This is the Project Title

Your Name

Master of Science
Cyber Security, Privacy and Trust
School of Informatics
University of Edinburgh
2022

Abstract

This skeleton demonstrates how to use the `infthesis` style for MSc dissertations in the School of Informatics. It also emphasises the page limit and associated style restrictions for Informatics dissertations with course code `INFR11077`. If your degree has a different project course code, then it is likely to have different formatting rules. The file `skeleton.tex` generates this document and should be used as a starting point for your thesis. Replace this abstract text with a concise summary of your report.

Research Ethics Approval

Instructions: *Agree with your supervisor which statement you need to include. Then delete the statement that you are not using, and the instructions in italics.*
Either complete and include this statement:
This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: ???
Date when approval was obtained: YYYY-MM-DD
[If the project required human participants, edit as appropriate, otherwise delete:]
The participants' information sheet and a consent form are included in the appendix.
Or include this statement:
This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Your Name*)

Acknowledgements

Any acknowledgements go here.

Table of Contents

Chapter 1

Introduction

The predawn of the 21st century was marked by unprecedented growth in the information technology field, especially after the commercialisation of the World Wide Web. Every household got plugged into the internet, and every person could navigate through web pages, shop online, send emails and communicate with others across the globe. During that era, the internet was a safe place predominantly used by academics and hobbyists. However, once everyone became a potential user, crime shifted from the analogue to the digital world.

Many hacker groups have different reasons for attacking private communication channels, targeting personal information, such as passwords, credit card numbers, and government emails. Hackers can be (a) cybercriminals who want to exploit users' data for profit, (b) governments targeting private messages for national interest or (c) security researchers challenging and enhancing systems.

Nowadays, we overcome most of the vulnerabilities related to plain-data transfers over the wire/air using the notion of cryptography. Even though this mechanism seems to provide adequate security, it fails to encapsulate metadata concealment. Metadata such as the message size, the communication duration, the message origin and destination leak information about an online encrypted, ”secure” communication. A global adversary passively eavesdropping on every network node can statistically observe a message's sender and receiver with a certain probability. Characteristically, the former National Security Agency (NSA) and Central Intelligence Agency (CIA) director General Michael Hayden said that ”we kill people based on metadata”. Hence, researchers have been steering toward new means of communication that respect users' privacy and improve anonymity.

Mix Networks (Mixnets) provide a solution to the metadata anonymity issue by routing equal-sized messages through a chain of nodes (Mixes). Mixes shuffle messages using different techniques to obliterate any link between the sender and the receiver.

The Mixnets pioneer David Chaum proposed Cascade networks as a tool for a hard-to-trace electronic mail exchange system back in 1981. Nevertheless, his idea did not succeed due to (a) the high computational requirements of such a system, (b) the significant network latency and (c) the lack of quality implementations[]. Soon, researchers realised the potential of Mixnets and their applications to many domains. Such applications include remailers[], instant messaging apps[], or even more recently, blockchain transaction routing apps that provide anonymity[]. In addition, researchers have developed two simulators[] to empirically evaluate Mix Networks' performance and anonymity. When designing a Mixnet, we can have several design configurations. With the term configurations, we refer to different mixing techniques and topologies, which we will explain later in this document.

Within the context of this project, we are interested in experimenting with the existing Mix Network simulators, trying to answer the following questions:

1. Which simulator is better to adopt for future research and development?
2. How do simulators compare in terms of performance (time required to run a simulation and memory used)?
3. What features do simulators implement?
4. How do simulators score in terms of their realisation?

Additionally, according to published related work[], both simulators support static scenarios assuming a network is operational as initialised. This fact raises the following questions:

1. What happens in a situation where Mixes fail arbitrarily and the network is imbalanced?
2. If Mixes fail, can we achieve the same or worse anonymity?
3. How is the network latency affected?

The two hypotheses of the current dissertation have been motivated by these questions. The existing literature does not elaborate on these matters, yielding a research opportunity. In more detail, our first hypothesis assumes that both simulators perform equally and can contribute to future research. Our second hypothesis focuses on the behaviour of imbalanced stratified Mixnets, assuming that anonymity downgrades since some Mixes receive less traffic. To test our hypotheses, we run a series of experiments and propose a framework that enables us to compare both simulators to a certain degree.

The contribution of this project is twofold. We initially assess both simulators quantitatively and qualitatively to find their potential. Secondly, we extend one of the simulators to support a Stratified network where nodes crash arbitrarily, during the transmission rounds.

The conducted analysis suggests that the MiXiM simulator offers a plethora of features; however, the code seems to be buggy and not entirely functional. On the other hand, Simulator[] (Ania's project has not an official name and is called "Simulator") has fewer features but works appropriately, has cleaner code, and therefore can be trusted for further development. Nonetheless, it is worth mentioning that Simulator is a high memory demand software. Remarkably, increasing the number of users in a simulation requires lots of memory, while the simulation runtime sharply increases.

In addition, we observe that a dynamically imbalanced Stratified Mixnet is not compromised in terms of anonymity since all mixes will receive the same amount of traffic in the long term. Nevertheless, there is a notable difference when the network is statically imbalanced. A Stratified Mixnet with a constantly different number of nodes on each layer achieves various levels of anonymity. As noticed, the number of mixes in the middle layers is not significantly affecting anonymity, while the opposite is true for the first and last layers. Furthermore, assuming the same traffic while increasing the number of mixes per layer indicates a decline in anonymity. Also, we find out that the number of users sending and receiving messages over the network significantly impacts anonymity. In other words, anonymity sharply increases when a network facilitates thousands of users. Finally, the current realisation of Stratified networks seems not to affect the average end-to-end transmission latency.

The following chapters are organised as follows. Chapter 2 briefly introduces background knowledge, including Mixnets fundamentals and the anonymity metric used. Chapter 3 focuses on comparing the provided simulators, qualitatively and quantitatively. Chapter 4 covers the replication and comment of existing literature using the simulators. Chapter 5 presents our analysis and experiment conducted to bring conclusiveness to our research questions. Finally, chapter 6 summarises the key points and lessons learned from this project and discusses future directions on this topic.

Chapter 2

Background

This chapter focuses on delivering essential information and background knowledge needed to understand Mix Networks. Next, we describe the structure of a Mixnet and how it operates, as well as various system arrangements regarding network topologies and mixing techniques. Also, we introduce and define the metric of entropy, which we will use to measure anonymity in subsequent chapters.

2.1 Mix Networks Fundamentals

Mix Network is a communication system which enables us to exchange messages from one end to another, hiding metadata. In more detail, metadata includes but is not limited to geographical location, message sender and receiver, transmission time and frequency, size of the message and the message sequence. Mixnets remove time and sequence-related correlation factors between a message sender and receiver as they route same-size messages through a chain of nodes called Mixes. Mixes can be arranged into specific formations called topologies, defining how they are interconnected. Whenever a Mix receives a message m, it is permuted within a set of other messages M. The messages' permutation is called mixing and can be conducted using various techniques. Mixing aims to hinder an adversary from linking input and output messages of a Mix. Still, a global adversary can observe a sender initiating a message transmission and a receiver acquiring a message. For this reason, Mixnets use decoy/cover traffic to confuse an adversary, determining a real from a fake message.

Furthermore, each topology and mixing technique has advantages and disadvantages, yielding scalability and performance versus anonymity tradeoffs. In the following sections (... ..), we thoroughly analyse available topologies and mixing techniques, explaining their differences.

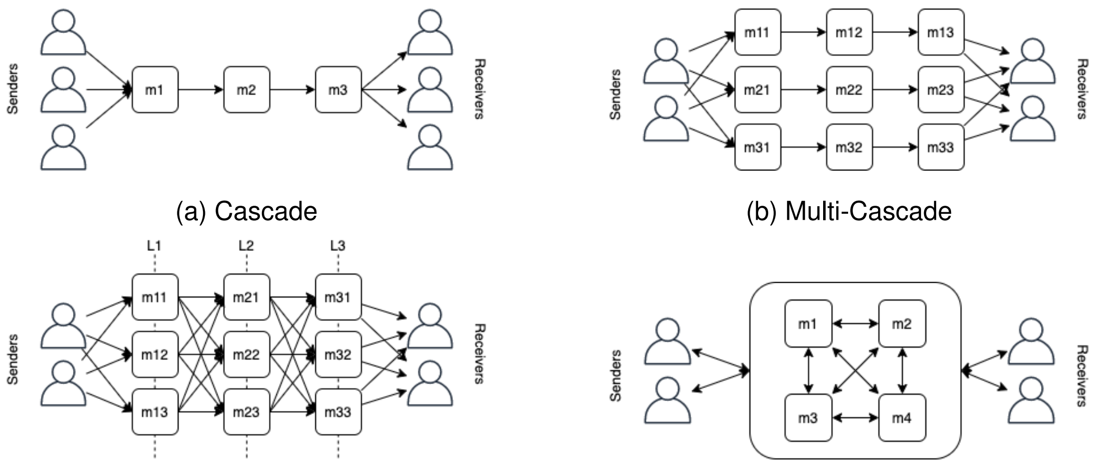
2.2 Mix Network Topologies

- **Cascade:** This topology defines a simplified version of Mixnets, where all messages pass through a predetermined chain of nodes[6]. However, Cascade topology has many limitations in terms of scalability. Notably, its maximum throughput is caped to the throughput of a single Mix; thus, network latency is high as well. Besides that, If we have any malicious node on the chain, the whole system can be compromised or fail to deliver any messages, turning the Mix into a single point of failure. We can see what a cascade topology looks like in Figure..

