform delays in Chapter 4 for such large amounts of inputs. Our simulators can be reused with a different input stream or other types of mixes.

The simulators log the delay and the anonymity for every message. Mixes are empty at the beginning of the simulation. The first message that is taken into account for the results is the one that arrives when the first input has been flushed with 99% probability. All messages flushed after the last arrival to the mix are also discarded for the results. This is done in order to eliminate the transitory initial and final phases. In our simulations, the number of rounds discarded in the initial phase is 3, and the number of rounds discarded in the final phase is 39. The total number of rounds for our input traffic is 11.846.

6.4.1 Attack Model and Anonymity Metrics

The adversary considered in this chapter is a global, external, passive and static attacker which performs traffic analysis on the mix, and it is the same attacker as described in Chapter 4. In this chapter, we apply the theoretical analysis developed in Chapter 4 to a real life scenario, in order to study and compare the

anonymity properties of two working implementation of mixes, one pool mix and one continuous mix.

The attacker observes the incoming and outgoing messages going through the mixes. He knows all internal parameters of the mix so he can effectively compute the anonymity of the messages using the formulas for pool and continuous mixes presented in Chapter 4.

6.5 Analysis of the Input Traffic

It is a common assumption in the literature that the arrivals at a mix node follow a Poisson process. We have analyzed the input traffic, and found that it does not follow a Poisson distribution nor can it be modeled with a single time-independent parameter.

A Poisson process is modeled by a single parameter λ representing the expected amount of arrivals per (fixed) time interval. If the arrivals to a mix are assumed to follow a Poisson process with an average of λ arrivals per time interval Δt and we denote the number of arrivals in such a time interval by X, then X is Poisson distributed with parameter λ: $X \sim \text{Poiss}(\lambda)$. It is important to note that λ is *time-independent*.

In our statistical analysis we first *assumed* that the process of arrivals *was* a Poisson process and we estimated the parameter λ. The latter was done by taking the maximum likelihood estimate given the number of arrivals per time interval $\Delta t = 15$ minutes ($N = 11800$). We also constructed a 95% confidence interval for this estimate. In this way we found $\hat{\lambda} = 19972$ with confidence region $[19891; 20052]$. Then we performed a goodness-of-fit test to determine if we can reject the hypothesis

$$H_0 : \text{the number of arrivals per time interval } \sim \text{ Poiss}(\bar{\lambda}) \ ,$$

where $\bar{\lambda}$ varies over the constructed confidence interval. The goodness-of-fit test we used is the well-known χ-square test (df=$n - 1$=11802). Using a significance level of 0.01, the null hypothesis gets rejected (χ-value=826208)!

In Fig. 6.2 we show the number of messages received by the mix per hour. Figure 6.3 shows the evolution of the arrivals per day. We can observe that the traffic that arrived at the mix during the first month is much heavier than in the following three months. The drop in traffic load was possibly caused by the fact that the mix went off-line for a short period of time. This made the statistics on its reliability show worse confidence levels. Fewer users would then choose this particular node as part of their path. We have applied our analysis to both the high and low traffic periods of time independently, without finding a traffic pattern that responds to a known probability distribution. This shows that the

input traffic pattern that gets to a mix node can be highly unpredictable and that the assumption of λ being time-independent does not necessarily hold.

Figure 6.4 shows the frequency in hours and Fig. 6.5 the frequency in days of receiving a certain number of arrivals. We can see that in most of the hours the mix receives less than 20 messages.

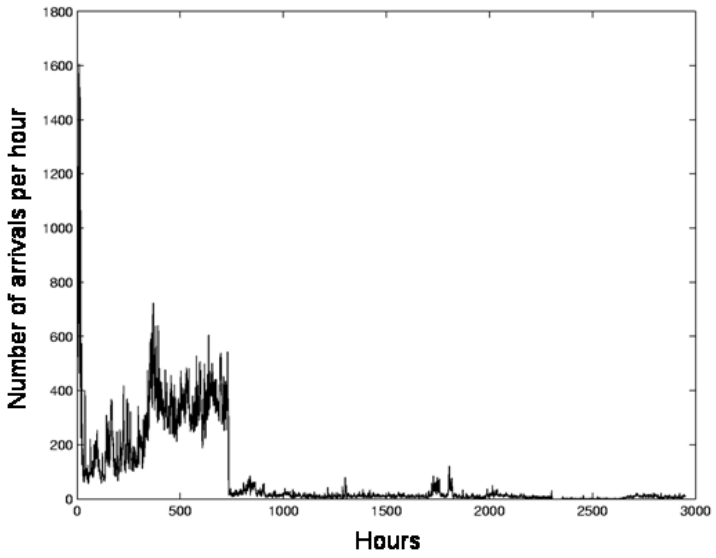


Figure 6.2: Number of Observed Arrivals per Hour

6.6 Analysis of Mixmaster

We have simulated a Mixmaster node as explained in Section 6.4. Mixmaster is a pool mix and processes messages in batches. The recipient anonymity of each message that arrived in a round is the same. Equivalently, all outputs of a round have the same sender anonymity value. In this section we show the results obtained in our simulation.

In Fig. 6.6 we show the correlation between the recipient anonymity and the delay for every message going though Mixmaster. Every message is represented in the plot by a dot defined by the delay experienced by the message and its recipient anonymity. We can see in Fig. 6.7 that sender anonymity takes similar values.

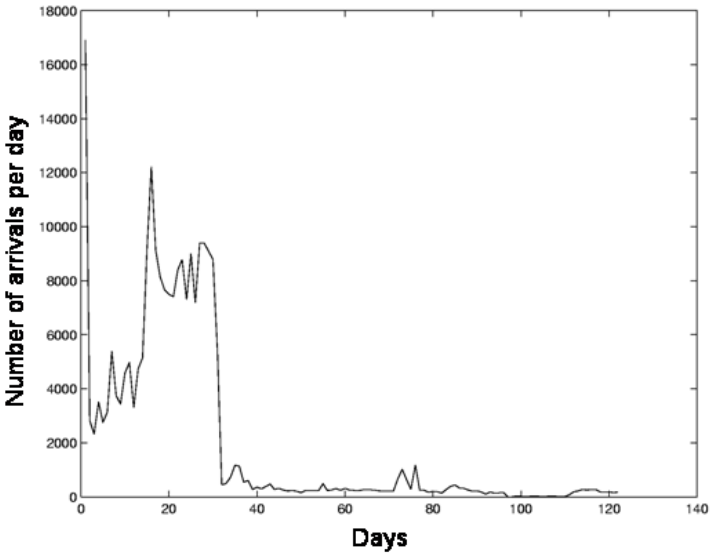


Figure 6.3: Number of Observed Arrivals per Day

Figures 6.8 and 6.9 extend to three dimensions, and show the correlation between anonymity, delay and number of inputs per round. In these figures we can see that the lower anonymity and higher delay values are correlated with a low traffic load. An increase in the traffic load greatly reduces the delay while anonymity increases. We can also confirm that the both sender and recipient anonymity behave similarly for a given traffic load.

The first conclusion we come to when observing the figures is that there is a lower bound to the anonymity of Mixmaster. It is worth noting that, so far, we do not know any theoretical analysis of pool mixes able to predict the anonymity a pool mix provides, and prior to this analysis there were no figures on the anonymity that Mixmaster was actually providing. With this simulation, we can clearly see that Mixmaster guarantees a minimum sender and recipient anonymity of about 7. This means that the sender (recipient) of a message gets a minimum anonymity equivalent to perfect indistinguishability among $2^7 = 128$ senders (recipients).

We can see that the minimum anonymity is provided when the traffic (arrivals) is low. As the traffic increases, anonymity increases, getting maximum values of about 10 (i.e., equivalent to perfect indistinguishability among $2^{10} = 1024$)

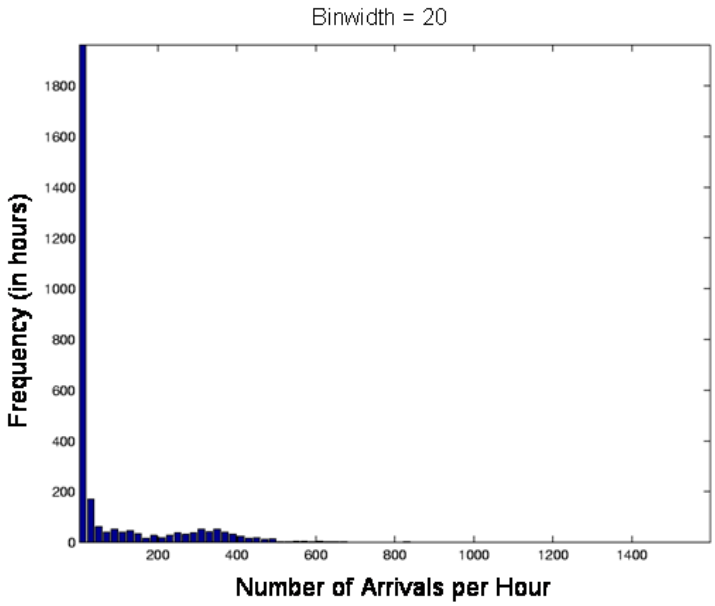


Figure 6.4: Frequency Analysis of Inputs in Hours

senders or recipients. We also observe that the delays of the messages don't take high values, unless the traffic load getting to the mix is very low.

