Feasibility of Mix Networks for Instant Messaging

Master's Thesis

by

Christian Loehle

at the

Universität Konstanz

Faculty of Sciences

Department of Computer and Information Science

1. Supervisor: Prof. Dr. Marcel Waldvogel

2. Supervisor: apl. Prof. Dr. Sven Kosub

January 2021

Abstract

Private messaging has seen widespread interest in the recent years. While end-to-end encryption is widely adopted among messengers now, metadata analysis can be used unrestrictedly. To prevent accumulation of metadata at providers and network links, the traffic they route needs to be anonymous and unlinkable to any potential sender or receiver. Mix networks are an anonymous communication network design that has been studied academically since 1981. The traditional use case was Email-like, asynchronous one-to-one communication that is used without any context other than the one provided in the message itself.

One problem of mix networks was unpredictable delivery delay which made these systems unpractical. Recent research has been putting uses of mix networks closer to that of traditional messaging.

In this work feasibility of Instant Messaging over mix networks is analyzed with regards to its usability (i.e. the introduced delays), the anonymity of users and the security.

Contents

Chapter 1

Introduction

Private messaging is one fundamental practical problem that is still unsolved. While end-to-end encryption has seen widespread adoption it only applies to message's contents. But this usually is not even enough to provide the famous *Confidentiality* of IT-systems. Since metadata, like message size, can provide enough information to an adversary to infer the contents of a message. Further, confidentiality turned out to be just one building block for private messaging.

Even confidential messages can be associated to identities (sender and receiver), their location, sending and arrival time, message length, frequency of communication and communication volume. We call all of these metadata. Metadata proves to be more and more valuable. With metadata social graphs are easily built automatically without cumbersome reading of messages' contents. Since all metadata comes in some known format, unlike human messages, it is suitable for mass surveillance.

> Metadata absolutely tells you everything about somebody's life. If you have enough metadata you don't really need content.
>
> (Stewart Baker, former NSA general counsel)

The widespread centralized messaging systems like WhatsApp, Telegram and Signal put all metadata in the hands of one party. But even federalized messaging systems like XMPP suffer similar metadata problems. While the metadata will ideally only be known to the involved parties' providers, these can suffer data leakage or be legally forced to hand out such metadata. A passive attacker eavesdropping the network traffic (at any of the nodes of a message's route) also gains access to metadata.

There is no communication without metadata, but it can be reduced and be made less informative to an adversary. Current means of messaging usually don't take any measures to curb metadata collection.

Private messaging must anonymize messages to unlink them from the involved identities. Only then no third party can infer any information from the metadata on the network. If anonymity is not required, identification can then be implemented ontop of such a system for the involved identities to confirm each other. Anonymity, therefore, is the foundation for any private messaging system.

1.1 Terminology & Assumptions

There is some common terminology in this work that will be explained here. While terminology in the literature is ambiguous we explicitly use the terminology as proposed by Pfitzmann et al. [PH00]

1.1.1 Network Setup

Anonymous Communication Networks (ACNs) in general assume some underlying network infrastructure of interconnected devices. ACNs that are built upon existing deployed networks are called *overlay networks*. For most ACNs the underlying network is simply the internet. Devices participating explicitly and knowingly in the ACN are called *nodes* (As opposed to network devices of the internet that are not aware of the ACN and treat its packet as any other packet, like network switches). The topology of the underlying network is not of further relevance as long as there is a way for any pair of nodes to

communicate. Similarly the security of the network links is not directly relevant, as they are usually treated equal as described later in 1.1.4.

Communication takes place in form of messages from senders to receivers. The set of possible senders and possible receivers is usually considered to be equal and has only semantic difference.

1.1.2 Anonymity

Anonymity intuitively means that an observed message cannot be linked to a user (sender or receiver). The canonical formal definition of anonymity borrows from Shannon's secrecy definition: [Sha49].

The definition allows the adversary to have some a priori information about the message. Formally the message m is randomly sampled from a message space $\mathcal{M}$, with a distribution known to the adversary. If *perfect secrecy* is given, the adversary will not learn anything new from observing the ciphertext $c = enc(m, k)$.

Anonymity is not concerned with the message's content but with the involved parties, namely the sender and receiver. [1] The a priori knowledge is called the *anonymity set* and consists of all possible actors that are involved with the message. Similarly to the message space the anonymity set is fixed (i.e. it cannot increase by any action) and finite. The attacker's a priori knowledge assumes a uniform probability distribution over the anonymity set. The attacker will then infer from observations an *anonymity delta*, a change in probabilities of some elements (i.e. identities) of the anonymity set being more likely than others. [2]

The *anonymity set* is the total set of possible users for that message, e.g. for a letter in a postal box anyone who used that postal box since the last emptying can be considered a possible sender. In analogy to the classic cryptography model we often refer to the elements of the anonymity set as the *set of Alices* or the *set of Bobs*.

A more in depth analysis of anonymity will be done in Section 2.3.

1.1.3 Mix Networks

A single mix can be thought of as a store-and-forward device in the network. Additionally to forwarding its goal is to unlink the incoming and outgoing packets. This is achieved through changing the packet's appearance (e.g. through decryption) and its flow (by reordering incoming and outgoing packets). An instance of a mix network is simply called the network and the participating mixes are referred to as the network's *mix nodes*.

Delays refer to the delays introduced by the mix network as a message traverses through. As mixes are often chained with multiple hops of mixes we call the delay that originates from an individual mix node the *hop delay*. Traditional network delays as in network routing latencies are of no particular concern for this work, making the term *unambiguous*.

1.1.4 Threat Models

The most often used threat model in mix network literature is that of a *Global Passive Adversary* (GPA). A GPA is assumed to be able to eavesdrop all traffic on all links simultaneously. For certain attacks active abilities are additionally given to the adversary. These can include inserting own messages and delaying or dropping arbitrary messages.

An adversary can also *collude* which can be thought of as giving them additional abilities. For example can a GPA gain additional information about some user if they have a colluding mix network node or a colluding user of the system (e.g. communicatee with the target).

The attacker's goal is usually to deanonymize a particular user. If not explicitly stated otherwise the reader can assume that the attacker's goal is to deanonymize a user. Other goals, particularly Denial-of-Service attacks, which aim to make the network as a whole unusable and force users to communicate over less secure channels, are explicitly stated when studied in this work.

[1] Encryption is usually added on top of anonymous systems.

[2] In practice this is conveyed by a warning of ACNs that they do not hide the fact that the user is using an ACN. Anonymization (e.g. Tor's obfuscation projects) that also hides this is hard to capture formally and not effective in practice. [WDA+15] [HHG19] An analogy would be between cryptography and steganography. In fact steganographic techniques are often used in censorship-resistant communication. [HBS13]

1.2 Anonymous Communication Networks

We introduce some common *Anonymous Communication Networks*. The goal of all ACNs is anonymization, but their threat models differ greatly. Since we are only concerned with mix networks in this work, we can differentiate between ACNs that consider the GPA as part of their threat model and ones that do not.

1.2.1 Traffic Analysis

Traffic analysis is the method of a third party to acquire metadata by observing the traffic patterns. With bidirectional stream-based traffic the analysis has been shown to be very powerful. Song et al. could infer the contents of an ssh session by traffic analysis. [SWT01] Further Miller et al. analyzed HTTPS traffic to infer content related to medical data entered in a web form without touching the actual HTTPS encryption. [MHJT14] With recent data retention laws traffic analysis becomes a bigger threat while still being underestimated.

> Traffic analysis not cryptanalysis is the backbone of communications intelligence.
>
> (Susan Landau & Whitfield Diffie, Privacy on the Line: The Politics of Wiretapping and Encryption) [DL07]

Danezis et al. examined the history and threats of traffic analysis more thoroughly. [DC07]

ACNs primary goal is to defend against traffic analysis. Likewise is traffic analysis the primary method of attack for the adversary. [BMS01] Cryptographic primitives and implementations are assumed to be unbreakable for the adversary in our work.

1.2.2 Tor

Tor is not a mix network and does not aim to protect against a GPA doing traffic analysis. [Din]

There are no anonymizing delays and no cover traffic, the traffic is encrypted and routed through several nodes, called relays, until it leaves the Tor network through so-called exit nodes, relays that allow outbound traffic.

While considering a weaker threat model, Tor has succeeded in widespread adoption of an ACN. Its technical details also inspired practical mix network development, such as the Public-Key Infrastructure (PKI) providing an overview of the network to clients. This PKI and its trust is spread among several *authorities*, trusted nodes that vote on the network's state such as which nodes will be included and what their public keys are. This eliminates the single point of failure and distributes that trust.