- **Multi-Cascade:** This topology removes the Cascade topology limitations of scalability, and single point of failure, by deploying multiple parallel Cascades. Figure .... depicts this topology. As we can see, the main advantage of this formation is the greater availability and scalability. The user can select a different Cascade to transmit a message. Also, having multiple Cascades allows the network to process more messages simultaneously. However, if a Mix on a Cascade is malicious, we will get less anonymity since there is no way of routing traffic between the different Cascades.

- **Stratified:** This topology comprises a set of layers. On each layer, we can have a fixed number of mixes. Every Mix that belongs to layer l is bonded to every node that belongs to layers $l-1$ and $l+1$[8]. Variations of this topology are (a) fully connected nodes as described above and (b) semi-connected nodes where each node in layer l communicates with only a set of nodes on layers $l-1$ and $l+1$[7]. Stratified topology is flexible regarding scalability and latency because we can handle more traffic by adding more mixes in each layer. Also, having many mixes in each layer makes the network crash and fault tolerant. Anyhow, we must note that if a stratified topology gets too large and the traffic for some reason is reduced, anonymity is also reduced. This happens because messages split to more nodes, and therefore we have less mixing. We can overcome this issue by generating decoy traffic, but we must recognise that an extra computational cost accompanies this. Figure .... illustrates a fully connected stratified topology.

- **Mesh:** In this topology, Mixes are loosely arranged. Specifically, there is a link between every Mix in the network. A mixing route can start and finish at any node. Nevertheless, the muddled paths a message can take on this topology make it hard for researchers to measure anonymity. Hence, it is not widely realised and used. Figure 1.d depicts a mesh topology.



(a) Cascade

(b) Multi-Cascade

(c) Stratified

(d) Mesh

Figure 2.1: Mixnets Topologies

2.3 Mixing Techniques

- **Time-Based:** The time-based mixing technique aims to collect incoming messages and permute and forward them every t seconds, where t is a system-defined variable. Figure ... illustrates how time-based Mixes work. The advantage of this mixing technique is that we can control the message transfer delay. Nonetheless, on the downside, we do not know and govern how many messages we mix. This is because a different number of messages may be received in separate rounds. As a result, the number of messages shuffled each time is not the same; therefore, the anonymity set can sometimes be large while sometimes smaller. A small set of messages infers reduced anonymity. To keep anonymity above a baseline, we must incorporate complementary cover traffic, which comes with an extra computational expense.

- **Threshold-Based:** Similar to time-based mixes, a threshold Mix buffers messages into a queue until a predefined threshold is reached. Then, we shuffle and forward messages to the next hop[]. In contrast with timed Mixes, threshold Mixes allow us to control the anonymity set of messages being processed. Nonetheless, this technique has some disadvantages as well. Primarily, the problem arises when the network does not have much traffic and the receiving buffer does not reach its limit shortly. As a result, message delivery will be delayed significantly, which is not ideal for the user's experience. The solution to this problem is to integrate cover traffic in order to fill a Mix with messages more frequently. As mentioned before, this comes with a computational overhead.

- **Pool-Mix:** This hybrid mixing technique[25] derives features from timed, and threshold mixing approaches. We release a fraction of the messages every t seconds if a threshold is reached. The bright side of this realisation is that a message leaving the Mix can be either a new message which just entered the pool or an old one which was not picked during the last emission. There is no way of knowing which one was picked. Consequently, anonymity is improved. At the same time, one could assume this feature is a bug. In other words, a message can stack in the pool for quite some time before arriving at its destination. We can observe that this yields unexpected delays. Besides that, a pool Mix requires many messages to work optimally. Therefore, as mentioned earlier, this solution might need to include cover traffic to increase the network's capacity.

- **Continues:** This mixing technique depends on a delay given by the message sender[26]. The main advantage of this approach is that messages are not shuffled together, and therefore, there is no statistical inference. Each message is treated individually, and the dispatch timestamp is not affected by the arrival of any other message.

2.4 Measuring Anonymity

As many scientists stated in the past, ”Entropy is the natural order of the universe”, meaning natural chaos. One way to measure anonymity in Mixnets is entropy. In other words, we are trying to measure the chaos between message transmissions. Hence, we use the Shannon entropy formula (the result is in bits) to achieve that. Equation (2.1) describes that formula. High entropy values imply better anonymity than lower values. There is no way of identifying acceptable entropy levels since this depends on the network size and adversary capabilities. In the following formula, we denote as $p(x)$ the probability that an attacker can assign to a user being the message sender. For instance, if we have an anonymity set of 1000 indistinguishable messages within a Mix buffer, we have 9.96 bits of entropy. ($Entropy = -\sum_{i=1}^{1000} p(\frac{1}{1000}) \log_2 p(\frac{1}{1000}) = 9.96bits$).

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{2.1}$$

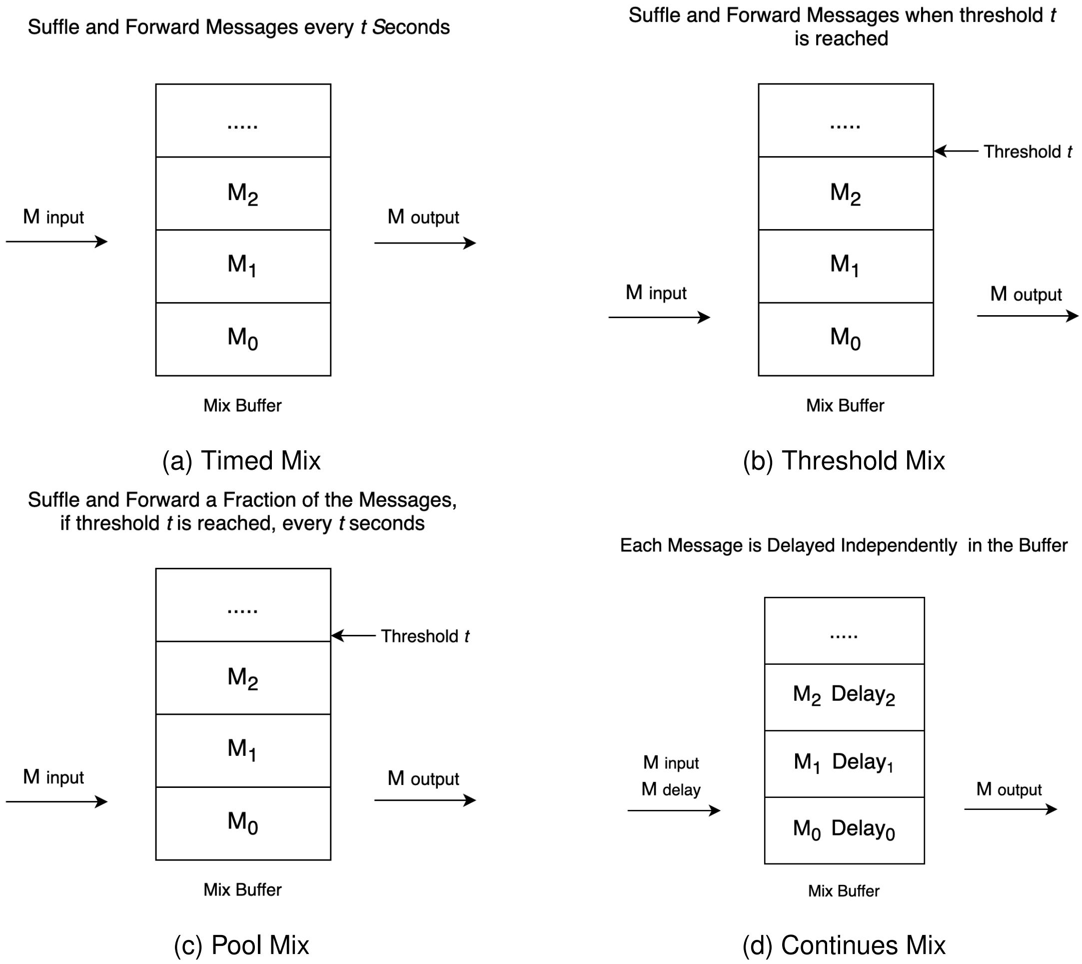


Figure 2.2: Mixnets Mixing Techniques

Chapter 3

A Deep Dive Into The Simulators

This chapter describes MiXiM and Simulator, criticising both projects qualitatively and quantitatively. Next, we provide information about the simulators' realisation, development decisions, execution workflow and supported features. We also conduct a baseline experiment, measuring their needs for computational resources. Finally, we aggregate this information on a competitive grid to decide which software is potentially the best to adopt for future research and development.