In order to study the behavior of the mix under different traffic loads, we have plotted values of delay and anonymity obtained in the simulation for the rounds with few arrivals (low traffic), intermediate number of arrivals (medium traffic), and many arrivals (high traffic).

We have selected the low, medium, and high traffic taking into account the data statistics of the arrival process:

Low traffic: all rounds where the number of arrivals was between the first and third quartile ($0 \leq$ number of arrivals $\leq$ 17); hence 50 percent of the rounds are denoted as low traffic.

Medium traffic: all rounds where the number of arrivals was greater than the third quartile but lower than the outlier bound (17 $<$ number of arrivals $\leq$ 41).

High traffic: all rounds with outlier values for the incoming messages (number of arrivals $>$ 41).

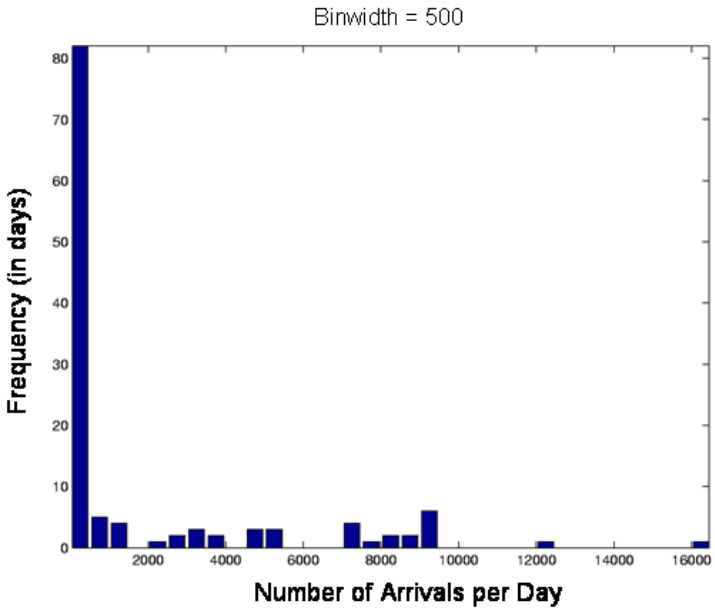


Figure 6.5: Frequency Analysis of Inputs in Days

In Figure 6.10 we show the minutes of delay of every message (the x-axis indicates the evolution in time, i.e., we slowly move towards the right with each new round). We can see that the delay only takes high values when the traffic is low. The high delay values registered at the end of the medium and high traffic load figures can be explained by the fact that those rounds were followed by the drop in traffic. Note that the delay experienced by a message going through a pool mix like Mixmaster depends on the traffic load of the round in which the message arrived and the following ones, with decreasing importance. A message arriving to the mix at round R leaves the mix k rounds after with probability ($k = 0$ means that the message leaves the mix in the same round it arrived):

$$\Pr(k) = P(n_R) \ , \ \ k = 0 \ .$$

$$\Pr(k) = P(n_{R+k}) \prod_{i=0}^{k-1} (1 - P(n_{R+i})) \ , \ \ k > 0 \ .$$

In Figure 6.11 we show the recipient anonymity of every message (the sender anonymity presents very similar characteristics). We can see that as the traffic

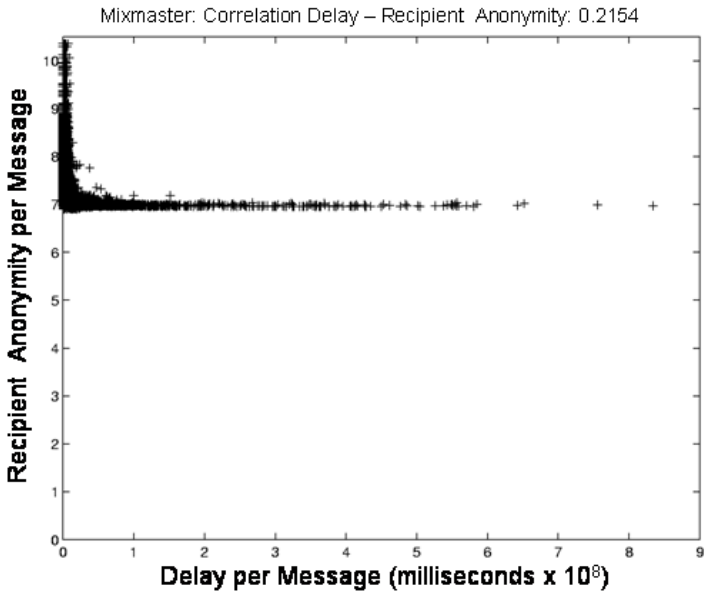


Figure 6.6: Correlation Recipient Anonymity - Delay for Mixmaster

increases, the anonymity provided takes higher values. No matter how low the traffic load is, the anonymity provided by Mixmaster is always above 7.

6.7 Analysis of Reliable

The theoretical method proposed in [KEB98] that gives a probabilistic prediction on the anonymity provided by Reliable is based on the assumption of Poisson traffic. As we have seen, this assumption does not hold for anonymous email traffic.

We have simulated a Reliable mix as explained in Section 6.4. Reliable treats every message independently: when it receives a message it delays it for a predetermined amount of time (selected from an exponential distribution) and then forwards it. We represent a star, '*', per message.

In Figures 6.12 and 6.13, we present the sender and the recipient anonymity provided by Reliable for the real stream of inputs we have considered. We can see that the anonymity takes minimum values close to zero, which means that some of the messages can be trivially traced by a passive attacker. The maximum values

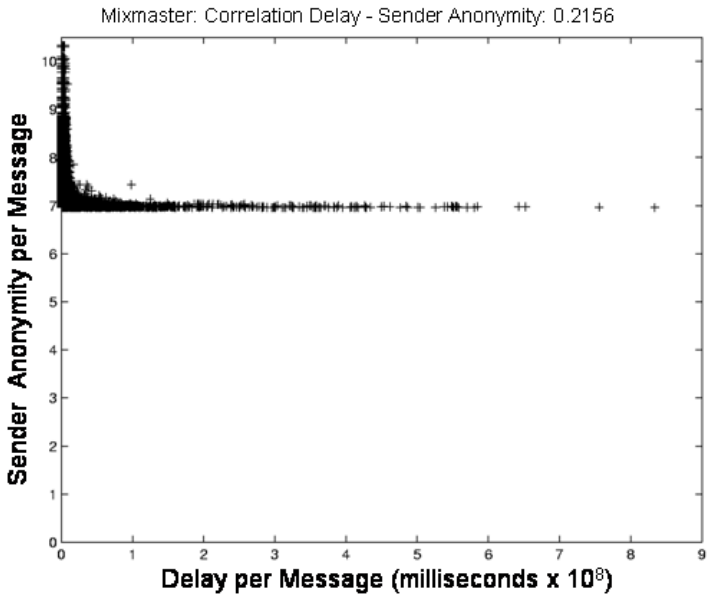


Figure 6.7: Correlation Sender Anonymity - Delay for Mixmaster

of Reliable's anonymity for this input are lower than Mixmaster's maximums. Figures 6.14 and 6.15 show the highly correlated values of sender and recipient anonymity for both Reliable and Mixmaster, respectively. We can clearly see that for Reliable some of the messages get nearly no anonymity, while the ones of Mixmaster get at least sender and recipient anonymity 7.

6.8 Other Factors that Influence Anonymity

We have evaluated the anonymity strength of the mixing algorithms implemented in Mixmaster and Reliable. Additional factors have a direct impact on the anonymity provided by the system. Concerns such as the security of the underlying operating system, host server integrity, proper implementation of the cryptographic functions provided by the remailer software, and likelihood of administration mistakes all contribute to the overall anonymity these software packages can provide. We assume that no active attacks against the software occurred during the development or compilation process, though additional concerns are present in that area [Tho84].

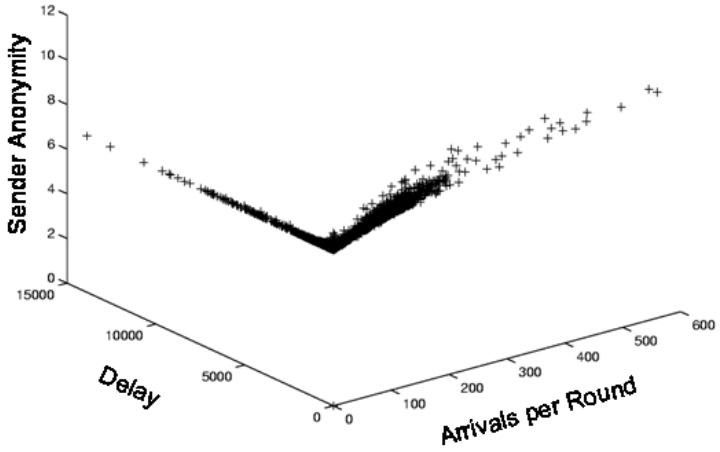


Figure 6.8: 3D Plot: Sender Anonymity - Delay - Traffic Load for Mixmaster

This work does not aim to be an in-depth analysis of the full spectrum of host-attacks against remailer nodes. Nevertheless, it is important to mention some significant differences between Reliable and Mixmaster that may affect their ability to provide adequate anonymity for their users.