Since the technical details of Tor are more widely understood, it makes sense to draw parallels to Tor when describing mix networks.

While Tor does not protect against a GPA doing traffic analysis, it was widely believed to be an unpractical threat anyway. More recent research shows very effective traffic analysis attacks. [NBH18] [SEV+15] [JWJ+13]

1.2.3 DC-Nets

DC-Nets are based upon the theoretical Dining Cryptographers problem which asks, if everyone participates, can one of the participants transfer information (1 bit) without anyone else knowing whom it was sent from. The solution to this problem (broadcasting 1 bit among all participants anonymously) can then be generalized to an ACN, the DC-Net.

DC-Nets perform in rounds, which require computation and transmission of all participants. The most basic solution let's one participant add a random bit to the outputs, which can then be cancelled out. [Cha88] [GJ04]

For practical use an obvious problem for DC-Nets is that clients might go offline if they don't use the network to communicate actively. Even more so because of the high computation overhead and bandwidth usage, which is then still required. DC-Nets also require pairwise keys with all participants of the network, which would scale badly in practice.

There is has not been any real world use of DC-Nets, but they can be seen as the opposite of mix networks: they consider the GPA, but introduce no further delay to the traffic (apart from communication

and computational overhead). The anonymization solely comes from everyone sending and receiving the same amount of traffic.

1.2.4 Mix Networks

Mix networks achieve anonymity by using several layers of mixes. Each mix tries to unlink incoming and outgoing messages by shuffling them. This introduces additional delay as messages will have to wait at a mix node for further messages to be shuffled with.

The assumption is then that enough mixes in the network are *honest* and do not collude with the attacker, such that at one of the mixes they cannot tell which of the outgoing messages is the sender's message with a significant advantage over guessing.

The corresponding privacy notion is referred to as *Sender Unlinkability*.

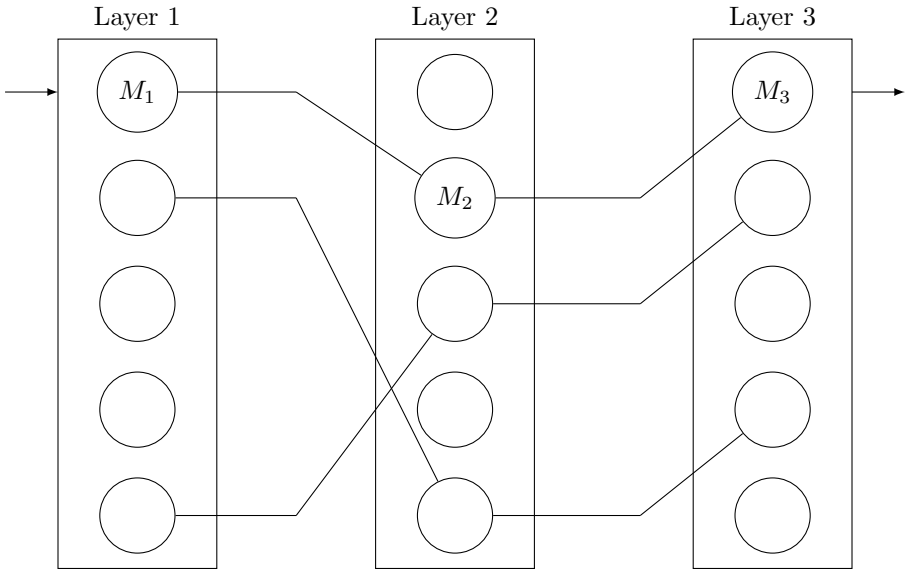

Figure 1.1: A typical mixnet topology: the stratified topology

To understand how Sender-Unlinkability is achieved, let's follow a message through a mixnet depicted in figure 1.1. Sender Alice chose mixes M_1, M_2 and M_3 when crafting the message m. [3] It is obvious that the message needs to change (bitwise) appearance at each mixing stage, otherwise an observer can link them easily. This is achieved through encrypting the message in an onion-fashion (or layered) with each of the mixes' public keys. Thus the message will be crafted like this

$$m = Enc(K_{M_1}, Enc(K_{M_2}, Enc(K_{M_3}, text))).$$

Where K_{M_i} is the corresponding key for M_i and $text$ is the actual message Alice intends to send to Bob (which additionally may or may not be end-to-end encrypted).

So far an observer can still easily correlate the message going in and out at each mix. In fact the figure looks similar to Tor, which as we noted earlier does not protect against an adversary as powerful as a GPA (which we call Eve for now). The mixes will need to shuffle the messages coming in before forwarding them. Generally, we can say that the message is delayed by some time at each mix. A mix *firing* refers to the point in time where the messages (some or all) are forwarded. The set of messages that are forwarded when the mix fires is called the *batch*. There are many different strategies for determining the time of firing and what messages are included in the batch, we will discuss them in section 2.1. For now the reader may consider all mixes firing all available messages every day at 1pm.

When the observer Eve wants to follow Alice's message, they end up at M_1. When M_1 fires it will forward the n messages it has received in the last 24 hours. For Eve all n messages are equally likely to

[3] Note that Alice must therefore know the network's information. If M_1 chose the next mix, they could obviously choose other colluding mixes.

be the one from Alice. This same process repeats for M_2 and M_3 and then it is forwarded by M_3 to Bob. Note that Eve does not know that Alice and Bob are communicating, that is exactly what she wants to find out. Eve does know the details of the mixing strategy and therefore in our case knows that the message will be fired at the next possible 1pm, additionally we give her the advantage of knowing when the message arrives at Bob. She knows by observing that Alice has sent a message in the last 3 days and she knows everyone who received a message through the mixnet at the same firing time as Alice's message arrived.

As Eve has to consider all users who have received a message at that time frame, she cannot tell whom Alice's message was addressed to. [4]

1.2.5 Anonymity Trilemma

We will briefly mention the *Anonymity Trilemma* that proved bounds of the three parameters anonymity, latency and bandwidth. If we want to achieve a certain degree of anonymity we have to sacrifice either (or a combination of) low-latency and low bandwidth. So we can already say that DC-nets do not sacrifice any latency, therefore are high bandwidth. Mix networks on the other hand mostly sacrifice their latency through delays at each hop of the mix network. [DMMK18]

A further analysis of the anonymity trilemma is in chapter 5.3.

1.2.6 Other ACNs

We will briefly describe other approaches of ACNs and how they relate to mix networks and the IM use case. These can be categorizated by their method of anonymization, threat model or security goal respectively. It should be noted that these design proposals have come out of academic research and none of them have seen serious real-world adoption.

DC-Nets Basis

Herbivore is a practical proposal for a DC-Net with acceptable latency and reasonable (although still high at around 300 KB/s for text messaging) bandwidth usage. For this the authors restricted DC group size significantly. Groups are connected with each other through servers, but they do not belong to the same anonymity set. In their evaluation they assume $10 - 40$ members in a group (and therefore in the anonymity set) of which only $1 - 4$ are sending. [GRPS03]

Dissent [WCGFJ12] is a DC-Nets approach for anonymous group communication. The group must be a fixed set of participants. Dissent guarantees sender-anonymity among participants of the group. Their prototype supported up to 40 members with latencies of several minutes, increasing cubically with the group size.

Mix Network Basis

Vuvuzela [vdHLZZ15] consists of a global mix chain and inbox providers. Interactions from clients to their inboxes go through the mixes in sequence. Mixes add noise (dummy traffic) according to a differential privacy definition to hide access patterns at the inboxes. While the author claim to be able to have a 36 second end-to-end delay, the global chain does not scale well when mix servers are added, since each server adds its own noise to the inboxes to guarantee privacy. Unfortunately, the number of mix servers is critical to distributing trust in the system, as one mix server must be honest to provide privacy. For just 6 mix servers they evaluate the end-to-end latency to be over two minutes. Further the dialing protocol to initiate a conversation requires significant client bandwidth, because it works in a broadcast manner.

Stadium reduces reliance over Vuvuzela by not relaying on one global mix chain. [TGL+17] The lowest latency they report in their evaluation is still around 400 seconds.

XRD [KLD20] is a design that implements messaging based on an underlying mix network with a verifiable shuffle to protect against active attacks. As XRD inboxes are not retrieved through the mix

[4] Avid readers might doubt this, since Eve knows that M_1 was used as a first mix, so only receivers having a path from M_1 come into question. Such questions about mixnet topology are analyzed in Section 2.2. For now the reader is hopefully convinced by considering that with increasing user size and randomly chosen mix paths this will approach all receivers anyway.

network, the inbox is linked to the corresponding identity. The design goal is to pretect against metadata collection and does not enable communication of parties anonymous to each other. [5]

The XRD communication takes places in rounds, during which all users will send an equal amount of messages to prevent traffic analysis. Since the messages move in globally-synchronized rounds, relatively high delay lower bounds are introduced. The lowest number mentioned by the authors is arounds 200 seconds, disqualifying it for IM usage.