3.1 Simulator

Piotrowska's work[] focuses on creating a simulator for evaluating the anonymity, latency, bandwidth overhead and scalability of Mix Networks over different design configurations. Her study mainly used the Simulator to compare existing projects - Elixxir, HOPR and Nym - deployed on Mix Network infrastructure with different layouts.

The developed software realises many features that capture real-world use cases of Mix Networks, but it is not limited to it. Even though the simulator can handle peer-to-peer (p2p) network simulations, we are not interested in such network formations within the context of this project. Next, we will look deeper into the Simulator's intrinsics, reporting on its core functionality, available features and input parameters. Additionally, we will run the simulator against a baseline configuration file using profiling tools[] and capture useful information about the simulation execution. This type of information will subsequently enable us to comment on simulation bottlenecks.

After analysing Piotrowska's simulator, we understood the code intrinsics and captured its approach when simulating a network. The project is well-designed, with the main simulation entities abstracted into classes. In particular, there are abstractions for the following entities: (a) Network, (b) Mix Node, (c) Client, (d) Message and (e) Packet. Even though it appears to exist a provision for splitting messages into packages, it seems that this part of the code is incomplete, and consequently, all messages are of one packet size.

Furthermore, this simulator allows users to run experiments given many options. In more detail, there is support for many topologies, such as (a) Cascade, (b) Multi-Cascade, (c) Stratified and (d) P2P. Besides, there is an option for two mixing techniques (a) batch and reorder - threshold - and (b) poisson mixing -continues. The current implementation has default-enabled cover traffic generation for both clients and Mixes. Finally, the Simulator outputs the anonymity metrics of (a) entropy and (b) unlinkability.

3.1.1 Execution Workflow

The workflow of executing a standard simulation scenario is as follows:

1. Read user-generated configuration file.
2. Initialize global/environment variables.
3. Initialize Loggers.
4. Create and initialize a Network object comprising a set of Clients and Mix Nodes.
5. From the Clients' set, select two message senders and one recipient randomly.
6. All clients begin generating cover traffic, while the initially elected senders will also generate real traffic. The message routing is selected randomly on every packet hop.
7. The above process is simulated for three phases, burnin, execution, and cool down. During the burnin step, clients and Mixes generate only cover traffic, and no logging is performed. On the other hand, logging is enabled during the execution stage. During that phase, clients and Mixes send both real and cover traffic. We log entropy for every real message on every hop. Finally, we stop sending real messages during the cool-down phase while we continue sending dummy messages.
8. Ultimately, the Simulator presents results to the user after inspecting the execution logs.

3.1.2 Input Parameters and Configuration File

To begin with, the Simulator takes as input some command-line arguments defining the simulation mode and necessary directories, where we can find the configurations file, and directories where the output results will be dumped. There follows a list of the accepted command-line arguments and their description:

`-mode` **(required)** *(string)*: This describes the mode in which we run the simulator. The accepted modes are *test*, *test_diff*, *transcript*, *synthetic traces*, and *anon*; however, only the *test* mode is entirely realised. We have no clue about the functionality or purpose of the other modes.

`-exp_dir` **(required)** *(string)*: This argument declares the directory's path, where the Simulator will dump any experiment logging files.

`-config_file` **(required)** *(string)*: This is the path to the configuration file, describing the simulation settings.

The following command-line arguments are declared but never actually used in the Simulator. We assume that a future continuation of this project will frame their purpose. These arguments are `-test`, `-datadir`, `-hour`, `-12hour`, `-minute`, `-day`.

Moving to the configuration file, we can observe nine sections describing different simulation aspects. We present a sample configuration file in Figure ...

- **Section 1** - *Experiment Id*: In this section, we provide a simple label to help keep different simulations in order.

- **Section 2** - *Logging*: This section allows us to enable or disable logging. The attribute `dir` extends the logging directory path specified in the command-line arguments. The attributes `client_log` and `mix_log` shown in the configurations sample are never used throughout the simulation.

- **Section 3** - *Simulation Phases*: This section is related to simulation phases - burnin, execution and cool down. Mainly we can define the duration of each stage.

- **Section 4** - *Network Topologies*: This part of the file describes the supported network topologies of the simulator - Cascade, Stratified, Multi-Cascade and P2P - and their properties, such as the number of layers and layer size for stratified topologies or length and wideness for Cascade/Multi-Cascade topologies.

- **Section 5** - *Packets*: This section defines the packet size (feature not realised).
- **Section 6** - *Messages*: This section defines the message size (feature not realised, all messages are of length one packet).
- **Section 7** - *Mixing Configurations*: Here, we can find Mixes settings like the average delay before sending a packet (continues mixing) and the batching of messages according to specific batch size (threshold mixing).
- **Section 8** - *Clients Information*: Here, we can specify information about clients, such as the number of participating clients, how often the client adds a real message to the send buffer, the send rate, and if the client is generating cover traffic, and if so, at what rate. It is important to note that the attributes `rate_ack`, `ACK`, `retransmit`, `dummies_acks`, and `max_retransmissions` are never used.
- **Section 9** - *miscellaneous*: This section keeps information denoting the length of a message-id and the number of total real messages we want to send during the simulation.

3.1.3 Baseline Simulation and Profiling Measurements

In our endeavour to assess Simulator, we run a series of experiments to identify its performance and quality of results. In particular, the baseline simulation examines the Simulator's behaviour in terms of space, memory, time complexity, and the network's entropy. We run the same experiment for the three established topologies - Cascade(10 Mixes), Multi-Cascade (3x10 Mixes) and Stratified (3x10 Mixes) - while increasing the simulation duration from 100 to 10000 time ticks. We also configured the mixing technique to Poisson (i.e. Continues), the number of clients to 100, and the number of target_send_messages to 1000.

Our anticipation regarding memory usage is that all experiments will deliver the same results for all topologies except Cascade. We establish our assumption on the fact that a single Cascade topology has fewer Mixes than the other topologies described earlier. However, according to Figure..., our hypothesis is false since the Simulator uses the same amount of memory - roughly 247 megabytes - in all experiments. In a later chapter, we will see that memory usage solely depends on the number of users we are simulating.

In regards to space complexity, we expect to have differences among topologies. Specifically, a Cascade Mixnet needs to route traffic through a chain of nodes. The network's capacity aligns with a single mix's capacity. Therefore, every time we dump each Mix's logs, we observe log lines to repeat because messages stall there for a long time. A Multi-Cascade Mixnet performs better since there are parallel chains of Mixes. The Stratified topology should be the best since messages continually move from one Mix to another and do not repeat in the log files so many times. Indeed, our hypothesis is accurate, and we can notice that in Figure ...

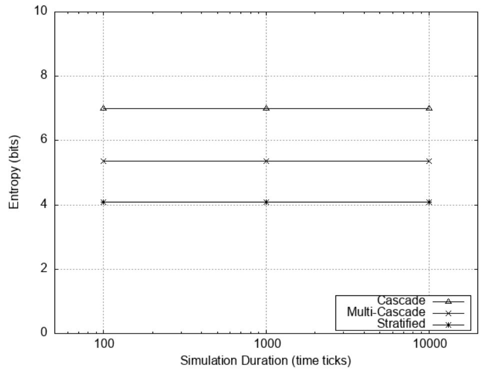
Execution time should follow the same pattern as space complexity, given the same reason concerning the nature of each topology, as noted before. We can observe in Figure ... that this is mostly true, except for the Simulation duration of 1000 time ticks. This negligible time offset on Figure ... can be caused by other processes running on our workstation at that moment.

Further, we need to check if Piotrowska's simulator reasonably captures the notion of entropy for each topology. According to our measurements in Figure ..., Simulator reports higher entropy for Cascade topology. There follows Multi Cascade and Stratified. These results are acceptable since aggregating more messages on each hop increases Mix's anonymity set, so entropy is higher.

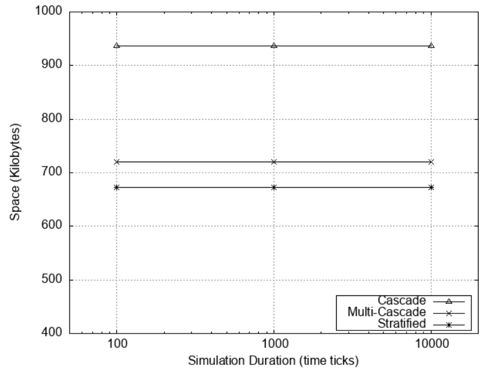
Finally, we run Piotrowska's project using a *cProfiler*[] to get visual information and statistics about its execution tree (sequence of functions called). We get a detailed representation of the execution tree in Figure. We also analyze the project using *SonarQube* to get information about the quality of the code. In the first place, *cProfiler* reveals that 59.75% of the CPU usage was for processing packets, while the heaviest task running under the *packet processing* branch was *entropy update*. Entropy update took 38.76% of the total CPU usage, indicating a possible bottleneck to the whole execution. Regarding *SonarQube* results, we find out that 2.7% of the code is duplicated, and there are 57 *code smells*, 2 *bugs* and 5 *security issues* of no great importance.