6.8.1 Host Server Integrity

The security of an operating mix is dependent on the security of the underlying host server. Many factors can impact the underlying system's security. Some considerations include shared access to the system by untrusted users, access to key material on disk or in memory, and the ability to insert shims to intercept dynamically loaded libraries called by the remailer software [Tha03].

Reliable is limited to operation on the Windows platform. Mixmaster is portable, and has been known to run on a wide variety of operating systems. There have been instances of remailers based on the Mixmaster 3.0 codebase operating on SunOS, Solaris, SunOS, AIX, Irix, BeOS, MacOS X, Windows NT (natively and through the use of Cygwin), Windows 2000 (natively and through the use of Cygwin), Windows XP (through the use of Cygwin), FreeBSD, NetBSD, OpenBSD, and multiple versions of Linux.

Host server security is ultimately the responsibility of the remailer operator.

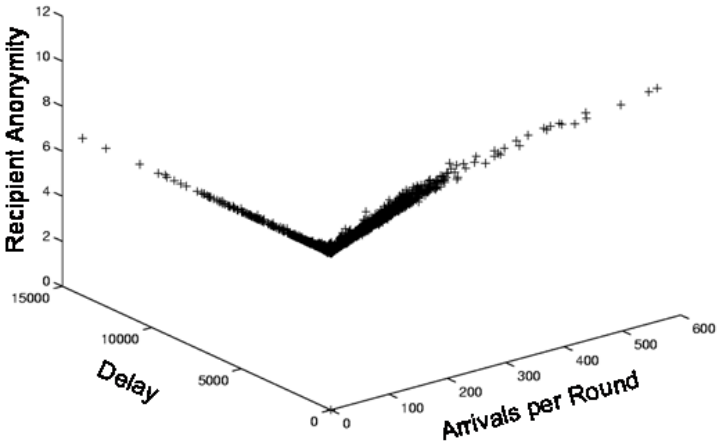


Figure 6.9: 3D Plot: Recipient Anonymity - Delay - Traffic Load for Mixmaster

6.8.2 User Interface Issues

In a privacy application client, an intuitive user interface is essential in order to ensure that the software is used consistently and correctly [Sas02]. A greater level of skill can safely be assumed when designing privacy software that is intended to be operated as a service, however. Most anonymity systems, including mix implementations, do imply a significant degree of complexity. Since the operation of a public Internet service involves the correct configuration and maintenance of the host server, this necessary complexity is acceptable as long as the operator's skill level is sufficient. The level of skill required to properly install, configure, and operate a mix node should not exceed what is required to properly install, configure, and operate the server itself.

The software packages we evaluated differed with regard to their interface complexity in a number of areas.

In general, Reliable has a greater "ease of use" factor with respect to its interface. Mixmaster automates many important tasks, such as adaptive dummy generation, key rotation and key expiration announcement, and integrates more easily with the host MTA (MTA stands for Mail Transport Agent, e.g. sendmail or postfix). Reliable's installation process is easier, but its build process requires the use of third-party commercial applications and assumes experience with Windows development, so most users will install a pre-compiled binary. Compilation of Mixmaster is performed through a simple shell script.

At first glance, it appears that Reliable will be easier for hobbyists to operate than Mixmaster. However, Mixmaster's difficulty does not rise above the diffi-

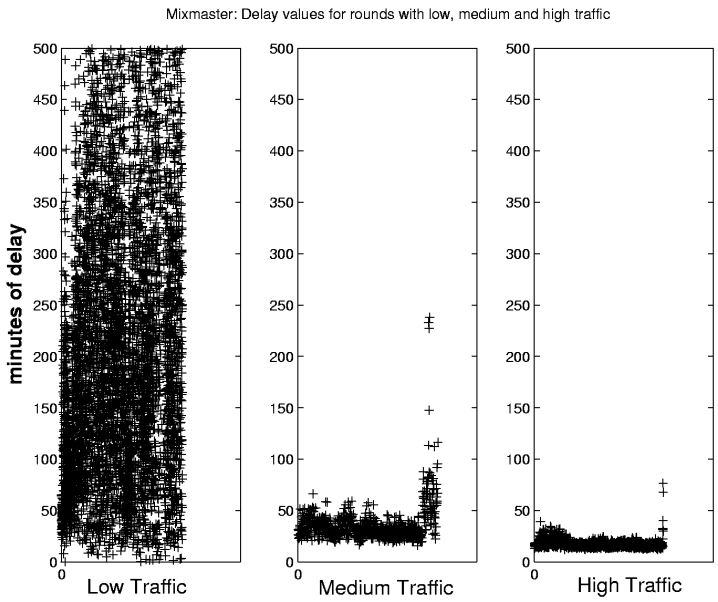


Figure 6.10: Delay Values for Mixmaster

culty of maintaining a secure Internet-connected server, and thus has little effect on the overall security of a mix node deployment.

6.8.3 Programming Language

While the most critical factor in the creation of secure code is the manner in which it is written, some languages lend themselves to greater risk of exploitable mistakes. An inexperienced or unskilled programmer will always be in danger of making an application insecure. The choice of programming language merely sets the bar for the required level of experience and ability necessary to develop applications in that language safely. Thus, when evaluating the likelihood of the existence of exploitable code in an application, it is worthwhile to consider the programming language used to create that application. Mixmaster is written in C, while Reliable is written in Visual Basic. Since neither Mixmaster nor Reliable was written by seasoned software developers, we assume a level of experience that would allow for simplistic security mistakes. The bulk of the code for Mixmaster 3.0 was written by Ulf Möller as his first major software development project while completing his undergraduate computer science degree [MÖ2]. He has since gained respect as a skilled cryptographic software developer for his open source and

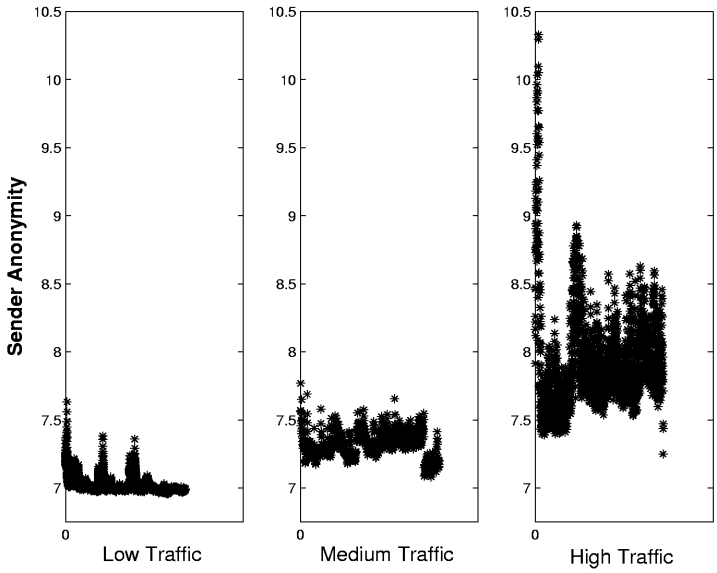


Figure 6.11: Anonymity Values for Mixmaster

proprietary development projects. Reliable was authored under a pseudonym, and we can only speculate about the level of experience of its author. (There has been no known communication with the author of Reliable since February, 2000).

6.8.4 Source Code Documentation

To facilitate source code review and verification of an application's correctness with regard to its implementation of a protocol, it is beneficial for there to be both good commenting in the source code and a clear specification for its behavior.

While neither program is sufficiently commented or written clearly enough to allow a reviewer to easily learn how either system works by reading the source code alone, there exists a complete specification of the Mixmaster node behavior [UMS03]. No such specification or description exists for Reliable.

6.8.5 Included Libraries

In addition to the standard POSIX libraries provided by the compilation OS, Mixmaster 3.0 (the version of Mixmaster evaluated here) requires that the zlib [DG96]

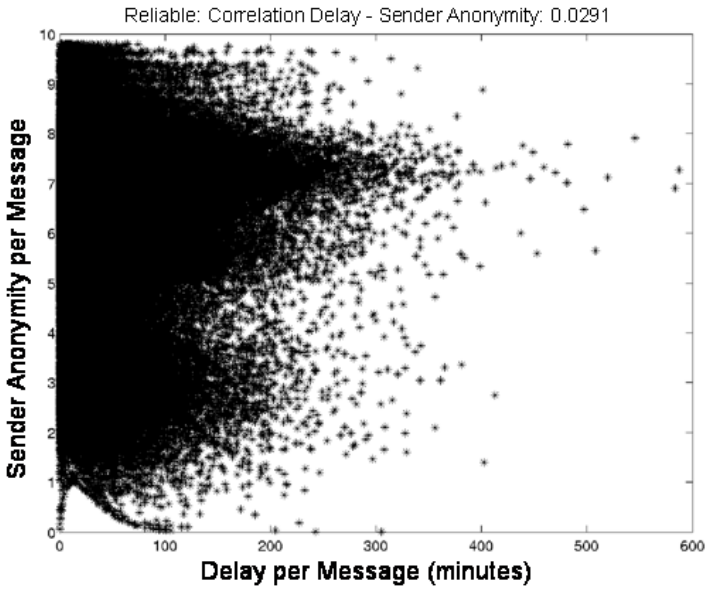


Figure 6.12: Correlation Delay - Sender Anonymity for Reliable

and OpenSSL [CEHL] libraries be included. Optionally, Mixmaster also links with pcre [Haz] and ncurses [BHRPD].