Atom [KCGDF17] is another mix network based design, that incorporates protection against active attacks to a trusted third party. The proposal claims to scale well with respect to the number of users and server, but their use case is a microblogging scenario. A message here is received and read by many users of the system. The numbers the authors give in their system evaluation is a 28 minute message latency for one million users and a maximum message length of 160 bytes.

Private Information Retrieval

Private Information Retrieval (PIR) is a method of querying a database (server) without revealing which records were queried. The cost of PIR is additional query bandwidth and computational overhead. PIR has been of interest more recently to be another approach for building ACNs.

Riposte [CGBM15] proposes a messaging system with anonymity against a GPA. Their proposed use case is a messaging board, or a *private Twitter*. The authors identify the use case of mostly readers and few writers and leverage this for their design. While the design achieves a throughput of several client requests per second, [6] it does not scale well for private messaging. They demonstrate it can be used for this purpose, but clients would need to tolerate hours of latency for messages.

Popcorn [GCM+16] uses PIR to hide what media content is consumed. The authors propose some significant improvement performance-wise but the work is focused on a CDN (Constant Distribution Network) use case, like streaming platforms. Such a use case is very high volume, but many users share the same records of interest. Popcorn leverages this intensively, making the improvements unlikely to be useful for IM, with low user traffic volume, high frequency and mostly records that are only relevant for one other user (in one-to-one messaging).

DP5 [BDG15] and AnNotify [PHG+17] are designs for notification and presence service in a publisher-subscriber model that uses PIR. Both use variable-length epochs, where an update will be published in the next epoch (in the best case). A lower bound for epoch times is not given, but they mention 1 minute. Further the protocols performance relies on write-once, access many times, which would not necessarily be the case for an IM usage.

Always-On Connections

Traffic analysis can be prevented if the traffic streams of all users are equal without introducing delays apart from several relays to hide correspondence. Such systems usually allow for only one correspondence per user at a time in that bandwidth limit. Unfortunately, these system proposals often neglect the anonymity loss by users going offline (which is reasonable for them, as bandwidth usage is equal, even when the system is not actually used to communicate).

Herd proposes such an always-on connection to be used between VoIP systems. [BCC+15] Apart from the high bandwidth waste during non-usage, their threat model is restricted to so called *trust zones* where traffic analysis does not take place.

Drac [DDTL10] is a constant stream connection for VoIP that persists among trusted *friends*. The set of *friends* are revealed to an observer, which poses a problem for anonymity and usability. While the anonymity sets are comparatively small, they persist over a longer time.

Weaker Threat Models & Security Goals

Aqua [LBCZ+13] proposes a system design for anonymizing high-bandwidth usage like BitTorrent. They use network nodes with constant traffic in between and at the edges (from a client to a node) to divide

[5] This prevents many use cases, consider a whistleblower who wants to communicate with a journalist. If they are a high-profile whistleblower, they might not trust the journalist to not reveal their identity and prefer it to not be known to even their communication partner.

[6] In the anonymity world this is considered efficient.

users into groups of similar bandwidth usage. These are then padded to a constant traffic rate. Therefore Aqua only provides k-anonymity among the k users of similar bandwidth usage.

1.2.7 Reencryption Mix Networks

We covered traditional, often called *decryption mix networks*, as the input and output at a mix are changed bitwise by the mix performing a decryption of the message. For this a client has to encrypt the message with the mixes' public keys respectively beforehand. A different approach that finds practical use in voting systems is that of a reencryption mix network. Instead of being decrypted at a mix, all mixes share a key pair. At each stage the messages are being changed bitwise by reencrypting the message with respect to the public key of the mix network. After mixing, the mixes have to do a final decryption of their shared secret key.

Reencryption mix networks find application in voting schemes, where the primary goal is to hide not a vote's content but the author (what is referred to as a sender in the messaging world). For actual messages intended for a particular recipient, the mix network additionally needs routing information and the recipient's public key. Instead of decrypting at the final hop, the mix network will perform a *key switch*, an operation where the ciphertext is encrypted under a different public key (the receiver's) without ever decrypting it to a plaintext. [Jak99] If the mixes do not collude, the plaintext of the messages are not revealed and the sender cannot recognize their message. Such a mix network to be used as a two-way communication was proposed by Golle et al. [GJ03]

As the setup cost of such a mix network is high due to the mixes sharing their keys, universal re-encryption mix networks have been proposed. [GJJS04] Instead of reencrypting with a mix network public key, reencryption factors are given. The reencryption takes place for the recipient's public key, without the mixes being able to identify that public key. As a result there is no routing information and all messages are posted to a bulletin board, where every recipient must decrypt every message to see if it is intended for them. The massive overhead makes this unpractical for instant messaging applications. Danezis also identified some attacks against universal reencryption mix networks. [Dan07]

Additionally the high level of synchronization needed for the joint decryption will be problematic for instant messaging use cases. We will thus focus on decryption mixnets for the rest of this work.

1.2.8 Instant Messaging

So far we've considered communication between two parties in the form of messages. This work considers the nuances of instant messaging (IM) in particular.

While the user interaction of an IM application is clearly distinct from a general messaging (i.e. email) application, they are not directly related to the underlying technical implementation (that is using the mix network).

The most obvious criteria is clearly delivery speed which should *feel* instantenous. But other than that IM applications extend normal messaging by contact lists, chat history, group messaging and presence (some even up to the point of notifying if the other person is currently typing). These are hard to formalize in terms of technical requirements, but for now we will consider them as follows:

1. A bidirectional communication with participants identifying each other. [7]

2. Messages should be delivered as fast as possible. We set an arbitrarily chosen limit at 1 minute for our analysis.

3. Messages are exchanged at a high frequeny (the content of one email is often split up into many short IMs).

Proper group messaging is left out of scope for this work and requires further research.

[7] This is obvious in the context of IM, but historically mix networks were used for anonymous remailers, some of which were only able to hide a message's origin without providing a way to reply non-publicly to its sender.

1.3 Motivation

The internet experience for end-users today is mostly web-based, this poses a huge challenge for mix networks and anonymization. In the case of mix networks additional delays are introduced and throughput is limited. This seems impossible to reconcile with webpages, where sizes of several megabytes are not unusual. Applications have moved to being web applications with constant connectivity and interactivity, even though the underlying use case might not strictly need this. Consider webmail: A user might not care if their email is delivered with 5 minutes delay, but if their network packets have a 5 minute delay the webmail application will not be usable.

Instant Messaging is one of the instances where non-webbased applications are still used. WhatsApp for example uses XMPP, an open protocol, although a proprietary derivation of it. [Fou]

Users are willing to install a messenger application instead of using a webapp. This opens up the possibility to integrate the technical complexity a mix network-based messenger requires without much additional burden to the user.

Furthermore mix networks are a promising approach to private messaging as they put little load on the client, both bandwidth and computation-wise. High bandwidth solutions are not practical for mobile users, which is the trend for personal devices. Therefore, if widespread private messaging is to be achieved, mix networks are a suitable candidate. For this we examine the feasibility of mix networks for the IM use case.

Existing IM applications that promise anonymity and no trusted party like Briar and Ricochet use Tor. We believe that the IM requirements are not as strict as a stream-oriented bidirectional channel like Tor provides. Therefore relaxing the requirements to the ones suited for IM might allow using other anonymization techniques, namely introducing delay (i.e. mixing messages). So if we can deploy anonymization techniques with stronger threat models without affecting user experience significantly it should be done.

Chapter 2

Mix Network Design

We will introduce current mix network research in this chapter. First we examine the mixing strategy concerning a single mix, then we look at how they can be arranged in a topology to form a mix network.

2.1 Mix Strategies

The mix as a building block has to permute the inputs and outputs such that the permutation is not conceivable by an adversary observing traffic. We already described that this requires bitwise unlinkability of messages. Further it requires messages to be indistinguishable in size.

Definiton 2.1. *Messages M output from a mix are said to be indistinguishable, if an adversary cannot guess which input message m resulted in the output message m' with significant advantage over guessing for any $m \in M$.*

For now we assume this is guaranteed by the packet format. [1]

The last requirement to prevent basic traffic analysis is to hide order and timing. This is achieved by shuffling or batching messages at the mix. Several batching strategies were proposed in mix network research.