3.2 MiXiM

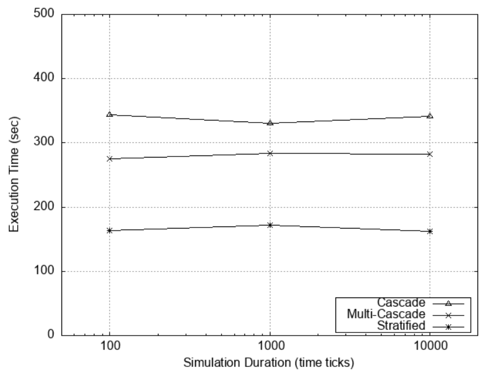
Guirat et al. carried out a similar attempt to develop an event-driven Mix Network simulator. In particular, Guirat et al., in their study[], present MiXiM, a simulation framework that helps academics and industry professionals assess various Mix Networks design options. Next, we will delve into the simulator implementation, reporting on its functionality, available features and input parameters. Besides that, we aim to run the simulator against a baseline scenario, reporting on its outputs and execution statistics. This information will help us decide if MiXiM is a well-built and feature-proof tool that



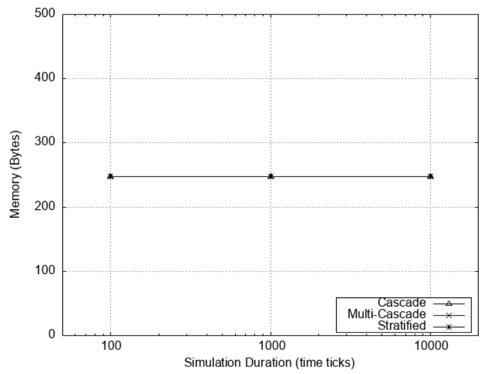
(a) Entropy



(b) Space



(c) Execution Time



(d) Memory

Figure 3.1: Simulator Baseline Experiments

brings out the most.

After analyzing MiXiM, we understood the code intrinsics and captured its approach when simulating a network. Likewise, Piotrowska's work is well-designed and abstracted into similar entities: (a)Simulation, (b) Network, (c) Mix Node - Poisson Mix, Pool, Timed Mix, Threshold Mix - (d) Client, (e) Relay - abstraction of an attacker - and (f) Message.

Furthermore, the simulator allows users to run experiments given various configurations. In more detail, there is support for many topologies, such as (1) cascade, (2) multi-cascade, and (3) stratified. Additionally, there is an option for three mixing techniques (1) Timed mixing and (2) Poisson mixing and (3) Threshold Mixing. Besides that, the users can enable dummy traffic generation, giving a specific generation rate. Another outstanding feature is the introduction of corrupted mix nodes during the simulation. Finally, the Simulator outputs the anonymity metric of entropy.

3.2.1 Execution Workflow

The workflow of executing a standard simulation scenario is as follows:

1. Read user-generated configuration file.
2. Create and initialize a simulation object passing all the input parameters found in step 1.
3. Call the run function of the simulation object.
4. Create and initialize a Network object comprising a set of Clients and Mix Nodes, including possible corrupted nodes. Connect nodes with their neighbours.
5. Then all clients begin generating traffic according to the given λ generation variable. In the meantime, dummy traffic is generated as well.
6. The simulation is executed in a single phase lasting for several time ticks declared in the configuration file. During each message transmission from one node to another, entropy is updated.
7. After inspecting the execution logs, MiXiM presents the entropy to the user.

3.2.2 Input Parameters and Configuration File

MiXiM, compared to Simulator, does not take any command-line arguments. We only observe a single configuration file with six different sections. We present a sample configuration file in Appendix .... Following, we provide a thorough description of the aforementioned file.

- **Section 1** - *DEFAULT*: Under the DEFAULT section, we can see the number of clients sending and receiving messages in a network and a parameter called lambda_c representing the rate of generating messages per time tick.
- **Section 2** - *TOPOLOGY*: In this section, we describe settings for all three supported topologies - Stratified, Cascade and Free Route. The parameter fully_connected enables us to have a complete or a semi-connected Stratified topology. Also, here we define the routing strategy by choosing between the source - users choose the route - or hop_by_hop - random selection. Other parameters in this section include E2E, the minimum end-to-end transmission delay,

and n_layers, l_mixes_per_layer and n_cascades, which describe the number of layers and the corresponding number of mixes on each layer.

- **Section 3** - *MIXING*: This section is related to the available mixing techniques. We have the mix_type, which can be Poisson, Time, or Pool and mu, which is the delay on each mix node. For each of the aforementioned mixing techniques, the variables timeout, threshold, and flush_percent, match each case accordingly.

- **Section 4** - *DUMMIES*: This section regards dummy data generation. The user can set the variables client_dummies to enable or disable dummy messages, rate_client_dummies to control the dummy data generation, link_based_dummies to send dummies dropped on the next hop, multiple_hop_dummies to send dummies that last for many hops, and rate_mix_dummies to generate dummy messages on the Mix.

- **Section 5** - *NODE_SELECTION*: All attributes in this section are not required during the simulation and might be the fragments of older software versions.

- **Section 6** - *THREAT_MODEL*: Here, we define the number of corrupted nodes and if we want them uniformly distributed across the network layers.

3.2.3 Baseline Simulation and Profiling Measurements

Section 3.1.3 described our attempt to evaluate the Simulator according to a baseline scenario. We make the same assumptions for MiXiM, running the same experiments. Unfortunately, our expectations did not match the results since we could not reproduce every simulation.

Mainly, we managed to simulate Stratified topologies for 100 and 1000 time ticks. Attempting to run experiments for Cascade and Multi-Cascade network arrangements failed due to run-time errors. Simulating a Stratified topology for 10000 time ticks also resulted in a crash. We were hesitant about the failure causes, so we decided to probe the software code-base to uncover potential issues.

Indeed, skimming the source code, we managed to spot four bugs we present in the following list.

1. The parsing section of the code accepts as valid topologies ”cascade”, ”multi-cascade”, and ”stratified”. Nevertheless, this is not the case for some files - Network.py, Simulation.py and Client.py - where the code expects a topology

called ”XDR”. This contradiction causes MiXiM to crash, given the topologies ”cascade” and ”multi-cascade”.

2. The variable n_cascade, which describes the number of parallel Cascades in Network.py, is hard coded. Hence, the simulation is not assuming the user’s configuration.

3. In file Simulation.py, a variable is declared as otherClient, while the software developer used the name other_client when initialising the variable. Consequently, the initialisation is meaningless, and otherClient has no value.

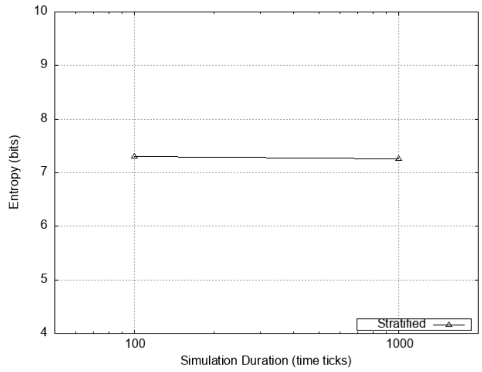
4. The initial parsing of boolean parameters from the configuration file is incorrect since the code used is a Python2 command which, when using Python3 - which other project libraries require - always returns True. Therefore, all boolean parameters in the configuration file are True, no matter their actual value.

Anyhow, the purpose of this project is not to fix someone else’s work but asses it. Consequently, we did not proceed with further code reviews or repairs.

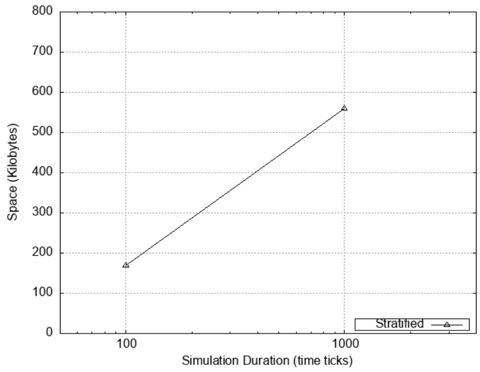
In regards to the successful experiments, we can observe the results in Figure. Surprisingly, the execution time and space used to dump log files increase linearly for different simulation time-frames. According to our previous code analysis, this is justifiable since clients in MiXiM generate messages based on the λ rate. The longer the simulation, the more messages are generated. Hence, more processing time and storage space are required. On the other hand, Simulator uses only the number of specified target messages regardless of the duration. Furthermore, it is notable that in terms of memory usage, MiXiM is sufficient, requiring only 21.4 megabytes of RAM for all our successful trials. Lastly, we cannot deduce further valuable insights from these graphs since, as mentioned earlier, we failed to perform all baseline experiments.