Reliable requires many native Windows system calls as well as the third-party application, Mixmaster 2.0.4. Mixmaster 2.0.x has an entirely different codebase than that of Mixmaster 3.0. While Reliable relies on the Mixmaster 2.0.4 binary for some of its functionality, Reliable is an independent application in its own right, and should not be considered a mere extension to the Mixmaster codebase.

6.8.6 Cryptographic Functions

Both Mixmaster and Reliable avoid direct implementation of cryptographic algorithms when possible. Mixmaster 3.0 relies strictly on OpenSSL for these cryptographic functions. Any attackable flaws in the cryptographic library used to build Mixmaster that affect the security of the algorithms used by Mixmaster may be an attack against Mixmaster as well. It is understood that flaws in the cryptographic algorithms will affect the security of software that relies upon those algorithms. However, since most attacks on cryptographic applications are due to flaws in the implementation, care must be taken when evaluating the shared cryptographic libraries.

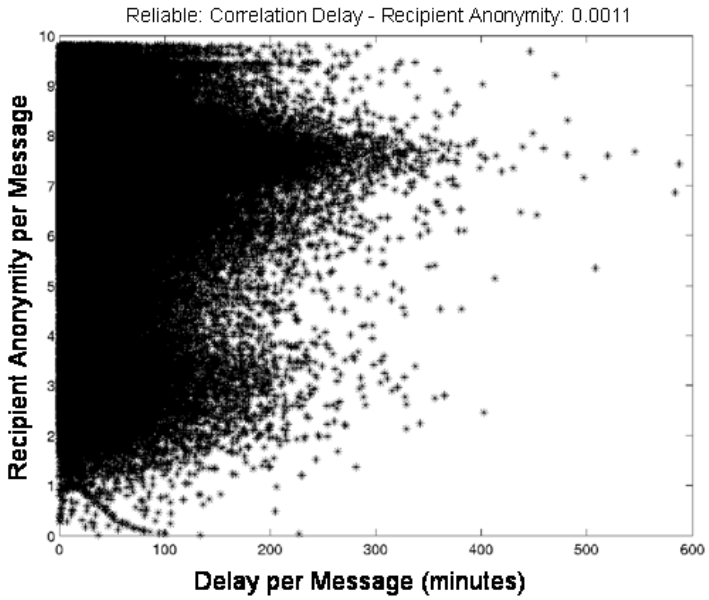


Figure 6.13: Correlation Delay - Recipient Anonymity for Reliable

Reliable abstracts the cryptographic operations one step further. To support the Mixmaster message format, Reliable acts as a wrapper around the DOS version of Mixmaster 2.0.4. Thus, any attack against the Mixmaster message format due to implementation flaws in Mixmaster 2.0.x will work against Reliable as well. Mixmaster 2.0.4 relies on the cryptographic library OpenSSL or its predecessor SSLeay for the MD5, EDE-3DES, and RSA routines. Prior to the expiration of the RSA patent, versions of Mixmaster 2.0.x offered support for the RSAREF and BSAFE libraries as well. The use of these versions of Mixmaster is largely abandoned.

6.8.7 Entropy Sources

The quality of the entropy source plays an extremely important role in both the pool mix and S-G mix schemes. In pool mix systems, the mixing in the pool must be cryptographically random in order to mix the traffic in a non-deterministic way. The timestamps that determine how long a message should be held by an S-G mix implementation must also be from a strong entropy source for the same reasons. In addition, the Mixmaster message format specifies the use of random data for its message and header padding.

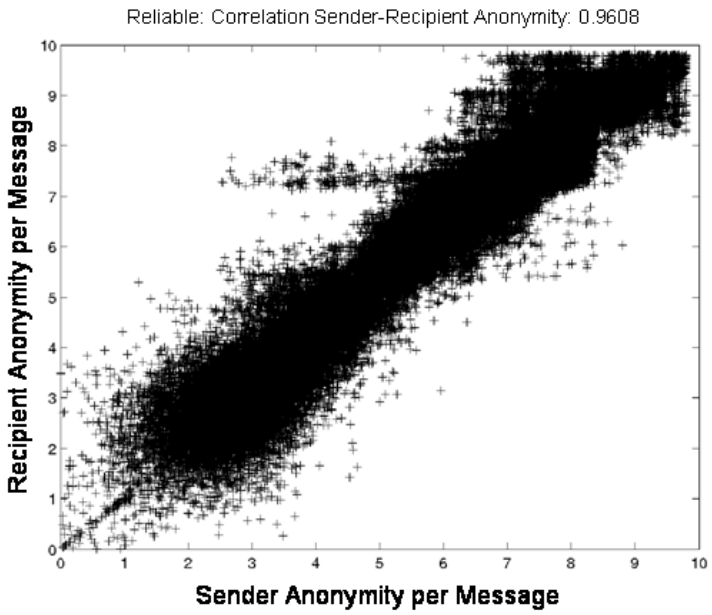


Figure 6.14: Correlation Sender - Recipient Anonymity for Reliable

Software is dependent on its underlying operating system for a good source of entropy. Cryptographic quality entropy is a scarce resource on most systems, and therefore the entropy sources provided by most modern operating systems actually provide PRNG output which has been seeded with truly-random data. Note that systems that employ the use of noisy diodes or other plentiful sources of entropy have less of a concern for entropy pool exhaustion.

Mixmaster uses OpenSSL's rand_ functions. OpenSSL relies on its internal PRNG seeded with various system sources to provide cryptographically strong entropy. Reliable uses the standard Windows system call, Rnd(), when obtaining entropy, with the exception of message and header padding (which is done by the supporting Mixmaster 2.0.4 binary). The Rnd() function is not a cryptographically strong source of entropy [Cor]. Rnd() starts with a seed value and generates numbers which fall within a limited range. Previous work has demonstrated that systems that use a known seed to a deterministic PRNG are trivially attackable [GW96]. While its use of Rnd() to determine the latency for a message injected into the mix is the most devastating, Reliable uses Rnd() for many other critical purposes as well.

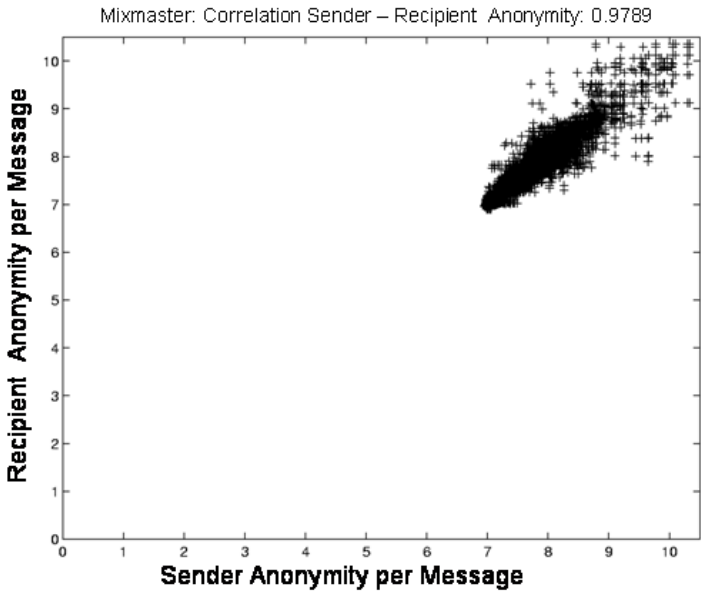


Figure 6.15: Correlation Sender - Recipient Anonymity for Mixmaster

6.9 Conclusions

In this chapter we have analyzed the traffic pattern of a real traffic stream going through a working mix node and found that the traffic is not Poisson, as it is commonly assumed in the literature. The traffic pattern is highly unpredictable. Therefore, no assumptions on the traffic should be made when designing a mix.