2.1.1 Threshold Mix

Mixes aim to mix incoming messages such that outgoing messages cannot be linked to the incoming ones anymore. The first approach proposed by Chaum [Cha81] was for a mix to wait for n messages until it forwards all of them at the same time in random order. This is referred to as the *threshold mix* with n being the value of the threshold. It follows that the anonymity set is guaranteed to be of size n (assuming distinct senders).

The expected delay is unknown and dependent on the incoming traffic rate that mix. Threshold mixes can be attacked easily with a so-called $(n-1)$ attack, during which an adversary inserts own messages to fire a mix. The adversary then excludes their own messages from the anonymity set. Ideally the adversary could insert $(n-1)$ of their own messages, therefore only leaving one message whose origin then is revealed. The vulnerability to the $(n-1)$ attack inspired the following mixing strategies.

2.1.2 Timed Mix

A *timed mix* will fire after a certain time t has passed. The anonymity set is therefore all messages that have arrived since the previous firing. This makes the anonymity analysis depend on the traffic rate. Additionally an active attacker can delay incoming messages, such that the anonymity set is reduced (See $(n-1)$ attack 4.3).

[1] Details on packet formats can be found in the according literature like the Sphinx packet format [DG09]

2.1.3 Timed and Threshold Mix

Threshold and timed can be combined to a firing strategy that either fires when both conditions are satisfied or either are satisfied. The analysis applies similarly to such a hybrid firing strategy.

2.1.4 Pool Mix

The Mixmaster mix network implementation used *pool mixes* in addition to the firing strategy. [MCPS04] The mix retains a constant pool of messages, only some of which are forwarded when the mix fires. A pool mix trades a higher delay variance for an increased anonymity set. In fact, since the anonymity set approaches infinity over time (a message could theoretically never be flushed from the pool), analysis of anonymity for pool mixes uses information-theoretic anonymity in favor of the anonymity set cardinality. For further information we refer to section 2.3 and the according literature. [Ser04] [SN03] Pool mixes also transform the $(n-1)$ attack into an attack with uncertainty (See section 4.3).

In the Mixminion system nodes use a *dynamic* timed and threshold pool mix, which fires every t seconds if some amount of message are in the pool. The dynamic pool mix (or Cottrell mix), retains a fraction of messages in the pool instead of a constant amount. When firing, a constant fraction of messages in the pool are forwarded. The messages are chosen independently of how long they are already in the pool. [DDM03]

2.1.5 Continuous Mix

Kesdogan et al. proposed a different approach to the batching strategy, the *stop-and-go mix*. [KEB98]

Instead of batching all currently held messages at the mix it treats each message individually. A message is delayed at the mix without regarding any other message being delayed by the mix. It is clear that the delay has to be drawn from a probability distribution to perform any mixing at all.

We will only consider delays drawn from an exponential distribution as they are proven to be optimal with respect to provided anonymity in relation to expected latency by Danezis. [Dan05]

A delay is drawn from an exponential distribution with parameter λ. For the random variable of the delay time D it then holds that

$$E[D] = \frac{1}{\lambda} \qquad Var[D] = \frac{1}{\lambda^2}.$$

We will later investigate the feasibility using the quantile function:

$$F^{-1}(p;\lambda) = \frac{-\ln(1-p)}{\lambda}, \qquad 0 \leq p < 1$$

The exponential distribution has the *memoryless property*.

Definiton 2.2. *a continuous random variable X is called memoryless if*

$$Pr(X > t+s | X > t) = Pr(X > s)$$

holds.

In the context of the stop-and-go mix this means that at any time any message that is held at the mix is equally likely to be forwarded next, regardless of its delay time up to that point.

The anonymity relies on the mix having enough other messages in the window of the exponential distribution parameter. We will investigate these consequences in chapter 5.3. Like with the pool mixes, the anonymity set based analysis cannot be used for continuous mixes, instead information-theoretic anonymity will be used in chapter 3. [Dan05] [YXB05]

2.1.6 Evaluation for Instant Messaging

We leave the anonymity analysis aside and focus on our requirements for IM.

According to our requirements we need to ensure a timely message delivery. Therefore mixes involving threshold batching strategies will be unpractical, their delivery time cannot be predicted ahead of time.

Timed mixes

For timed mixes we have the parameter t, the distance between each firing. Considering a single mix we can assume the message arrival time to be uniformly distributed and get $E(D) = t/2$. When cascading timed mixes the firing moment is usually synchronized as to prevent partitioning of the batch when two mixes fire right after each other and not all messages of the batch have been transmitted yet.

Assuming we use three cascaded timed mixes[2] we have $E(D) = 2t + t/2$. Thus if t is small enough timed mixes are a candidate latency-wise. For our assumption of delivery times within 1 minute we would have $t = 24s$.

Continuous mixes

For continuous mixes we have to consider unlikely high delay times for the exponential distribution. These are tolerable if they are unlikely enough. We compare these with network packet losses, where 1% is an acceptable packet loss rate. With reliable network protocols like TCP these lead to retransmissions after they have not been acknowledged for some time after the expected delivery time. Therefore packet losses in TCP also lead to additional delays.

Again cascading three mixes we need to hit the 99th quantile of 20 seconds 3 times in a row and we get

$$\lambda = \frac{-ln(1 - \sqrt[3]{0.99})}{20} = 0.285021805 \qquad \Longrightarrow \qquad E(D_\lambda) = 3.5085035.$$

2.2 Mix Network Topology

A single mix will have the anonymity properties previously discussed, but it has to be fully trusted. To distribute this trust mixes are chained for a message. Different choices of how to arrange a set of mixes, so called *mix network topologies* have been studied, namely cascades, free route and the stratified topology.

With increasing user size, topologies also try to address the issue of scalability, as a single mix handling a very large amount of messages is infeasible.

We divide mixes into *honest* mixes that work as intended, and *dishonest* mixes that collude with the adversary and reveal their message permutation. We assume n mixes of which d are dishonest, therefore the probability of choosing a dishonest mix with a single mix strategy is $\frac{d}{n}$.

For Instant Messaging the mix network topology plays a minor role, so we briefly recap the corresponding research. Cascades, in which fixed chains of mixes are globally known, are easier to compromise and are hard to scale. Free routes split the anonymity set into messages entering the mix network, messages leaving the mix network and messages staying in the mix network (assuming the adversary is aware of which nodes participate as mixes). The stratified topology divides the mixes into layers and lets the user choose a mix for each layer. The downside is that there needs to be a network consensus on the division of mixes into layers, which is of course critical to the anonymity. [DMT10] [Dan03]

2.3 Anonymity

The anonymity metric in the introduction was that of the anonymity set (cardinality). It works for timed and threshold mixes under the assumptions that all senders are distinct. This assumption might not be true as senders send more than one message in a round or because an attacker actively floods the mix for an $(n-1)$ attack. Further the anonymity set size is not a good metric for pool or continuous mixes.

Consider a pool mix (the firing strategy is irrelevant) with a pool size of 1. This mix will forward all available messages except for one randomly chosen message, which is kept for the next round. Since the process of choosing a message that is kept in the pool is independent each round, the same message can be chosen repeatedly and therefore remain in the pool indefinitely. Thus, the anonymity set are all messages ever sent, but their probabilites are vastly different.

Further consider a mix that has operated for r rounds and has a batch size of $n = 99$. For a message m_0 of the very first round to be in the batch it would have to survive r rounds and be selected for the

[2] a standard assumption in the literature and practical mix networks

current batch. Compare this with a message m_1 that was received in the current round.

$$P(m_0 \in batch) = 0.01^r * 0.99 \qquad P(m_1 \in batch) = 0.99$$

It is clear that the anonymity set size analysis is not suitable for pool mixes, or any mix where the time spent at a mix is not restricted, such as continuous mixes without a maximum delay.

2.3.1 Information-theoretic Anonymity

Since we could give probabilities for individual messages being linked to an outgoing message at a link, the next logical step is considering all such probabilities, thus the probability distribution over the anonymity set.

Serjantov et al. [SD03] and Diaz et al. [DSCP02] independently proposed using the entropy of the probability distribution of the anonymity set as a metric for anonymity. Information theory uses entropy as a measurement for information. In probability theory we can view this as a measure for uncertainty and applied for our adversary model entropy describes *How much more information does the attacker need to know everything.*

Definiton 2.3. *Given a random variable X, with possible outcomes x_i, each with probability $P(x_i)$, the entropy $H(X)$ is as follows:*

$$H(X) = -\sum_i P(x_i)\, log_2\, P(x_i)$$

We will focus on entropy as a measure of anonymity and refer to the probability theory literature [Fel68] for an introduction to the topic.