3.3 Compasisson Framework

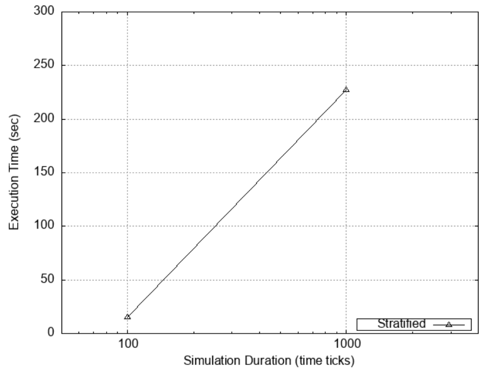
To bring conclusiveness to our simulator analysis, we present in this section a comparison framework. Our ultimate goal is to find out which simulator is better to adopt for further development in the future. A comprehensive comparison framework needs to evaluate multiple factors. Therefore we assumed the following KPIs (key performance indicators) (a) Topologies, (b) Mixing Techniques, (c) Routing Methods for Stratified



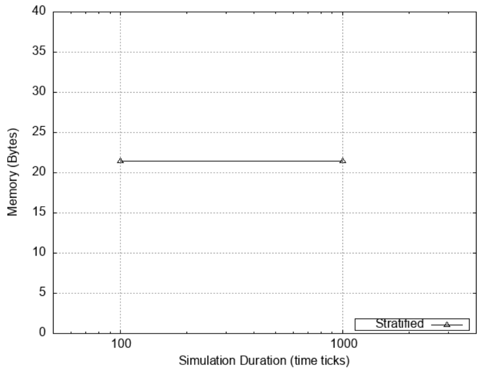
(a) Entropy



(b) Space



(c) Execution Time



(d) Memory

Figure 3.2: MiXiM Baseline Experiments

topology, (d) Cover Traffic, (e) Corrupt Relays, and (f) SonarQube Code analysis insights.

Table ... illustrates how MiXiM and Simulator compare in each aspect. We notice that MiXiM realises 13 features, while Simulator has only 9. Nonetheless, it is essential to prompt that most of MiXiM's features are not working, crashing on runtime. Also, both software achieves high scores in object abstraction, while code readability for MiXiM seems to be poor. Further, MiXiM has 81 code smells compared to Simulator, which has 57. Moreover, MiXiM has double the bugs and security issues compared to Simulator, which has 2 and 5, respectively. Eventually, both projects have negligible amounts of code duplication.

Ultimately, we decided that the best simulator so far is Piotrawskas Simulator. We justify our decision mainly based on the usability aspect. Even though MiXiM advertises a large set of features, it fails to deliver a working code-base. We are confident that Simulator is a well-designed software and can be improved so in the future support more features. Besides, Simulator satisfied our reasoning for valid and quality entropy

		Simulator	MiXiM
Topologies	Cascade	✓	✓
	Multi-Cascade	✓	✓
	Stratified - Fully Connected	✓	✓
	Stratified - Semi Connected		✓
	P2P	✓	
Mixing Techniques	Timed		✓
	Threshold	✓	✓
	Pool		✓
	Continues	✓	✓
Routing	Random Choice	✓	✓
Cover Traffic	On Client	✓	✓
	On Mix	✓	✓
	Multi-hop Dummies		✓
Relays	Corrupt Mixes		✓
Code Analysis	Code Readability	4/5	2/5
	Object Abstraction	5/5	5/5
	Code Smells	57	81
	Bugs	2	4
	Security Issues	5	10
	Duplicate Code	2.7%	0%
	Runtime Crashes		✓

Table 3.1: Simulators Comparisson Table

and network latency results.

Chapter 4

Replication Study

This chapter is dedicated to replicating existing literature[], reasoning about its outcomes. Minutely, a replication study frames the process of reproducing and analysing the experiments of other researchers under similar conditions. The purpose of a replication study is primarily to discover if current findings provide a valid basis for deciding future research paths.

Within the context of this dissertation, we conduct a short replication study, considering previous works[]. Our goal is to see how existing topologies, mixing techniques, cover/decoy traffic and the appearance of corrupter Mixes affect the latency and anonymity of a Mixnet. This analysis will help us to reason about Mixnets behaviour, further extending recent research in chapter 5.

Replicating previous work was challenging for many reasons. For our part, the experimental set-up was not clearly described, and some parameters were missing. Consequently, we had to make assumptions on a few occasions. Another essential aspect when replicating studies is to use similar environments. Nonetheless, non of the previous research had references to the employed hardware.

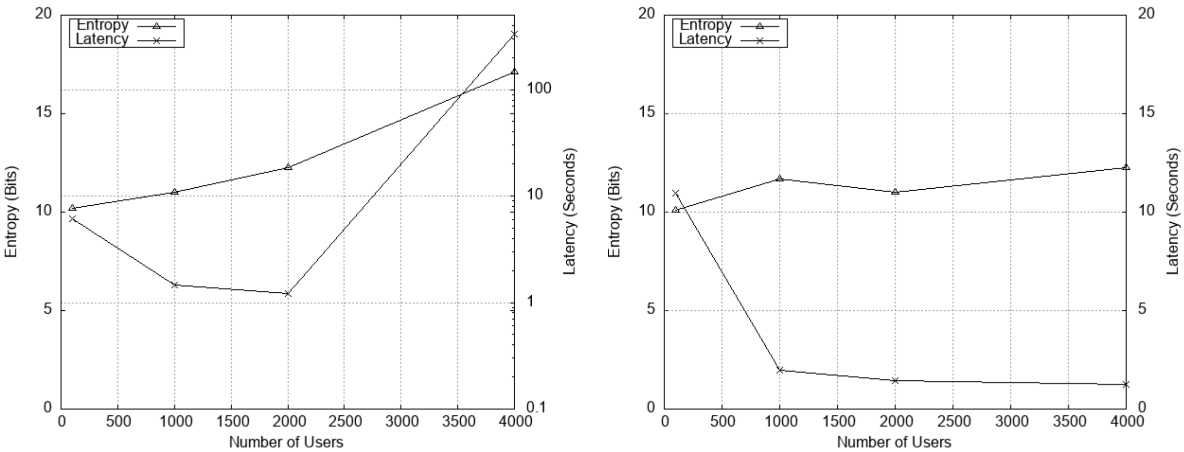
For this reason, we used our personal computer, comprising an Ubuntu machine facilitating a 6-core Intel core i5-8600K and 16 GB of RAM. The experiments conducted throughout chapters 4 and 5 use exclusively this equipment. At this point, it is worth mentioning that simulating more than 4000 clients on Piotrawska's Simulator required more than 16 GB of RAM, which was out practical limit. Therefore, we managed to replicate most of the experiments partially.

4.1 Studying the Effect of Mix Network Topologies and Mixing Techniques

Indubitably, different topologies and mixing techniques, as described chapter 2, can impact the anonymity and latency of a Mix Network. Piotrowska saught the same questions when writing her study on the anonymity trilemma[]. The conducted experiments evaluated and compared three Mixnets implementations - Elixxir Network, HOPR and Nym. Each one of these projects is designed using a different topology.

Nym integrates a stratified topology based on three layers with 3 Mixes each. Also, the mixing technique used is ”continues”, where the delay is randomly picked from the exponential distribution. On the other hand, Elixxir is a Chaumian cascade network based on the cMix protocol[]. Elixxir can be configured to run on a single Cascade, like cMix, or use multiple Cascades. Compared to Nym, Elixxir uses the batch and reorder mixing technique (for Elixxir Network, we assumed a 1000 message threshold). Lastly, HOPR uses a P2P topology, and for this reason, we will not include it in this work.

In the first place, we are going to compare Single-Cascade and Multi-Cascade network topologies. Intuitively, we expect that a Single-Cascade formation will have higher anonymity levels since all traffic is routed through a chain of Nodes. Shuffling the whole anonymity set of messages on each stop can provide greater anonymity. Nonetheless, we assume that latency can be high because the network's capacity is caped to the Mix's throughput. The latency can be low for a negligible number of clients, but it can get higher if we increase the user base.



(a) Single-Cascade Elixxir Network

(b) Multi-Cascade Elixxir Network

Figure 4.1: MiXiM Baseline Experiments

Indeed, Figures ... and ... portray the expected results. Assuming Cascades of length three and by increasing the number of users, we observe an increase in entropy as well. Also, we remark that the entropy increase rate is much more extensive for a Single-Cascade (1x3) than for a Multi-Cascade(2x3) topology. Nonetheless, there is evidence of poor latency in Figure 1a when the system facilitates more than 2000 users. Latency sharply increases from just above 1 second to almost 350 seconds. On the other hand, a Multi-Cascade design allows very low latency by sacrificing a bit of anonymity. From the above analysis, we can deduce that for Cascade and Multi-Cascade topologies, we have to choose between scalability and anonymity. We can not have both.

In contrast to Cascade topologies, a Stratified topology should scale better. This is intuitive since we can add more Mixes to each layer, distributing traffic into distinct routes. At the same time, we deem it is possible to achieve lower latency since not many messages get congested into a Mix.