We measure the anonymity of the pool mix scheme used in Mixmaster and Reliable by applying information theoretic anonymity metrics. Our comparison of the two predominant mixing applications shows that Mixmaster provides superior anonymity, and is better suited for the anonymization of email messages than Reliable. Mixmaster provides a minimum level of anonymity at all times; Reliable does not. Reliable's anonymity drops to nearly zero if the traffic is very low. In high-traffic situations, Mixmaster provides a higher maximum anonymity than Reliable for the same stream of input: 10.5 of Mixmaster versus 10 of Reliable. We have shown that Mixmaster provides higher average anonymity than Reliable for the same input and same average delay. Due to its nature as a pool mix, Mixmaster provides higher delays than Reliable in low traffic conditions. Comparatively, due to the nature of S-G mixes, Reliable's delay is not dependent on the traffic.

In addition, we have identified a number of key points of attack and weaknesses in mix software to which anonymity software designers need to pay particular attention. In addition to the areas of theoretical weakness that we have identified, we discovered a fatal flaw in the use of randomness in Reliable, which diminishes its ability to provide anonymity, independent of our findings with regard to the S-G mix protocol.

We can conclude from our analysis of the mixing algorithms used by these mix implementations that S-G mix variants such as the one used in Reliable are not suitable for use with systems that may have occurrences of low traffic on the network. While such S-G mixes may be an appropriate solution for systems with a steady input rate, they are not suited for systems with variable input traffic. Pool mixes such as Mixmaster should be preferred for systems with fluctuating traffic loads and relaxed latency constraints.

Chapter 7

Conclusions and Open Questions

I don't want to achieve immortality through my work.
I want to achieve it through not dying.
– Woody Allen

This thesis presents our main research results of the last four years. Section 7.1 presents a summary of the main conclusions and original contributions of this thesis. Section 7.2 presents the new challenges within the scope of this thesis.

7.1 Conclusions

This thesis dealt with the quantification of anonymity and its applications to anonymous communication networks. More in particular:

- Chapter 2 proposed a general method for the quantification of anonymity based on the information theoretic concept of entropy.
- Chapter 3 presented a framework for the analysis and design of anonymous communication nodes and cover traffic strategies.
- Chapter 4 analyzed and compared the anonymity properties of several node designs when subject to passive attacks.

- Chapter 5 analyzed and compared the robustness of several node proposals when subject to an active attack.

- Chapter 6 studied the anonymity provided by two anonymous email implementations.

Information Theoretic Anonymity Metrics. We proposed a general measurement model and derived two entropy-based information theoretic metrics, of which one is an original contribution of this dissertation. These metrics provide a general method to measure anonymity, to compare different systems, to evaluate the effectiveness of attacks on anonymity and to quantify gains and losses in anonymity which take into account the partial or statistical information obtained by an adversary.

The information theoretic metrics presented in this thesis are a versatile measurement tool. We have applied the metrics to evaluate the anonymity properties of both theoretical designs and practical systems.

The metrics proposed can be adapted to systems where anonymity can be defined in terms of unlinkability. Anonymous transactions are abstracted as IOIs (*Items Of Interest*); the anonymity of the subjects who are either originators or responders of a transaction can be computed applying the proposed formulas.

The metrics we presented provide relevant information on the anonymity of concrete subjects in concrete attack scenarios. We indicated that multiple measurements should be made in order to make a thorough evaluation of anonymity properties under all possible operative circumstances.

We indicated that the model is based on the probabilities adversaries assign to subjects; and that finding these probability distributions may not be easy in certain scenarios.

Taxonomy of Mixes and Dummy Traffic. We have introduced a taxonomy for mixes and cover traffic strategies. We identified the parameters that must be taken into account when designing, implementing or analyzing these components of anonymous communication networks.

We have proposed a model with which we can generalize classical pool mixes. This model is a powerful tool that gives us a new understanding of the batching strategies implemented by existing mixes. Also, new strategies that improve existing designs arise from the framework, which can choose an arbitrary function that leads to a certain anonymity/delay tradeoff. We have proposed as example the cumulative distribution function.

We have proposed the addition of randomness to the flushing algorithm, creating a variant of pool mixes named *binomial mixes*. As we have shown in this

thesis, this inexpensive change improves the robustness of mixes towards both passive and active attacks.

Passive Attacks on Mixes. We have applied the anonymity quantification tools and the framework for the analysis of mixes to the evaluation of the anonymity provided by pool mixes when they are subject of a passive attack. We have analyzed cover traffic strategies, and studied the certainty of passive adversaries when observing the information routed through anonymous communication nodes.

We have computed the anonymity provided by generalized mixes and continuous mixes, and provided compact and easy to implement formulas which allow the comparison of alternatives. We have indicated how to measure the anonymity when pool mixes insert dummy traffic in the pool or at the output.

We have indicated that the intuitive extension of the metric for taking cover traffic into account provides confusing results. We have clearly explained how it should be applied to obtain meaningful results and concluded that dummies generated by the mix contribute to recipient anonymity, but not to sender anonymity. Analogously, dummies discarded by the mix contribute to sender anonymity but not to recipient anonymity. We have found that inserting the dummies in the pool provides less anonymity and less latency than inserting them at the output.

Active Attacks on Mixes. We have studied the *blending* or $n-1$ attack. We have defined a set of parameters that indicate the effort required to deploy this active attack on deterministic pool mixes (timed and threshold), binomial pool mixes, continuous mixes and pool mixes with dummy traffic. For all these mixes, we have computed the remaining anonymity of a target message going through the attacked mix.

The parameters that define the effort of the adversary to deploy the blending attack are: average number of messages generated and delayed, total amount of time required to deploy the attack, and amount of time of observation required to start the attack.

We have indicated that the attack can be deployed faster against threshold mixes than against timed mixes. Consequently, the active attacker needs to delay less messages in the case of threshold mixes. The number of messages generated by the adversary to complete the attack is the same for timed and threshold pool mixes.

We have found two ways of deploying the blending attack on binomial mixes. The first method implies that the attacker must observe the mix long before the attack takes place, in order to accurately estimate the internal state of the mix. Alternatively, the adversary may start deploying the attack without an estimation of the internal state. We have analyzed the effort of the attacker and

the uncertainty of the results of the attack for the two possibilities. The remaining anonymity provided by binomial mixes in the absence of dummy traffic is very low.

We have shown that deploying the attack on continuous mixes requires a similar effort as the binomial or deterministic timed cases in terms of time required to complete the attack and number of messages delayed. However, in the continuous case the adversary does not need to generate messages in order to deploy the attack. The remaining anonymity provided by these mixes tends to zero as the adversary increments the time of attack.

We have found that making a deterministic pool mix insert the dummies at the output does not increase the effort required to the attacker to deploy the flushing phase of the attack with respect to the case of deterministic timed pool mixes without dummies. However, we saw that it does increase the remaining anonymity of the message up to $H = \log_2(d_k + 1)$, where d_k is the number of dummies inserted by the mix in the round in which the target is forwarded.

We have combined binomial mixes with dummy traffic inserted at the output, and found that the effort required to deploy the flushing phase of the attack increases with respect to the case of no dummies. Moreover, the remaining anonymity of the message is considerably higher than in the case of deterministic mixes with dummy traffic

We have seen that, regardless of whether the mix is deterministic or binomial, when dummies are inserted in the pool the remaining anonymity has similar (high) values to the case of binomial mix with dummies at the output. We have pointed out that the shortcoming of this configuration is that the effort of the attacker in terms of number of messages generated is lower than if the dummies are inserted at the output. We showed that the other effort parameters are similar to those of the binomial mix with dummy traffic inserted at the output.

Comparison of Two Practical Mixes. We have applied anonymity metrics to evaluate the anonymity properties of two working anonymous email implementations. First, we have analyzed the traffic pattern of a real traffic stream going through a working mix node and found that the traffic is not Poisson, as it was commonly assumed in the literature. We found a traffic pattern that is highly unpredictable. Therefore, we concluded recommending that no assumptions on the traffic should be made when designing a mix.

We have measured the anonymity of the pool mix scheme used in Mixmaster and Reliable by applying the information theoretic anonymity metrics presented at the beginning of this thesis. Our comparison of the two predominant mixing applications showed that Mixmaster provides superior anonymity, and is better suited for the anonymization of email messages than Reliable. We found that Mixmaster provides a minimum level of anonymity at all times; Reliable does

not. Reliable's anonymity drops to nearly zero when the traffic is very low. In high-traffic situations, Mixmaster provides a higher maximum anonymity than Reliable for the same stream of input. We have shown that Mixmaster provides higher average anonymity than Reliable for the same input and same average delay. Due to its nature as a pool mix, Mixmaster provides higher delays than Reliable in low traffic conditions.

We have identified a number of key points of attack and weakness in mix software to which anonymity software designers need to pay particular attention. In addition to the areas of theoretical weakness that we have identified, we discovered a fatal flaw in the use of randomness in Reliable, which diminishes its ability to provide anonymity, independent of our findings with regard to the S-G mix protocol.

We can conclude from our analysis of the mixing algorithms used by these mix implementations that S-G mix variants such as the one used in Reliable are not suitable for use with systems that may have occurrences of low traffic on the network. While such S-G mixes may be an appropriate solution for systems with a steady input rate, they are not suited for systems with variable input traffic. Pool mixes such as Mixmaster should be preferred for systems with fluctuating traffic loads and relaxed latency constraints.