The two notions of the anonymity set and the distribution over the set are often referred as the *possibilistic* and *probabilistic* anonymity respectively. Diaz et al. extended the entropy to a normalized *anonymity degree*, that is:

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

with $H_M = log_2(N)$ and N the cardinality of the anonymity set (i.e. the number of senders).

Alternatives

Before we continue evaluating mixes with this informationan-theoretic anonymity it should be noted that this is not perfect. Tóth et al. summarized their criticism of the metric as follows:

> [...] The main problem with this entropy based measure is that it tries to quantify the amount of information that is required to break totally the anonymity of a message, i.e. to definitely identify actual sender of a message. However in practice we have to consider an attacker successful, if he can guess the sender of some selected messages with a good probability, in specific cases significantly greater than in the case of the uniform distribution. [THV04]

Instead they propose a so-called *local anonymity measure* with a parameter Θ. If the measure holds any message cannot be linked to a sender with probability greater than Θ. Additionally, they could construct, given an entropy value, a probability distribition with arbitrarily bad local anonymity i.e. with one sender being assigned a high probability.

While we view the criticism and proposal of the local anonymity measure as valid, we will still be using the canonical information-theoretic approach. While an anonymity system ensuring the local anonymity measure would be ideal, it is not fruitful to analyze current systems proposed in the literature and used in the real world with such a measure. The paper itself propose a protocol that ensures the property but requires a constant set of senders, with each sender sending exactly one message for each interval. Their argument shows that if a reasonable Θ is desired this constraint cannot be relaxed and it would be hard to reconcile it with real-world scenarios. [3] In fact, they could only show their local anonymity measure for sending messages, even though they defined the measure anagolous for the receiver side.

[3] For the IM use case this interval would have to be very small, resulting in large amounts of dummy traffic. [TH05]

Following their argument shows that for an interval each recipient needs to receive exactly one message, too. While ensuring that each recipient gets at least one message is conceivable (with dummy traffic), the coordination between all users required for a recipient to receive at most one message seems impossible.

Andersson et al. did a comparison of these different anonymity metrics. [AL08].

Dummy Traffic

It was already touched upon that senders may be able to send messages that are not for communication, so called dummy traffic, the same applies for mixes. This can provide security against an active adversary as seen in section 4.3. Furthermore it can provide security against a passive adversary. If the mix generates dummy traffic, the amount of incoming and outgoing messages will not match up. Therefore the adversary cannot be sure how many (real) messages are held at the mix at any time. [DP04]

Chapter 3

Evaluation of Information-theoretic Anonymity

Throughout this chapter the already presented mixing strategies will be evaluated with respect to the information-theoretic anonymity measure of entropy described in the previous chapter.

As part of this work simulations were implemented to be able to compute the anonymity provided by the mixes in an accurate manner. First the model and the assumptions will be described and justified. Based on the model the implementation of the simulation and its results are explained. Finally, the aim of the chapter is to answer the following questions:

- What degree of anonymity can at best be expected with IM requirements?
- Can the different mixing strategies be compared without sacrifing generalizibility of the results?
- How do the mixing strategies proposed in the literature fare in the IM setting with respect to anonymity? Does any one stand out?

3.1 Poisson Process

As providing anonymity means mixing messages to make them indistinguishable any study of anonymous communication will have to make some assumption on what scenario of messages are used as input. User input messages can be modelled with random variables.

3.1.1 Definition

Definiton 3.1. *A continuous random variable X is said to have the memoryless property iff*

$$P(X > s + t \mid X > t = P(X > t).$$

The only continuous distribution with the memoryless property is the exponential distribution. Informally speaking, memorylessness means that if the expected waiting time for an event to happen is t and s time has already been waited, the expected waiting time is still t.

An example where a random variable being memoryless seems intuitive is waiting for cars to pass by the street. If the expected time between cars passing by is 5 minutes, then even though no car passed by in the last 3 minutes, the expected time until the next car passes by is still 5 minutes.

Definiton 3.2. *A sequence of event-couting random variables $\{N(t),\ t \geq 0\}$ is a Poisson process with rate $\lambda > 0$ iff*

1. *$N(0) = 0$.*
2. *in disjunct periods the number of events is independent.*
3. *periods with equal length have an equal distribution of the number of events contained in them.*

4. $P\{N(h) = 1\} = \lambda h + o(h)$.

5. $P\{N(h) \geq 2\} = o(h)$.

From condition 2 and 3 the distribution of inter-arrival times between two events must be memoryless and therefore be an exponential distribution with parameter λ for the continuous case. [1]

3.1.2 User Input as Poisson Process

Modelling user input messages as a Poisson process has been the standard assumption, as Diaz notes in the original paper proposing entropy for anonymity measurement: [DSCP02]

> During the attack, we consider the number of senders constant, and senders behaving as independent, identical Poisson processes. This is a standard assumption for modeling the behavior of users making phone calls [Fel68]. This means that all users send, in average, the same amount of messages, and the interval of time between one message and the next one follows an exponential distribution.

In a more recent work Madani also used Poisson-modelled input throughout: [Mad]

> The incoming traffic, in accordance to the widely accepted assumption in the field, is Poisson distributed.

Diaz et al. [DSD] used real-world mix network traffic to analyze anonymity and have found it to be highly skewed and therefore argue that it cannot be modelled as a Poisson process. But it has to be noted that they analyzed for very long intervals (two graphs showing 3000 hours and 140 days). While real-world IM traffic might be appealing to use at first, their results will hardly be generalizable to a mix network-based IM scenario. For example, even if the underlying user behavior would be the same, if the mix network requires it, minor adjustments to message timing could be taken as long as they do not interfere with usability of the system. The Loopix system already does such an adjustment by having all its outgoing messages be exponentially distributed at the client.

Further Serjantov [Ser07] investigated the authors data and found:

> We briefly reexamined the same data and looking at shorter time horizons we find the distribution broadly Poisson [...].

Unfortunately, they do not explain what it means to be *broadly* Poisson more precisely. A formal notion could be helpful as outlined in section 3.1.4.

3.1.3 Exponential Mix

For exponential mixes which are chained, another aspect can be taken into consideration. We defend the assumption of modelling the arrival times as a Poisson process with the observation that user behavior does not directly translate into the arrival time (of one of the later mixes). Instead at least one delay drawn from the exponential distribution with parameter μ is applied, thus masking the arrival time from the sending time (of the user).

For exponential mixes distribution of arrival times (within a round so to say) also becomes relevant. Therefore, while a Poisson process might be more justifiable to be assumed for the exponential mix, it is also more crucial to the analysis that the input is actually Poisson. As such Kesdogan et al. when proposing the exponential mix also assumed Poisson input. [KEB98]

Loopix

Loopix, a modern mix network design based on exponential mixes, further defend the analysis of using the Poisson process by arguing that the traffic must be Poisson, since every client and mix generates traffic according to a Poisson distribution and combining Poisson processes yields a Poisson process. More precisely they claim: [PHE+17]

[1] For a formal proof we refer to the standard literature of queueing or probability theory. [Fel68]. The definition for the Poisson process is adapted from [Mye12].

> Honest clients and mixes generate drop cover traffic, loop traffic, and messaging traffic following a Poisson process. Aggregating Poisson processes results in a Poisson process with the sum of their rates, therefore we may model the streams of traffic received by a Poisson mix as a Poisson process. It is the superposition of traffic streams from multiple clients. It has a rate λn depending on the number of clients and the number of mix nodes.

Unfortunately, they do not take into account user churn, that is, clients leaving the network from time to time. Therefore the amount of Poisson processes merged will differ over time, violating the strict Poisson requirements.

3.1.4 Alternatives

Alternatives to the Poisson process as an input model have not been studied at all. This is not surprising as anything that is more realistic needs to take user behavior into account. This makes results less generalizable and thus less interesting for academical research.

Information-theoretic anonymity is used in more concrete scenarios when studying how mixes behave under attack. For a study of the mix itself under normal operation there does not seem to be any real alternative to modelling user inputs as a Poisson process.

Criticism of the Poisson process as a model is still valid and cannot easily be dismissed. In practice one will not observe a perfect Poisson process anyway, thus a notion of how good the Poisson model fits for the observed data would be useful. This notion would have to consider the interval that is relevant for the chosen mixing strategies and maybe even how vulnerable the strategy is to skewness from Poisson processes. For example it seems intuitive that exponential mixes chained together make even non-evenly distributed input more and more Poisson at each hop. A more indepth analysis on what properties are desired and how they can be formalized and measured could provide a basis for future research.