Indeed, as we notice in Figure ..., Nym's Stratified implementation - 3 layers of 3 Mixes each - can achieve greater anonymity with very low latency. We presume that latency is at the level of 0.3 seconds because Nym realises a continuous mixing technique, while earlier, Elixxir used a batch and reorder technique, meaning that batch size can affect latency. Moreover, entropy is in an uptrend, showing that by increasing the number of users, anonymity increases as well. Therefore, Nym's Stratified configuration can handle significant traffic with increasing anonymity and low latency.

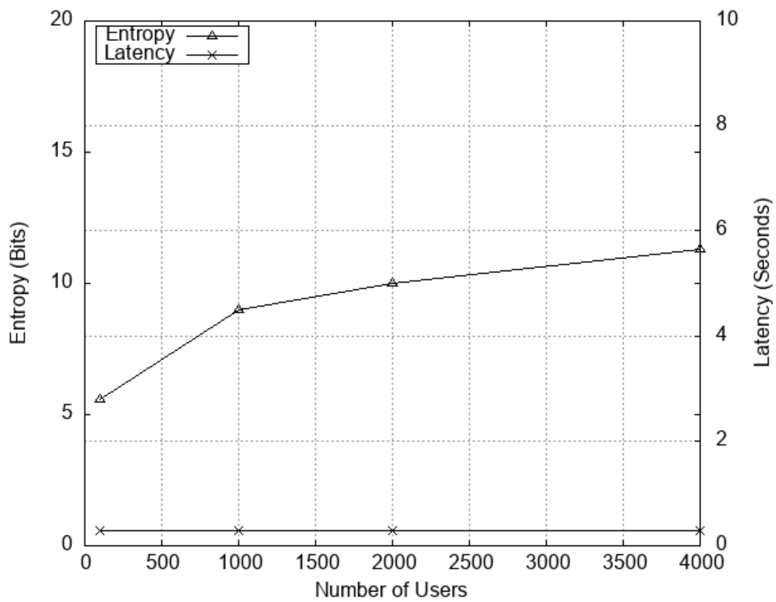
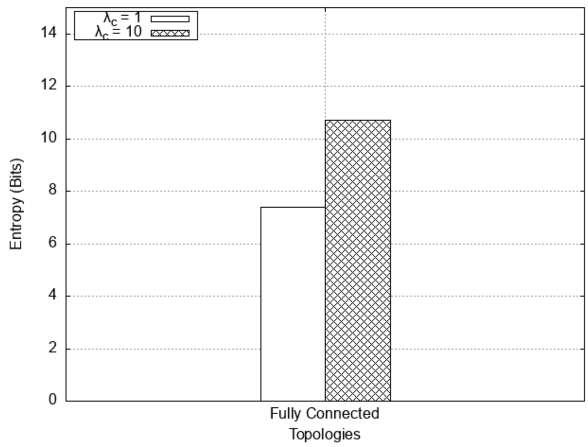


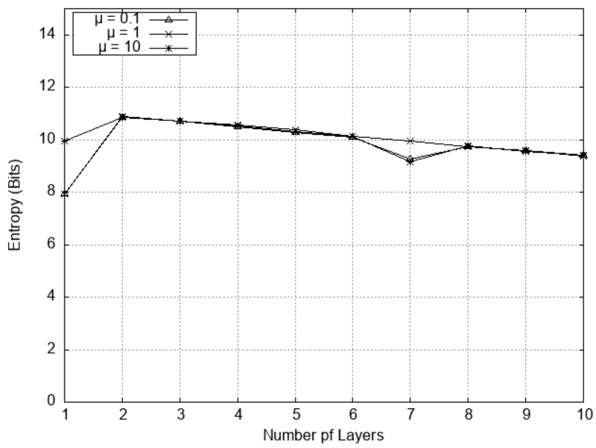
Figure 4.2: Stratified Nym Network

Using MiXiM, we get similar results. In particular, Figure ... depicts the anonymity of a Stratified topology of 3 layers comprising 10 Mixes each, simulated by setting the λ message generation factor to 1 and 10, respectively. We observe that entropy increases when traffic rises, conforming to the previous experiment's results.

Also, Figure .... shows us the impact on entropy when increasing the number of layers, assuming different average delays on mixing - presuming Continuous mixing. We observe anonymity growing between 1 and 2 layers; however, there is a steady decline afterwards. We are not sure why this is happening. Intuitively, the longer the route a message takes, the bigger should be the entropy. Besides, assuming various mixing delays, the entropy should have been higher for more extended time frames. According to the simulation results, this is not true, as the drawn curves seem primarily identical. Nonetheless, we are not confident about the quality of the results since, as we have studied in chapter 3, MiXiM has many bugs and unrealised features.



(a) Impact of different topologies on Entropy



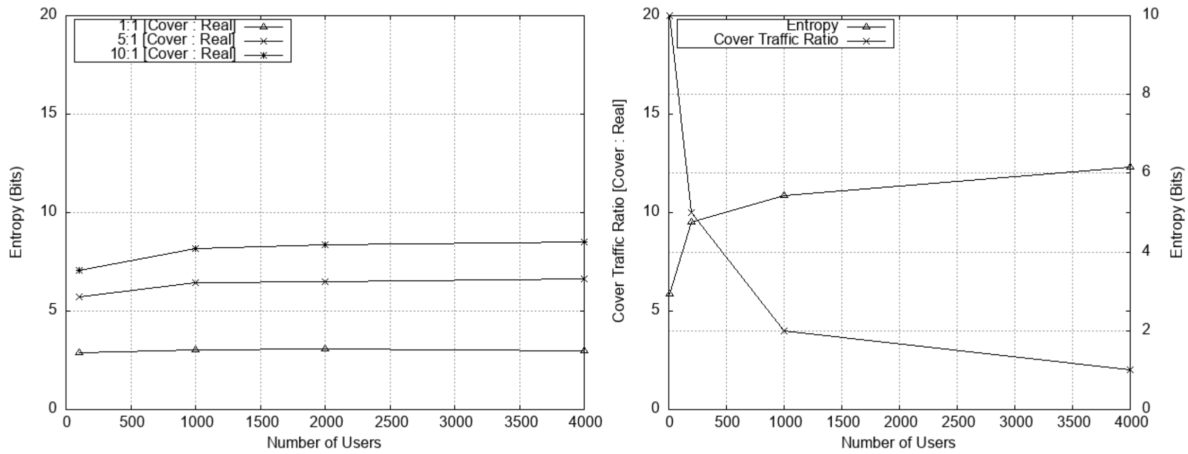
(b) Impact of different number of layers and average delay on Entropy

4.2 Studying the Effect of Traffic

Cover traffic is an essential tool that both MiXiM and Simulator incorporate. Notably, using dummy messages, we increase the anonymity set during the mixing stage. Allegedly, we anticipate greater anonymity when having more traffic. Also, cover traffic enables us to aim for the same anonymity levels while reducing latency. Theoretically, we achieve that by increasing traffic and keeping the average delay low - assuming the continuous mixing technique.

4.2.1 Cover Traffic

Figure ... illustrates the effect of cover traffic on anonymity, assuming different cover traffic ratios. We observe that for higher ratios, anonymity is greater. Also, increasing the number of users contributes to anonymity. Notably, in Figure ... we notice that decreasing the cover traffic ratio from 10:1, to 2:1 [Cover:Real] yields acceptable levels of entropy, which is further boosted by the increasing number of users.



(a) Impact of Cover Traffic on Entropy

(b) Impact of Cover Traffic on Entropy

4.2.2 Real Traffic

Real traffic is generated according to the number of users in our network. Therefore, more users imply higher anonymity levels. In fact, as noted by [simulator] and [mixim], increasing the average delay of Continues mixing offers greater anonymity. Nevertheless, what if we want to keep latency down? As we notice in Figure ... this can be achieved by keeping the average delay down. As more and more users join a Stratified network, the anonymity increases; thus, we can employ low mixing delays. As a result, we have a network with low latency yet constant anonymity.

4.3 Studying the Effect of Corrupt Relays

Another exciting subject investigated by Guirat et al. is the appearance of corrupted Mixes within a Stratified network. The notion of a corrupt node implies that the adversary knows all input and output messages with complete certainty.