7.2 Open Questions

From the results of our research, we can distill several new technical challenges that are particularly worth further research and effort in the future.

- Explore the possibility of using the min-entropy as metric (instead of Shannon's entropy). This metric may be useful in scenarios in which the priority is to avoid that any subject appears linked to an IOI with a relatively high probability.

- Study the impact of adding randomness to the threshold and/or timeout of a pool mix.

- Research optimization techniques for pool mix functions. For a given minimum quantity of anonymity required, study the pool mix functions that minimize the delay.

- More research is needed to find an efficient way of computing the probability distributions that lead to the anonymity provided by a mix network (as a whole) with dummy traffic.

- In order to find the most efficient cover traffic strategies, quantitative results for the anonymity achieved with different distributions of dummy traffic should be analyzed.
- Find how to measure anonymity if the dummy generation is dependent on the traffic flow. Study if these dependencies make the system stronger or more vulnerable to attacks.
- Analyze the effort and results of the blending attack on continuous mixes when cover traffic is generated by the mix.
- Design dummy policies that provide a strong protection against blending attacks.

Glossary

Binomial Mix: Pool mix that outputs a number of messages that follows a binomial distribution with respect to the number of messages contained in the pool (as opposed to *deterministic mixes*).

Chaumian Mix: See **Threshold Mix**.

Continuous Mix: Mix that delays messages going through it according to a delay distribution (typically exponential).

Cottrell Mix: See **Timed Dynamic Pool Mix**.

Deterministic Mix: Pool mix that outputs a number of messages that is deterministic with respect to the number of messages contained in the pool (as opposed to *binomial mixes*).

Dummy Traffic: Fake messages that are generated and transmitted by mixes in order to hide the traffic pattern of real messages and difficult traffic analysis.

Mix: Communication node that takes a number of input messages, and outputs them in such a way that it is hard to link an output to the corresponding input (or an input to the corresponding output) with certainty.

Pool Mix: Mix that has an internal memory (called *pool*) where certain messages are stored between rounds. Pool mixes can be represented in the GMM by a function that indicates the fraction of messages that are sent depending on the number of messages that have been collected. Pool mixes can be deterministic or binomial in their way of selecting messages. They may flush according to a threshold or a time condition.

Round: Pool mix cycle of collecting input messages, reordering and forwarding messages at the output.

Stop-and-Go Mix: See **Continuous Mix**.

Threshold Mix: Mix that collects N messages, reorders and forwards them. Threshold mixes can be seen as threshold pool mixes with a pool of size zero (also called **Chaumian Mix**). More generally, threshold mixes are those that forward messages when a certain amount of them has been accumulated by the mix (as opposed to *timed mixes*).

Threshold Pool Mix: Mix that forwards messages when it contains N messages. This mix sends a fixed fraction of the messages received while keeping the rest for future rounds.

Timed Dynamic Pool Mix: Pool mix that collects messages for a given amount of time T and forwards a fraction of them at the end of this period of time. This fraction is depends on the actual number of messages collected. Also called **Cottrell Mix**

Timed Mix: Mix that collects inputs for a given amount of time T and forwards all of them at the end of this period of time. Timed mixes can be seen as timed pool mixes with a pool of size zero. More generally, timed mixes are those that forward messages when a certain amount of time has past (as opposed to *threshold mixes*).

Timed Pool Mix: Mix that collects inputs for a given amount of time T and then forwards a number of them. This mix keeps a fixed number of messages in the pool and sends the rest.

Bibliography

[Abe98] Masayuki Abe. Universally verifiable MIX with verification work independent of the number of MIX servers. In *Advances in Cryptology, Proceedings of EUROCRYPT'98*, pages 437–447. Springer-Verlag, LNCS 1403, 1998.

[ADS03] Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the Economics of Anonymity. In *Proceedings of Financial Cryptography*, pages 84–102. Springer-Verlag, LNCS 2742, 2003.

[AKP03] Dakshi Agrawal, Dogan Kesdogan, and Stefan Penz. Probabilistic Treatment of MIXes to Hamper Traffic Analysis. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 16–27, 2003.

[And96] Ross Anderson. The eternity service. In *Proceedings of 1st International Conference on the Theory and Applications of Cryptology (Pragocrypt'96)*, pages 242–252. Czech Technical University Publishing House, 1996.

[BBC+94] Jean-Paul Boly, Antoon Bosselaers, Ronald Cramer, Rolf Michelsen, Stig Mjølsnes, Frank Muller, Torben Pedersen, Birgit Pfitzmann, Peter de Rooij, Berry Schoenmakers, Matthias Schunter, Luc Vallée, and Michael Waidner. The ESPRIT Project CAFE – High Security Digital Payment Systems. In *Proceedings of the Third Symposium on Research in Computer Security – ESORICS'94*, pages 217–230. Springer-Verlag, LNCS 875, 1994.

[BDD+05] Rainer Böhme, George Danezis, Claudia Díaz, Stefan Köpsell, and Andreas Pfitzmann. Mix cascades vs. peer-to-peer: Is one concept superior? In *Designing Privacy Enhancing Technologies, Proceedings of PET'04*, pages 243–255. Springer-Verlag, LNCS 3424, 2005.

[Ben01] Tonda Benes. The strong eternity service. In *Proceedings of Information Hiding Workshop (IH'01)*, pages 215–229. Springer-Verlag, LNCS 2137, 2001.

[BFK01] 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-Verlag, LNCS 2009, 2001.

[BFTS04] Ron Berman, Amos Fiat, and Amnon Ta-Shma. Provable unlinkability against traffic analysis. In *Proceedings of Financial Cryptography (FC'04)*, pages 266–280. Springer-Verlag, LNCS 3110, 2004.

[BGS01] Adam Back, Ian Goldberg, and Adam Shostack. Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., 2001.

[BHRPD] Zeyd Ben-Halim, Eric Raymond, Jürgen Pfeifer, and Thomas Dickey. Ncurses. `http://www.gnu.org/software/ncurses/ncurses.html`.

[BL02] Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. In *Designing Privacy Enhancing Technologies, Proceedings of PET'02*, pages 110–128. Springer-Verlag, LNCS 2482, 2002.

[BPS00] 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-Verlag, LNCS 2009, 2000.

[Bra93] Stefan Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology, Proceedings of CRYPTO'93*, pages 302–318. Springer-Verlag, LNCS 773, 1993.

[Bra95] Stefan Brands. Electronic cash on the internet. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 16–17, 1995.

[Bra99] Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates – Building in Privacy*. PhD thesis, Eindhoven Institute of Technology, 1999.

[Bro02] Zach Brown. Cebolla: Pragmatic ip anonymity. In *Ottawa Linux Symposium*, 2002.

[BSG00] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., 2000.

[Cam98] Jan Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998.

[CDG+03] Joris Claessens, Claudia Díaz, Caroline Goemans, Bart Preneel, Joos Vandewalle, and Jos Dumortier. Revocable anonymous access to the internet. *Journal of Internet Research: Electronic Networking Appliations and Policy*, 13(4):242–258, 2003.

[CDK01] Richard Clayton, George Danezis, and Markus G. Kuhn. Real world patterns of failure in anonymity systems. In *Proceedings of Information Hiding Workshop (IH'01)*, pages 230–244. Springer-Verlag, LNCS 2137, 2001.

[CE87] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Advances in Cryptology, Proceedings of CRYPTO'86*, pages 118–167. Springer-Verlag, LNCS 263, 1987.

[CEHL] Mark Cox, Ralf Engelschall, Stephen Henson, and Ben Laurie. The OpenSSL Project. `http://www.openssl.org/`.

[CFN88] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Advances in Cryptology, Proceedings of CRYPTO'88*, pages 319–327. Springer-Verlag, LNCS 403, 1988.

[CH02] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and Communications Security*, pages 21–30. ACM Press, 2002.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2):84–88, 1981.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[Cha90] David Chaum. Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. In *Advances in Cryptology, Proceedings of AUSCRYPT'90*, pages 246–264. Springer-Verlag, LNCS 453, 1990.

[Che95] Liqun Chen. Access with pseudonyms. In *Cryptography: Policy and Algorithms*, pages 232–243. Springer-Verlag, LNCS 1029, 1995.

[CL01] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Advances in Cryptology, Proceedings of EUROCRYPT'01*, pages 93–118. Springer-Verlag, LNCS 2045, 2001.

[CMS96] Jan Camenisch, Ueli Maurer, and Markus Stadler. Digital payment systems with passive anonymity-revoking trustees. In *Proceedings of the Fourth European Symposium on Research in Computer Security (ESORICS'96)*, pages 33–43. Springer-Verlag, LNCS 1146, 1996.

[Cor] Microsoft Corporation. Visual basic language reference, rnd function. *MSDN Library*. `http://http://msdn.microsoft.com/library/default.asp?url=/library/en-us%/vblr7/html/vafctrnd.asp`.

[Cot] Lance Cottrell. Mixmaster and remailer attacks. `http://www.obscura.com/~loki/remailer/remailer-essay.html`.