3.2 Computing Entropy

There has not been a comprehensive comparison of different mixing strategies with respect to expected delay and provided anonymity. This is because there are generally too many parameters to make such a comparison meaningful. Fortunately for this work and the IM use case we can fix the delay. For the exponential mixing strategy we showed what delay distribution yielded the desired properties. In this section the expected delay from the appropriately chosen λ is used for comparison. Therefore we have:

$$E(D_\lambda) = 3.5085035.$$

It has to be noted that we considered hitting the 99th percentile to achieve this value, therefore the variance of other mixing strategies will have to be taken into account where applicable.

Computing the entropy of the probability distribution requires some assumptions on the input. As described earlier we will be using Poisson process to model the input with different parameters. Since the entropy cannot easily be computed directly, we will sample concrete Poisson processes for input message arrival times in our simulation and calculate the yielded entropy. We consider this to give the most generalizable result, as we already have to make assumptions on the inputs being a Poisson process. Further assumptions are not necessary, as such we will simulate the behavior of a mix from a theoretical design viewpoint. No actual mixing will be sampled and for each message the probability that it still is in the anonymity set is stored. This differs to more concrete analysis often made from an attacker's viewpoint. A concrete analysis considers an adversary who can observe the number of messages leaving the mix and rule out input messages that must have been fired already due to the number of outgoing messages. We want to compare mixing strategies in general and are therefore interested in the theoretical entropy it yields, abstracting from any adversary capabilitiy.

As introduction we show results for a timed pool mix that keeps 1000 messages in the pool. Note that the expected delay is then dependent on the message volume, thus this graph is the only one not fixed to our $E(D_\lambda)$.

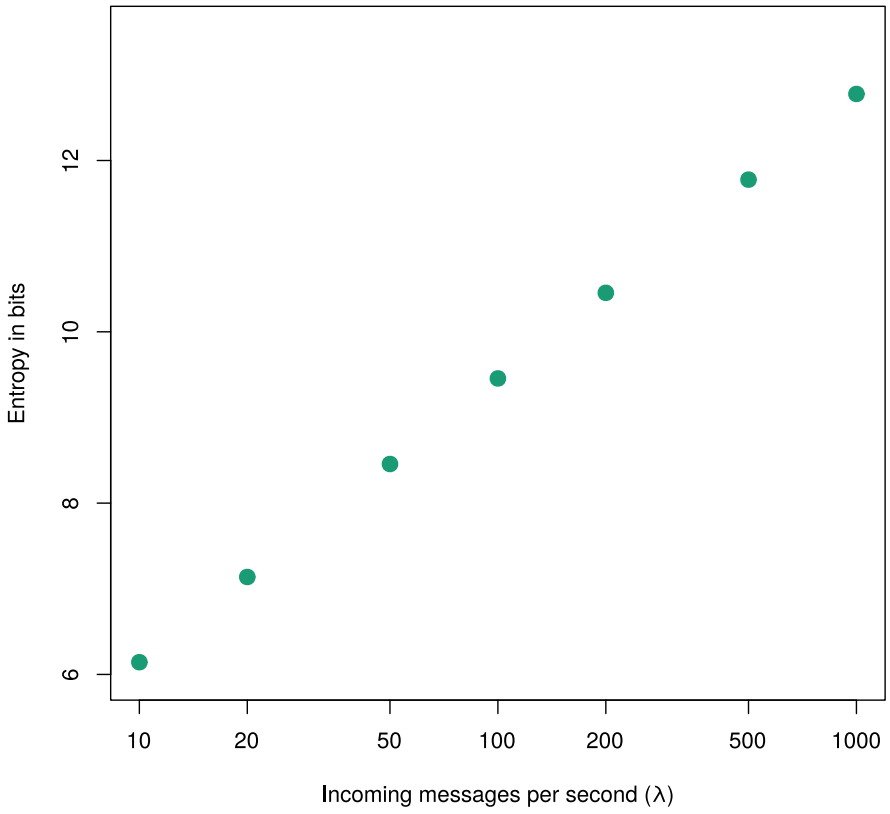

Figure 3.1: Entropy of a timed mix with period length 7.017. Average over 1000 trials.

3.2.1 Constant Fraction Timed Pool Mix

We want to compare different firing fractions f. With period length p we have an expected delay of

$$\frac{p}{2} + (1-f)p + (1-f)^2 p + ... + (1-f)^n p.$$

Using the limit of the geometric series sum to find the expected delay ED. [2]

$$ED = \frac{p}{f} - \frac{p}{2}$$

For a fixed expected delay $\mathcal{E}$, given a fraction f we can find the period length p.

$$p = \frac{\mathcal{E}}{\frac{1}{f} - \frac{1}{2}}$$

We will focus on the analysis of this firing strategy as under the assumption of the simulation (steady arrival rate) the Cortrell and binomial mix behave similarly. With concrete period lengths and firing fraction the variance can be calculated with the variance of the geometric distribution $Var(X) = \frac{1-f}{f^2}$ scaled up by the scalar of the period length. [3] Additionally one needs to take the arrival time distribution

[2]
$$\lim_{n\to\infty} \sum_{k=0}^{\infty} ar^k = \frac{a}{1-r}$$

[3] According to $Var(aX) = a^2 Var(X)$.

f	p	$Variance$
0.2	0.778	15.64
0.3	1.238	9.759
0.4	1.754	6.835
0.5	2.339	5.134
0.6	3.007	4.095
0.7	3.778	3.502
0.8	4.678	3.286
0.9	5.741	3.46

Figure 3.2: Period lengths and variances for fraction f with $E(D_\lambda) = 3.5085035$.

of a message within a period into consideration. This is assumed to be uniform from the start of the period until the end thus yielding a variance of $\frac{1}{12}p^2$. Since both are independent the total variance of the message delay is the sum of the variances:

$$Var(Delay) = \frac{1-f}{f^2} + \frac{1}{12}p^2$$

An overview can be seen in figure 3.2.[4] The results are mostly intuitive, as a lower fire rate leads to less consistency in the delay. From $f = 0.8$ upwards there is not quite obvious turning point at which the variance from the arrival moment distribution (which is increasing as p increases) outweighs the variance of the delay due to the firing.

Further we consider the expected pool size N_i at the time before firing round i. Every period, the expected number of messages arriving is pr and when firing the pool size is reduced by the factor f. Thus, we have

$$N_{i+1} = (N_i + pr) * (1 - f).$$

Again we apply the geometric series with a start term of pr and a common ratio of $1 - f$. Then the expected pool size approaches the sum of the series over time. We have

$$N = \lim_{i \to \infty} N_i = \frac{pr}{f}$$

as the expected pool size limit. Since we are concerned with mixes during their operation after running a reasonably long time we will use N directly for comparison for different mixes.

3.2.2 Timed Pool Mix

The constant timed pool mix will keep a constant number of messages m in the pool. The analysis is not very insightful on its own under the assumption of the model, but it is therefore able to show the shortcomings of the simulation. The pool of messages is kept to guarantee some anonymity in low traffic and changing traffic situations. According to our assumption the input traffic rate is constant. Thus, for an expected period length p we have an expected messages arriving in that time frame.

$$avg = r * p$$

Either avg is above the threshold m or below. If it is below, periods are skipped regularly, the mix will behave similarly to a mix with a period length of a multiple of p. The more normal case is that the threshold m is smaller than the expected number of messages each round. The constant timed pool mix is then reduced to a fraction timed pool mix with

$$f = 1 - \frac{m}{N}. \tag{3.3}$$

It should be noted that the average pool size is dependent on the input arrival rate. Thus we cannot directly compare mixes for different arrival rates with an equal period length like we did for the constant fraction timed pool mix in the previous section. Instead we would have to consider a different mix with a period length if we fix the minimum pool size. Thus we leave out a graph of this mix and refer to the previous section. A corresponding constant fraction mix can be found with equation 3.3.

[4] All numbers are rounded to 4 significant figures in this work.