The results illustrated in Figure ... refer to a Stratified network of 3 layers comprising ten mixes each. The mixing technique used is Continues, with an average delay of

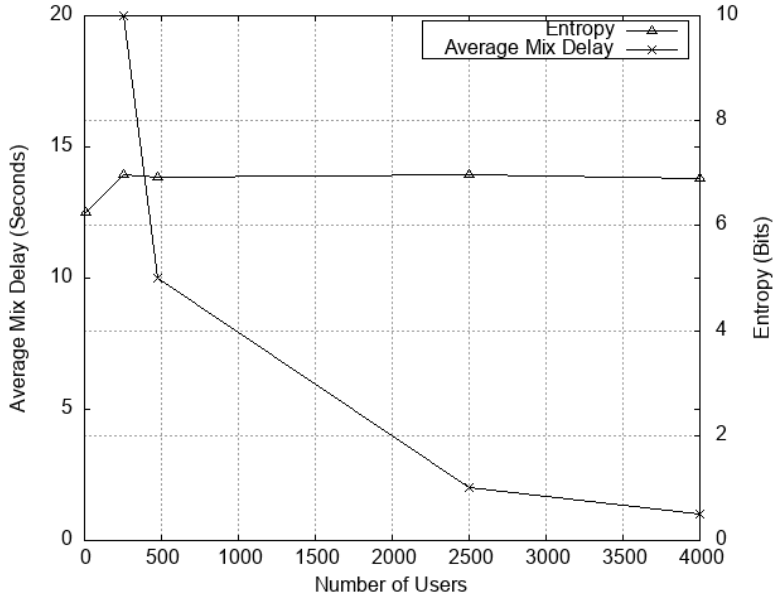


Figure 4.5: Constant Entropy with reduced Latency

0.1 seconds. Also, we facilitate 100 clients, each generating ten messages per second. For this experimental set-up, we anticipate declining anonymity by increasing the percentage of corrupted mixes.

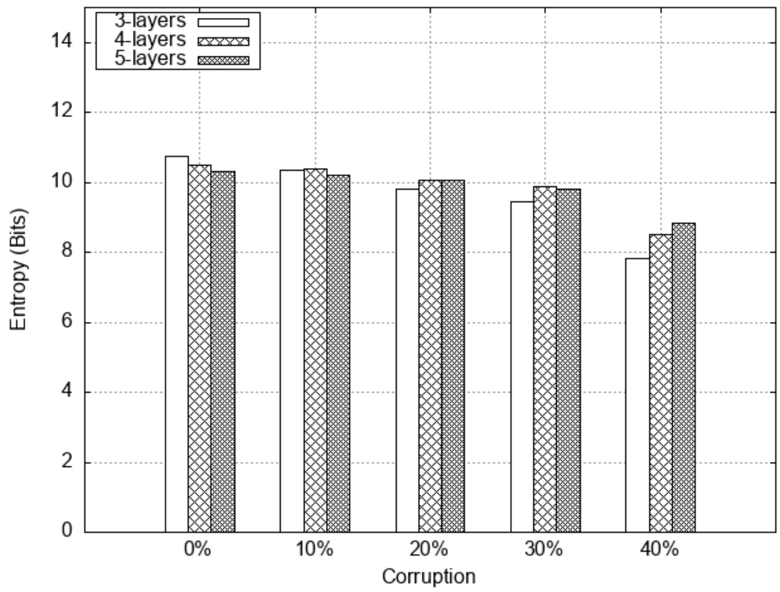


Figure 4.6: Impact of Corrupted Mixes on Entropy

Indeed, Figure .... depicts rigid results. There is a steady decrease in anonymity each time we increase the fraction of corrupt nodes. Moreover, it is interesting to observe that for 0% and 10% of corruption, the three-layer scheme provides better anonymity, while this is not true for 20%, 30% and 40% of corruption. Instead, we notice that the higher the corruption fraction, the lower the entropy is for the three-layer and four-layer Stratified Mixnet. In contrast, the entropy of a five-layer design seems to perform better. Intuitively, this is reasonable because a network with more layers provides the possibility of having less interference by an adversary. On the other hand, a network with fewer layers but more adversarial mixes implies a greater possibility of maliciously influencing the communication between mixes.

Ultimately, considering the effect of the number of layers on the network's anonymity according to [mixim], and the results of our experiment in Figure ..., we deduce that it is challenging to design a Mixnet. In the first place, we observe that increasing the number of layers does not much contribute to the anonymity while introducing corrupt relays appears to make more extensive networks necessary. Nevertheless, augmenting a Mixnet implies computational and hardware expenses. It is beyon question that there are some apparent tradeoffs between anonymity and scalability when adversaries emerge.

Chapter 5

Exteding Current Research Using Simulator

Chapter 5 discusses our main contribution to the Mixnets field, which is the simulation and evaluation of arbitrarily formed Stratified networks. Particularly, we assumed that a Stratified Mix Network might have imbalanced layers due to crashing Mixes. Within the context of this chapter, we study two different types of imbalanced Mixntes, which are (a) dynamically imbalanced and (b) statically imbalanced. Additionally, we examine the impact on anonymity caused by increasing the number of Mixes per layer. Lastly, any experiments in this chapter operated on our modified version of Piotrawska's Simulator[].

5.1 Stratified Networks with Increasing Number of Mixes

In previous research, Guirat et al. investigated the topic of increasing the number of layers in a Stratified network. In this section of chapter 5, we are experimenting with the number of Mixes per layer regarding Stratified networks.

Our experimental setup is pretty straightforward. We are using a Stratified topology and Continues mixing technique, with an average mixing delay of 0.1 seconds. Also, we are running many iterations of the experiment using a different number of Mixes per layer. Intuitively, we expect designs with more Mixes to yield worse anonymity. We justify our supposition, assuming that each Mix will eventually receive less traffic. It is a matter of the fact that less traffic implies a smaller anonymity set; hence a global adversary might be able to identify message senders and receivers smoothly. Still, assuming that we are using the Continues mixing technique, which has a memory-less property, the network's entropy should fluctuate in a tight range.

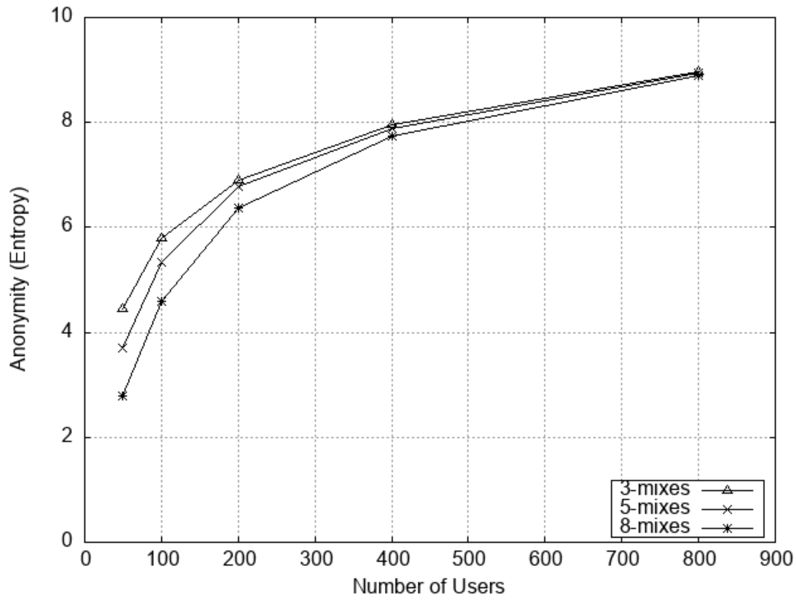


Figure 5.1: Effect of Increasing the Number of Mixes per Layer

Indeed, observing Figure ..., we deduce that entropy is initially different, but as we increase the network's user base, it converges. Potentially, we can declare that the three interpolations are logarithmic-like for a low number of users. Nonetheless, we are not sure if entropy diverges for a more extensive user base. Also, we notice that netowrks with more Mixes in each layer have lower entropy for 10 to 400 users. This is reasonable because, as mentioned earlier, less traffic is routed through mixes. Regarding the increased number of users, it is evident that larger user bases yield better anonymity overall.

5.2 Dynamically Imbalanced Stratified Network

The notion of a dynamically imbalanced Stratified network revolves around the idea of having random Mixes crashing arbitrarily through different message transmission rounds. Our goal is to make each Mix crash according to a given probability. Mainly, we extended Simulator to handle this feature by tossing a coin. In other words, before we sample a random route for a potential message, we call a function producing a Real number between 0 and 1. A value less than 0.2 implies that a Mix crashes successfully, with a probability of 20%. Also, in the provision of a non-live route (i.e. the subsequent layer has only malfunctioning Mixes and, therefore, we can not determine a valid path), we increase the message queue delay by 1 second. Our decision was natural since there is wasted time when a Mix tries to find the next live destination.

In our experimental setup, we assumed a Stratified network with three layers comprising ten Mixes. Also, we set the mixing techniques to Continues, with an average delay of 0.1 seconds. Further, we allow cover traffic of a 1:1 ratio. Then we ran our simulation for 20%, 40% and 60% probability of failure for each Mix individually.

Our initial theory considered that increasing the failure probability would decrease anonymity because a considerable amount of Mixes receive less traffic. This hypothesis is invalidated since anonymity seems to remain persistent for different failure probabilities. According to Figure ..., for a 20% crash probability, we get slightly higher entropy, assuming 100 clients in the network. Nonetheless, once we scale up the user base and crash probabilities, the anonymity ranges in equivalent levels. Besides that, our initial assumption for higher latency is false since, as we notice in Figure ..., latency remains constant.