[CPS94] Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. An efficient electronic payment system protecting privacy. In *Proceedings of the Third European Symposium on Research in Computer Security (ESORICS'94)*, pages 207–215. Springer-Verlag, LNCS 875, 1994.

[CPS96] Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. An efficient fair payment system. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pages 88–94, 1996.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer-Verlag, LNCS 2009, 2000.

[DÓ5] Claudia Díaz. Accountable Anonymous Communication Infrastructure. Technical Report, ESAT-COSIC, K.U.Leuven, Belgium, 2005.

[DA04] George Danezis and Ross Anderson. The economics of censorship resistance. In *Workshop on Economics and Information Security (WEIS'04)*, 2004.

[Dai96] Wei Dai. Pipenet 1.1. Usenet post, `http://www.eskimo.com/~weidai/pipenet.txt`, 1996.

[Dan02] George Danezis. Forward secure mixes. In *Proceedings of 7th Nordic Workshop on Secure IT Systems*, pages 195–207, 2002.

[Dan03a] George Danezis. Mix-networks with restricted routes. In *Desingning Privacy Enhancing Technologies, Proceedings of PET'03*, pages 1–17. Springer-Verlag, LNCS 2760, 2003.

[Dan03b] George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426. Kluwer, 2003.

[Dan04] George Danezis. *Better Anonymous Communications*. PhD thesis, University of Cambridge, 2004.

[DC05] George Danezis and Jolyon Clulow. Compulsion resistant anonymous communications. In *Proceedings of Information Hiding Workshop (IH'05)*, pages 62–76. Springer-Verlag, LNCS 3727, 2005.

[DCSP02] Claudia Díaz, Joris Claessens, Stefaan Seys, and Bart Preneel. Information Theory and Anonymity. In *Proceedings of the 23rd Symposium on Information Theory in the Benelux*, pages 179–186, 2002.

[DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 2–15, 2003.

[DFHM01] Roger Dingledine, Michael Freedman, David Hopwood, and David Molnar. A Reputation System to Increase MIX-net Reliability. In *Proceedings of Information Hiding (IH'01)*, pages 126–141. Springer-Verlag, LNCS 2137, 2001.

[DFM00] Roger Dingledine, Michael Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *Designing Privacy Enhancing Technologies*, pages 67–95. Springer-Verlag, LNCS 2009, 2000.

[DFTY97] George Davida, Yair Frankel, Yiannis Tsiounis, and Moti Yung. Anonymity Control in E-Cash Systems. In *Proceedings of Financial Cryptography '97*, pages 1–16. Springer-Verlag, LNCS 1318, 1997.

[DG96] Peter Deutsch and Jean-Loup Gailly. ZLIB Compressed Data Format Specification version 3.3. Request for Comments: 1950, 1996.

[Din] Roger Dingledine. Freehaven anonymity bibliography. `http://www.freehaven.net/anonbib/`.

[DMS03] Roger Dingledine, Nick Mathewson, and Paul Syverson. Reputation in P2P Anonymity Systems. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320. USENIX, 2004.

[Dou02] John Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS'02)*, pages 251–260, 2002.

[DP04a] Claudia Díaz and Bart Preneel. Anonymous communication. Electronic publication at WHOLES - A Multiple View of Individual Privacy in a Networked World, 2004.

[DP04b] Claudia Díaz and Bart Preneel. Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In *Proceedings of 6th Information Hiding Workshop (IH'04)*, pages 309–325. Springer, LNCS 3200, 2004.

[DP04c] Claudia Díaz and Bart Preneel. Taxonomy of mixes and dummy traffic. In *Information Security Management, Education and Privacy (SEC'04)*, volume 3, pages 215–230. Kluwer, 2004.

[DS03a] George Danezis and Len Sassaman. Heartbeat traffic to counter (n-1) attacks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, pages 89–93. ACM Press, 2003.

[DS03b] Claudia Díaz and Andrei Serjantov. Generalising mixes. In *Designing Privacy Enhancing Technologies, Proceedings of PET'03*, pages 18–31. Springer-Verlag, LNCS 2760, 2003.

[DS04] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Proceedings of 6th Information Hiding Workshop (IH'04)*, pages 293–308. Springer-Verlag, LNCS 3200, 2004.

[DSCP03] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Designing Privacy Enhancing Technologies, Proceedings of PET'02*, pages 54–68. Springer-Verlag, LNCS 2482, 2003.

[DSD04] Claudia Díaz, Len Sassaman, and Evelyne Dewitte. Comparison between two practical mix designs. In *Proceedings of 9th European Symposiumon Research in Computer Security (ESORICS'04)*, pages 141–159. Springer-Verlag, LNCS 3193, 2004.

[Fel50] William Feller. *An introduction to probability theory and its applications.* Wiley, 1950.

[FM02] Michael Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, pages 193–206. ACM Press, 2002.

[FO96] Eiichiro Fujisaki and Tatsuaki Okamoto. Practical Escrow Cash System. In *Proceedings of the 4th Security Protocols Workshop*, pages 33–48. Springer-Verlag, LNCS 1189, 1996.

[FR01] Eric Friedman and Paul Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Advances in Cryptology, Proceedings of CRYPTO'01*, pages 368–387. Springer-Verlag, LNCS 2139, 2001.

[GHPvR05] Flavio Garcia, Ichiro Hasuo, Wolter Pieters, and Peter van Rossum. Provable anonymity. In *Proceedings of the 3rd ACM Workshop on Formal Methods in Security Engineering (FMSE05)*, pages 63–72. ACM Press, 2005.

[GJ04] Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Advances in Cryptology, Proceedings of EUROCRYPT'04*, pages 456–473. Springer-Verlag, LNCS 3027, 2004.

[Gol00] Ian Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, UC Berkeley, 2000.

[Gol02] Ian Goldberg. Privacy-enhancing technologies for the Internet, II: Five years later. In *Designing Privacy Enhancing Technologies, Proceedings of PET'02*, pages 1–12. Springer-Verlag, LNCS 2482, 2002.

[Goo] Google. `http://www.google.com/`.

[Gro03] Christian Grothoff. An Excess-Based Economic Model for Resource Allocation in Peer-to-Peer Networks. *Wirtschaftsinformatik*, 45(3):285–292, 2003.

[GRPS03] Sharad Goel, Mark Robson, Milo Polte, and Emin Gun Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical Report 2003-1890, Cornell University, 2003.

[GRS96] David Goldschlag, Michael Reed, and Paul Syverson. Hiding Routing Information. In *Proceedings of Information Hiding (IH'96)*, pages 137–150. Springer-Verlag, LNCS 1174, 1996.

[GT96] Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium (NDSS'96)*, pages 2–16. IEEE, 1996.

[GW96] Ian Goldberg and David Wagner. Randomness and the Netscape browser. *Dr. Dobb's Journal*, January 1996.

[GW98] Ian Goldberg and David Wagner. TAZ servers and the rewebber network: Enabling anonymous publishing on the world wide web. *First Monday*, 3(4), 1998.

[GWB97] Ian Goldberg, David Wagner, and Eric Brewer. Privacy-enhancing technologies for the internet. In *Proceedings of the 42nd IEEE Spring COMPCON*, pages 103–109. IEEE Computer Society Press, 1997.

[Haz] Philip Hazel. Perl compatible regular expressions. `http://www.pcre.org/`.

[Hin02] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Designing Privacy Enhancing Technologies, Proceedings of PET'02*, pages 171–178. Springer-Verlag, LNCS 2482, 2002.

[Jak98] Markus Jakobsson. A practical mix. In *Advances in Cryptology, Proceedings of EUROCRYPT'98*, pages 448–461. Springer-Verlag, LNCS 1403, 1998.

[Jak99] Markus Jakobsson. Flash Mixing. In *Proceedings of Principles of Distributed Computing (PODC'99)*, pages 83–89. ACM Press, 1999.

[JAP] JAP Anonymity & Privacy. `http://anon.inf.tu-dresden.de/`.

[JC02] Els Van Herreweghen Jan Camenisch. Design and implementation of the idemix anonymous credential system. In *Research Report RZ 3419, IBM Research Division*, 2002.

[JJ01] Markus Jakobsson and Ari Juels. An optimally robust hybrid mix network (extended abstract). In *Proceedings of Principles of Distributed Computing (PODC'01)*, pages 284–292. ACM Press, 2001.

[JJR02] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353. USENIX Association, 2002.

[JM98] Markus Jakobsson and David M'Raïhi. Mix-based Electronic Payments. In *Proceedings of the 5th Annual International Workshop on Selected Areas in Cryptography (SAC'98)*, pages 157–173. Springer-Verlag, LNCS 1556, 1998.

[JMP+98] Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. Real-Time MIXes: A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications*, 16(4):495–509, 1998.

[KAP02] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In *Proceedings of Information Hiding Workshop (IH 2002)*, pages 53–69. Springer-Verlag, LNCS 2578, 2002.

[KEB98] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*, pages 83–98. Springer-Verlag, LNCS 1525, 1998.

[KP04] Dogan Kesdogan and Lexi Pimenidis. The hitting set attack on anonymity protocols. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, pages 326–339. Springer-Verlag, LNCS 3200, 2004.