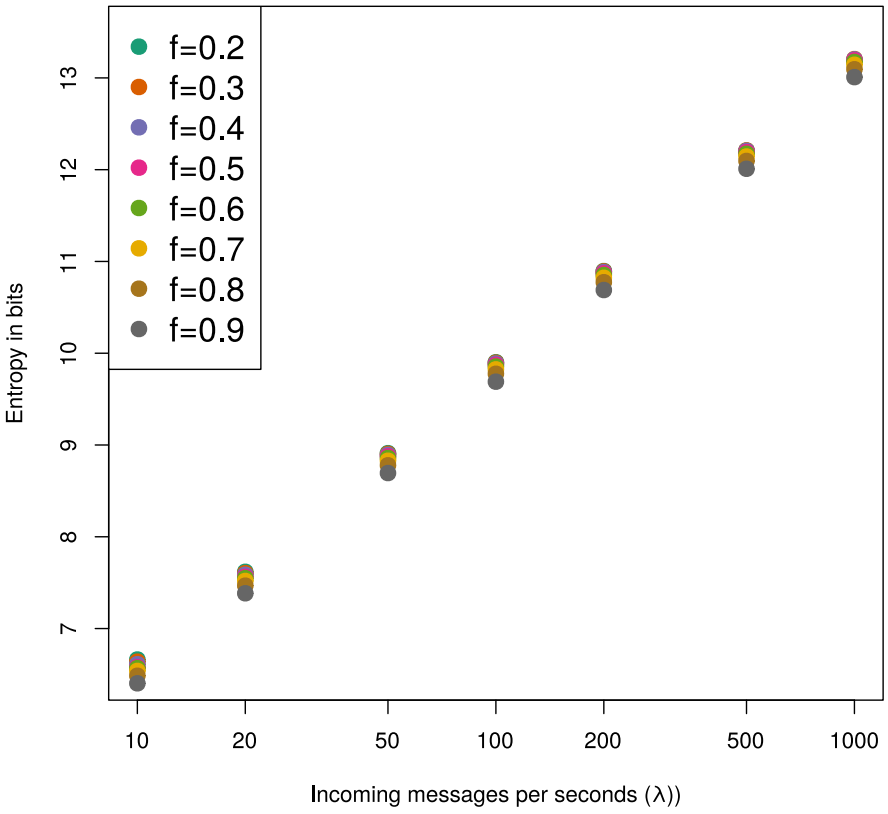

Figure 3.3: Entropy of timed pool mixes with varying period lengths with equal expected delay. Average over 1000 trials.

3.2.3 Dynamic Pool (Cottrell) mix

The dynamic pool or Cottrell mix is defined with both a fraction f and a threshold m. [SDS02] [5] If the pool is less than n, the mix does not fire, otherwise it fires the fraction f of messages. This f is respect to $n - m$, so instead of the constant fraction pool mix, the dynamic mix always stores m messages. Therefore, this mix strategy is a fusion of the two previous strategies. Similarly, we can reduce this mix for a fixed input arrival rate r to a constant fraction pool mix.

For this a formal definition about mix strategies in general is needed. Diaz et al. propose [DS03b]the generalized mix framework to define a mixing strategy by the function mapping the pool size to a fraction of messages to be fired from the mix. [6]

Definiton 3.4. *The generalized mix framework is determined by the function* $P : \mathbb{N} \to [0, 1]$. *At the time of firing with* n *messages in the pool, the fraction* $P(n)$ *of the pool will be output.*

We will use this approach to compare the dynamic pool mix to the constant fraction mix. [7]

Lemma 3.5. *The expected pool size* N *of a dynamic mix with threshold* m *and fraction* f *approaches* $m + \frac{pr}{f}$.

[5] The Cottrell mix was designed by Lance Cottrell for the Mixmaster protocol, but was not formally defined there. In fact in the Mixmaster protocol version 2 draft, the authors refer to the here cited paper for such a definition themselves. [MCPS04]

[6] Note that additionally to the function $P(n)$ the information about when the mix fires (i.e. $P(n)$ is executed) is needed.

[7] This only concerns the entropy-based anonymity analysis, Cottrell mixes can provide benefits apart from against a passive eavesdropping adversary, such as an active attack. See 4.3.

Proof. Again let N_i be the average pool size at the point of firing round i. Once N_i exceeds m it will never drop below m again because of the threshold and then it holds

$$N_{i+1} = (N_i + rp - m)f.$$

Let $i \geq \epsilon$ such that $N_i \geq m$ and let $\mathcal{N}_i$ be the corresponding fraction pool. (That is the pool size ignoring the constant m.) Then

$$\mathcal{N}_{i+1} = (\mathcal{N}_i + rp)f$$

and

$$\mathcal{N} = \lim_{i \to \infty} \mathcal{N}_i = \frac{pr}{f}$$

and for

$$N = m + \mathcal{N} = m + \frac{pr}{f}.$$

□

Theorem 3.6. *For each dynamic mix with fraction f_d and threshold m there exists a constant fraction mix with parameter f_c, s.t. $P(n)$ is equal under the generalized mix framework for the interarrival rate r.*

Proof. First we will show that there is a f_c with equal probability of each message leaving (i.e. the same $P(n)$) and then we will show that the expected pool size N_c for f_c is equal to N_d (i.e. equal n in the same scenario).

Under the generalized mix framework we have for the dynamic pool mix

$$P_d = (N_d - m)f$$

and for the constant fraction pool mix

$$P_c = N_c f_c.$$

Setting those equal:

$$(N_d - m)f_d = N_c f_c$$

Assume $N_c = N_d = N$:

$$(N - m)f_d = N f_c \qquad | : N$$

$$f_d - \frac{m f_d}{N} = f_c$$

$$f_c = \frac{(N_d - m)f_d}{N_d}$$

Now we show that for $f_c = \frac{(N_d - m)f_d}{N_d}$ the expected pool sizes are equal: $N_c = N_d$.

$$N_c = \frac{pr}{\frac{(N_d - m)f_d}{N_d}}$$

Substitute

$$N_d = m + \frac{pr}{f_d}$$

$$\implies N_c = \frac{pr}{\frac{pr}{N_d}} = N_d$$

□

Corollary 3.7. *The average expected delay and its variance of a message in a constant fraction mix with parameter f_c and a corresponding dynamic mix with parameter f_d and m are equal for the interarrival rate r.*

We will compare the simulation and theorem 3.6 result for $m = 100$, $f_d = 0.7$ and $p = 3$:

r	N_d	f_c	Entropy d	Entropy c
10	142.9	0.21	8.45	8.466
20	185.7	0.323	8.727	8.733
50	314.3	0.484	9.327	9.302
100	528.6	0.568	9.971	9.972
200	957.1	0.629	10.75	10.75
500	2242.9	0.669	11.92	11.92
1000	4385.7	0.684	12.867	12.86

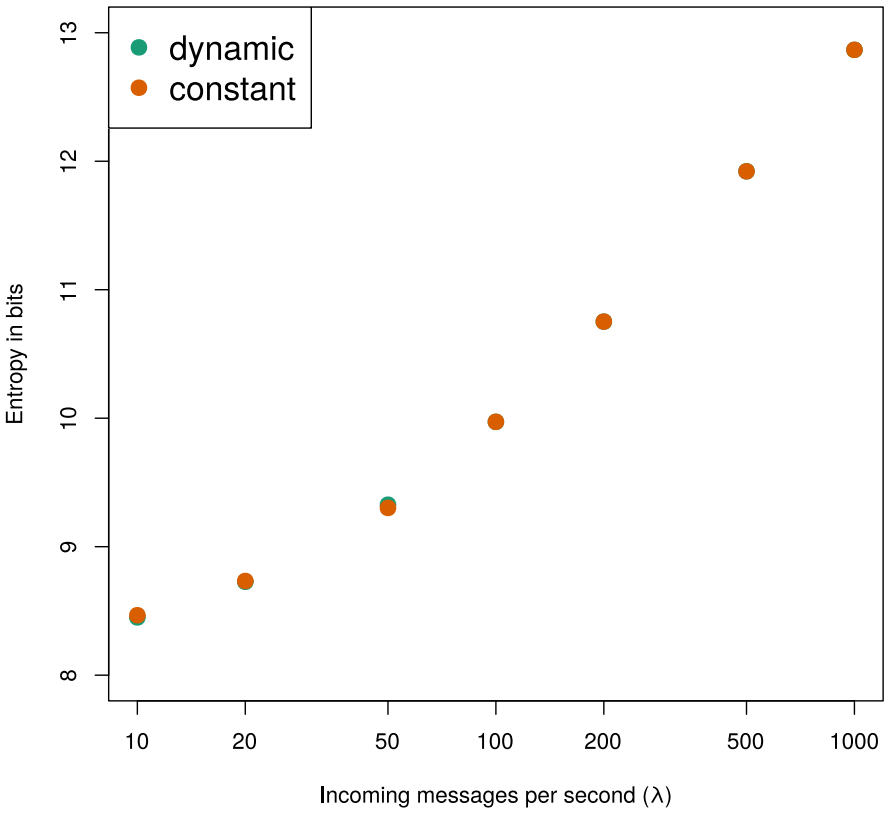

Figure 3.4: Comparison of the dynamic pool and its corresponding constant pool. Average over 1000 trials

Conclusion

Theorem 3.6 and corollary 3.7 show that under the assumptions of the model (i.e. the input follows a Poisson process) the dynamic and the constant fraction mix strategy are equivalent for chosen parameters. Therefore, if one seeks to find a superior mixing strategy, the classic expected delay to entropy tradeoff will not rule out either strategy. For this decision more involved attacks need to be considered as in chapter 4.