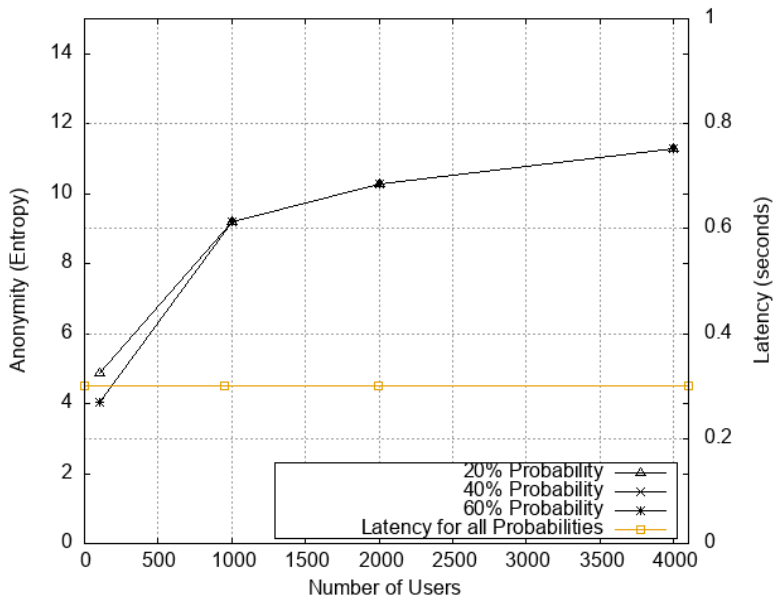


Figure 5.2: need to add caption

The above results are reasonable and justified through the simulation logs. In more detail, skimming the log files, we observe that invalid paths occur sporadically; hence the increased latency of a few messages does not affect the average network latency. Moreover, anonymity is not affected because entropy fluctuations seem to cancel out each other after simulating many messages for many users. In other words, a route of Mixes might not appear many times in the short term, but after sampling routes for 1000 messages, all possible paths seem to be selected an equal amount of times. Therefore, anonymity converges to a single value.

5.3 Statically Imbalanced Stratified Network

Examining the case study of a dynamically imbalanced stratified network was fascinating. A parallel idea arose to investigate statically imbalanced Stratified networks as well. Our perception of ”static” describes a formation where each layer has a fixed number of Mixes, yet this number is different. In particular, we seek to study the effect on anonymity when deploying more or fewer Mixes on the first, middle and last layer of a Stratified Mixnet.

Our experimental setup comprises a three-layer Stratified topology enabling Continues mixing with an average delay of 0.1 seconds. We also employ coverer traffic of 1:1 ratio. Initially, we assess the impact on anonymity based on a variable number of Mixes in the middle layer, testing one, two, and three Mixes setups. Next, we repeat the same trial starting with 1 Mix and steadily increasing this number to 3 Mixes per layer for the first and last layer, respectively.

According to our analysis in Figures ..., increasing the number of layers regardless of the layer position impacts anonymity likewise. This happens because all mixes repeat the same process in all layers and cover traffic flows continuously from senders to Mixes and from Mixes to recipients. Nonetheless, something different occurs assuming a smaller user base. Notably, a 3 Mix per layer configuration yields slightly higher entropy compared to 2 Mix layers. Likewise, a 2 Mix per layer network has narrowly taller entropy than a 1 Mix per layer design. The results of this Figure are credible since, by definition, routing more traffic into a Mix results in better anonymity.

Finally, we observe latency at a constant level of 0.3. The exact value was reported in the previous section of chapter 5, where we simulated a dynamically imbalanced network. This was expected since latency, as realised earlier, is not mainly affected in network designs utilising the Continues mixing technique.

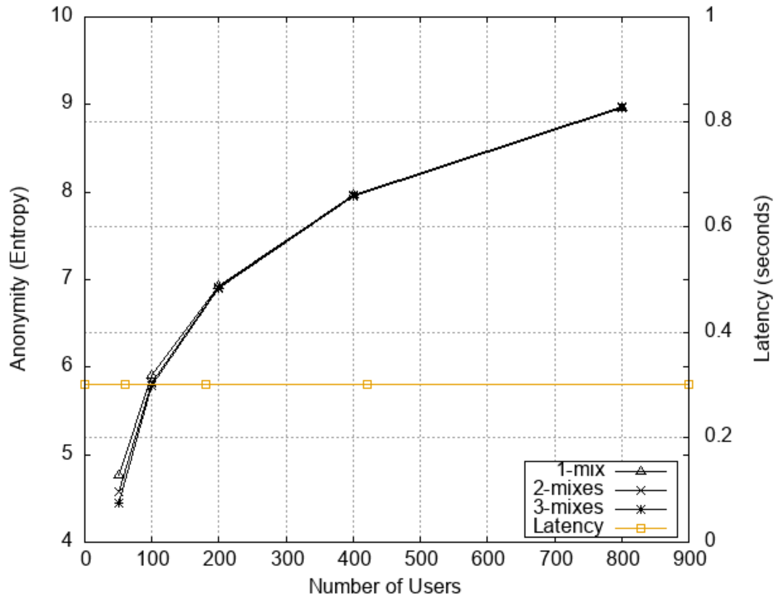


Figure 5.3: need to add caption

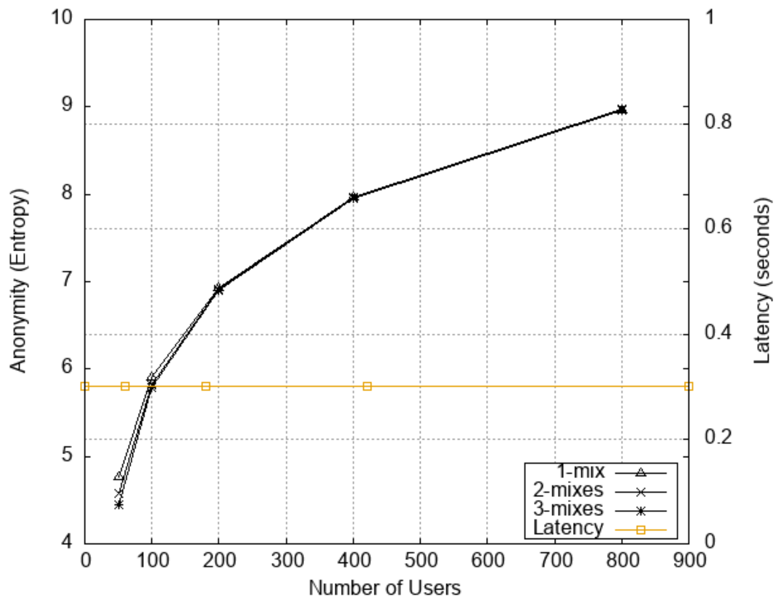


Figure 5.4: need to add caption

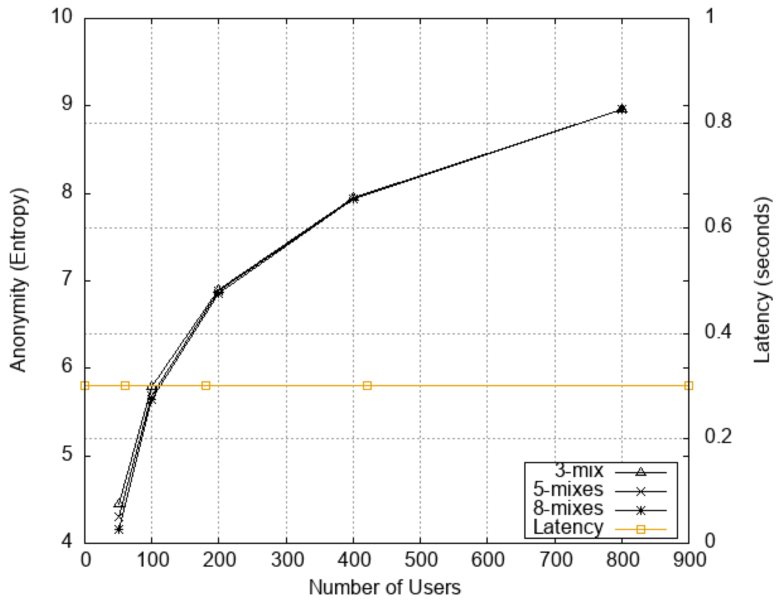


Figure 5.5: need to add caption

Chapter 6

Conclusion

6.1 Final Reminder

The body of your dissertation, before the references and any appendices, *must* finish by page 40. The introduction, after preliminary material, should have started on page 1.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default 1.5 spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages, such as `fullpage` or `savetrees`, change the margins of your document. Do not include them!

Over-length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

Bibliography

Appendix A

First appendix

A.1 First section

Any appendices, including any required ethics information, should be included after the references.

Markers do not have to consider appendices. Make sure that your contributions are made clear in the main body of the dissertation (within the page limit).

Appendix B

Participants' information sheet

If you had human participants, include key information that they were given in an appendix, and point to it from the ethics declaration.

Appendix C

Participants' consent form

If you had human participants, include information about how consent was gathered in an appendix, and point to it from the ethics declaration.