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym Systems. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, pages 184–199. Springer-Verlag, LNCS 1758, 1999.

[LRWW04] Brian Levine, Michael Reiter, Chenxi Wang, and Matthew Wright. Timing attacks in low-latency mix-based systems. In *Proceedings of Financial Cryptography (FC '04)*, pages 251–265. Springer-Verlag, LNCS 3110, 2004.

[MÖ2] Ulf Möller. Personal communication. Private email to Len Sassaman, August 2002.

[MK98] David Mazières and M. Frans Kaashoek. The Design, Implementation and Operation of an Email Pseudonym Server. In *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS 1998)*, pages 27–36. ACM Press, 1998.

[MK00] Masashi Mitomo and Kaoru Kurosawa. Attack for Flash MIX. In *Proceedings of ASIACRYPT 2000*, pages 192–204. Springer-Verlag, LNCS 1976, 2000.

[Nef01] Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 116–125. ACM Press, 2001.

[OA00] Miyaku Ohkubo and Masayuki Abe. A Length-Invariant Hybrid MIX. In *Proceedings of ASIACRYPT 2000*, pages 178–191. Springer-Verlag, LNCS 1976, 2000.

[PC95] European Parliament and European Council. Directive 95/46/ec. *Official Journal of the European Communities*, No L 281:31, 1995.

[Pfi94] Birgit Pfitzmann. Breaking Efficient Anonymous Channel. In *Advances in Cryptology, Proceedings of EUROCRYPT'94*, pages 332–340. Springer-Verlag, LNCS 950, 1994.

[PH04] Andreas Pfitzmann and Marit Hansen. Anonymity, unobservability, and pseudonymity: A proposal for terminology. Draft, v0.21, September 2004.

[PP90] Birgit Pfitzmann and Andreas Pfitzmann. How to break the direct RSA-implementation of MIXes. In *Advances in Cryptology, Proceedings of EUROCRYPT 1989*, pages 373–381. Springer-Verlag, LNCS 434, 1990.

[PPW91] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, 1991.

[Pro] KeyLog Pro. `http://www.keylogpro.com/`.

[PS00] Birgit Pfitzmann and Ahmad-Reza Sadeghi. Self-Escrowed Cash against User Blackmailing. In *Proceedings of Financial Cryptography 2000*, pages 42–52. Springer-Verlag, LNCS 1962, 2000.

[PSG00] Michael Reed Paul Syverson and David Goldschlag. Onion routing access configurations. In *DARPA Information Survivability and Exposition (DISCEX 2000)*, IEEE CS Press, 2000.

[Ray00] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *Designing Privacy Enhancing Technologies*, pages 10–29. Springer-Verlag, LNCS 2009, 2000.

[RP04] Marc Rennhard and Bernhard Plattner. Practical anonymity for the masses with morphmix. In *Proceedings of the Financial Cryptography Conference (FC 2004)*, pages 233–250. Springer-Verlag, LNCS 3110, 2004.

[RR98] Michael Reiter and Aviel Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):66–92, 1998.

[RSG98] Michael Reed, Paul Syverson, and David Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, 1998.

[Sas02] Len Sassaman. The promise of privacy. Invited talk, LISA XVI, November 2002.

[SBS02] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: A protocol for scalable anonymous communication. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, 2002.

[SCM05] Len Sassaman, Bram Cohen, and Nick Mathewson. The pynchon gate: A secure method of pseudonymous mail retrieval. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2005)*. ACM, 2005.

[SD02] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Designing Privacy Enhancing Technologies, Proceedings of PET'02*, pages 41–53. Springer-Verlag, LNCS 2482, 2002.

[SDS02] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, 2002.

[Ser04] Andrei Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, 2004.

[Sha48] Claude Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423:623–656, 1948.

[Shmng] Vitaly Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, forthcoming.

[SM05] Andrei Serjantov and Steven J. Murdoch. Message splitting against the partial adversary. In *Designing Privacy Enhancing Technologies, Proceedings of PET'05*. Springer-Verlag, LNCS (forthcoming), 2005.

[SS99] Paul Syverson and Stuart Stubblebine. Group principals and the formalization of anonymity. In *Proceedings of the World Congress on Formal Methods (1)*, pages 814–833, 1999.

[STRL00] 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-Verlag, LNCS 2009, 2000.

[STS99a] Tomas Sander and Amnon Ta-Shma. Auditable, Anonymous Electronic Cash. In *Advances in Cryptology, Proceedings of CRYPTO'99*, pages 555–572. Springer-Verlag, LNCS 1666, 1999.

[STS99b] Tomas Sander and Amnon Ta-Shma. Flow Control: A New Approach For Anonymity Control in Electronic Cash Systems. In *Proceedings of Financial Cryptography '99*, pages 46–61. Springer-Verlag, LNCS 1648, 1999.

[Tha03] Rodney Thayer. SlimJim: shared library shimming for password harvesting. Presentation, ToorCon 2003, September 2003.

[Tho84] Ken Thompson. Reflections on trusting trust. *Communications of the ACM*, 27(8), 1984.

[UMS03] Peter Palfrader Ulf Moller, Lance Cottrel and Len Sassaman. Mixmaster protocol - version 2. `http://www.abditum.com/mixmaster-spec.txt`, 2003.

[WALS03] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 28–41, 2003.

[WM01] Marc Waldman and David Mazières. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 126–135. ACM Press, 2001.

[WP90] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco: unconditional sender and recipient untraceability with computationally secure servicability. In *Advances in Cryptology, Proceedings of EUROCRYPT'89*, page p. 690. Springer-Verlag, LNCS 434, 1990.

[WRC00] Marc Waldman, Aviel Rubin, and Lorrie Cranor. Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, 2000.

[Zer] Zero-Knowledge Systems. Freedom Network. `http://www.zeroknowledge.com/`.

List of Publications

International Journals

1. Joris Claessens, Claudia Díaz, Caroline Goemans, Bart Preneel, Joos Vandewalle and Jos Dumortier, "Revocable anonymous access to the Internet", Journal of Internet Research: Electronic Networking Applications and Policy, Vol. 13, No 4, pp. 242–258, 2003.

Lecture Notes in Computer Science

1. Rainer Böhme, George Danezis, Claudia Díaz, Stefan Köpsell and Andreas Pfitzman, "Mix Cascades vs. Peer-to-Peer: Is One Concept Superior?", Designing Privacy Enhancing Technologies, D. Martin and A. Serjantov (Eds), LNCS 3424, pp. 243–255, 2005.

2. Claudia Díaz, Len Sassaman and Evelyne Dewitte, "Comparison between two practical mix designs", Computer Security (ESORICS), Samarati *et al.* (Eds), LNCS 3193, pp. 141–159, 2004.

3. Claudia Díaz and Bart Preneel, "Reasoning about the Anonymity Provided by Pool Mixes that Generate Dummy Traffic", Information Hiding, J. Fridrich (Ed.), LNCS 3200, pp. 309–325, 2004.

4. Claudia Díaz and Andrei Serjantov, "Generalizing Mixes", Designing Privacy Enhancing Technologies, R. Dingledine (Ed.), LNCS 2760, pp. 18–31, 2003.

5. Claudia Díaz, Stefaan Seys, Joris Claessens and Bart Preneel, "Towards measuring anonymity", Designing Privacy Enhancing Technologies, R. Dingledine and P. Syverson (Eds.), LNCS 2482, pp. 54–68, 2002.

International Conferences/Workshops

1. Claudia Díaz and Bart Preneel, "Taxonomy of Mixes and Dummy Traffic", Information Security Management, Education and Privacy, Y. Deswarte, F. Cuppens, S. Jajodia and L. Wang (Eds), Vol. 3 pp. 215–230, 2004.

2. Claudia Díaz and Bart Preneel, "Anonymous communication", WHOLES - A Multiple View of Individual Privacy in a Networked World. Internet publication, 7 p., 2004.

Journals (national level)

1. Claudia Díaz, Joris Claessens and Bart Preneel, "APES: Anonymity and Privacy in Electronic Services", Datenschutz und Datensicherheit, Vol. 27, No 3, pp. 143–145, 2003.

National Conferences/Workshops

1. Claudia Díaz, Joris Claessens, Stefaan Seys and Bart Preneel, "Information Theory and anonymity", Werkgemeenschap voor Informatie en Communicatietheorie, B. Macq and J.-J. Quisquater (Eds), pp. 179–186, 2002.

Claudia Díaz was born on November 11, 1974 in Madrid, Spain. She received the degree of Telecommunications Engineer (Ingeniero Técnico Superior de Telecomunicaciones) from the University of Vigo, Spain, in October 2000. Her Master's thesis (Proyecto Fin de Carrera) was partially done as Erasmus student at the research group COSIC (Computer Security and Industrial Cryptography) at the Department of Electrical Engineering (ESAT) of the K.U.Leuven, and dealt with the design and implementation of an environment for certification and authentication. In November 2000 she joined COSIC first as predoctoral student. She started her doctoral research in November 2001.