3.2.4 Expected Delay of Different Mixes

We can generalize the corollary 3.7 to a statement about expected delay for an expected pool size. Although expected pool size and expected delay have been studied extensively, the simple relation has not been stated to our knowledge.

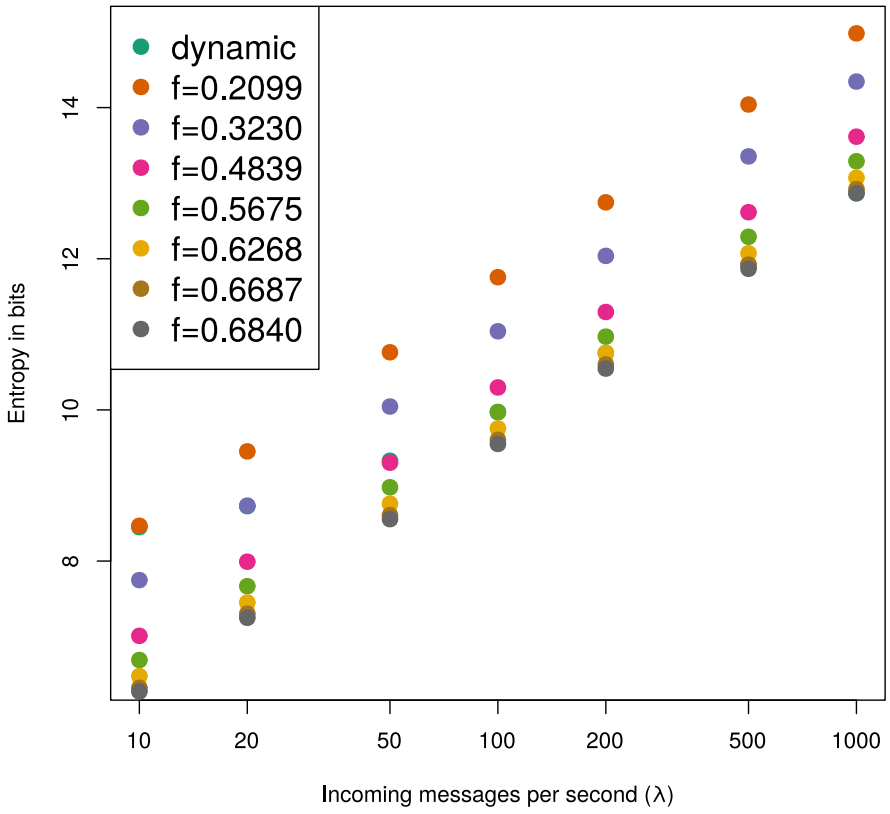

Figure 3.5: A comparison how equivalent dynamic and constant mixes behave under different interarrival rates. Average over 1000 trials.

Theorem 3.8. *For any (non-continuous) mix that has a constant expected pool size limit N (i.e. N does not diverge) for the interarrival rate r with period length p the expected delay ED_N is*

$$ED_N = \frac{p}{2} + p * \frac{N - pr}{N} + p * (\frac{N - pr}{N})^2 + ... = \frac{p}{1 - \frac{N-pr}{N}} - \frac{p}{2}$$

with variance

$$Var(D) = p * \frac{1 - \frac{N-pr}{N}}{\frac{N-pr}{N}^2} + \frac{1}{12}p^2$$

Proof. The key insight for this is that the messages in the pool at the time of firing are all equal for the flushing algorithm. That is for any two messages m_1 and m_2 the probability of them not being flushed is equal: $P(1) = P(2) = P(staying) = 1 - P(n)$. Additionally, since N is a constant the expected number of messages flushed need to be the expected incoming messages (i.e. pr) Therefore the expected $P(staying)$ for every message will always be $\frac{N-pr}{N}$. The delay is then the (shifted) geometric distribution with expectation $ED_N = \frac{p}{1-\frac{N-pr}{N}} - \frac{p}{2}$.

Furthermore, the variance of the delay is that of the underlying geometric distribution

$$Var(D) = p * \frac{1 - \frac{N-pr}{N}}{(\frac{N-pr}{N})^2}$$

summed with the variance of the uniform distribution $\frac{1}{12}p^2$ within the period length p. □

The only constraint of theorem 3.8 that N converges is less severe than it might seem from a theoretical viewpoint. For N to converge under a Poisson process-modelled input, it suffices that the expected incoming and outgoing message rates are equal. This is generally a desirable property to ensure functioning of a real-world mix (so it does not run out of memory and drop messages regularly).

3.2.5 Binomial Mix

The Binomial mix uses the firing strategy $P(n)$ at the end of every period. Instead of flushing $P(n)$ messages it uses the binomial distribution to determine the messages that are flushed. For each message in the pool it draws with $p = P(n)$ and includes that message in the flush accordingly. The literature often uses a normal distribution for $P(n)$. [DS03b] [Ser07]

Clearly the *binomial method* (i.e. treating $P(n)$ as a probability, not a fraction) does not change the expected pool size and the expected delay. [8] Unfortunately, theorem 3.8 assumes that the fraction $P(n)$ is flushed deterministically. In fact, if we combine the expected delay and the draw from the binomial distribution, which are independent, we receive a different variance. Additionally the variance of each binomial draw will be $N * f * (1 - f)$ where f is the expected number of messages to be fired. [9] Note that the variance is dependent on pool size and thus the interarrival rate, making an analytical comparison overly complex.

We will focus our attention on the binomial method used on otherwise regular constant fraction timed mixes for comparison. Figure 3.6 shows two of each mixes under different conditions with equal expected delay and expected pool size. The binomial mix performs similar enough with respect to the provided entropy to warrant the assertion that the additional variance has no significant effect anonymity-wise.

r	Timed 0.5	Binomial 0.5	Timed 0.7	Binomial 0.7
10	6.359	6.320	5.655	5.58
20	7.34	7.322	6.618	6.581
50	8.651	8.643	7.918	7.902
100	9.647	9.644	8.91	8.903
200	10.65	10.64	9.907	9.901
500	11.97	11.97	11.23	11.23
1000	12.97	12.97	12.23	12.22

Figure 3.6: Comparison of entropy for the constant fraction timed pool mix with $f = 0.5$ and $f = 0.7$ and its corresponding binomial mix. Simulated 100000 input messages, averaged over 100 trials.

Binomial+ Mix

The Binomial+ mix as introduced by Serjantov et al. [Ser07] proposes to use not one normal distribution for $P(n)$, but a family of normal distributions dependent on the pool size. The intention is to further hide the current pool size to an observer, but the properties that we study in this chapter remain the same. That is, under a constant arrival rate, the pool size will still converge to some $N \in \mathbb{R}_+$ and therefore the previous analysis is still valid. [10]

3.2.6 Continuous mix

For the analysis of the continuous mix we will restrict ourselves to the optimal exponential mix. Since the firing of messages does not happen at set intervals, an adaptation of the model is needed. To overcome this problem the simulation will sample two Poisson processes, one for the input as used in the previous

[8] While in theory the expected number of messages to be flushed should be untouched, the edge cases that the binomial distribution is cut off at 0 messages as a minimum and the current pool size as a maximum is unusual enough to be ignored.

[9] The variance of two independent variables X and Y is

$$Var(XY) = Var(X)Var(Y) + Var(X)(E(Y))^2 + Var(Y)(E(X))^2.$$

[10] We do not dispute the advantages of the Binomial+ generally, but it does not require a seperate analysis under the constant traffic assumption.

models, and additionally one for the outputs. Then for each sampled output the probability distribution over the input messages is computed. For this, the probability of each message surviving that long is computed using the cumulative distribution function of the exponential distribution. According to the construction of the exponential mix, this is the probability of the message m being delayed for at least the interval from $arrival_m$ [11] to the currently sampled output time. Since the output time is fixed and exactly one output is guaranteed, its probability distribution is then normalized.

The source code of this simulation is shown in figure 3.2.6.

```
#the exponential mix delays each input according to the
#exponential distribution with parameter lambda
#lambda <- 0.285021805 #=> delay: 3.5085035
exponentialmix <- function(numberofoutputs, inputarrivalrate, lambda) {
outputstoskip<-numberofoutputs #entropy will not be taken into account
numberofoutputs<-numberofoutputs+outputstoskip
numberofinputs<-numberofoutputs*3 #3x to capture in the middle
inputs<-rexp(numberofinputs,inputarrivalrate)
outputs<-rexp(numberofoutputs,inputarrivalrate)
inputtimes<-cumsum(inputs)
outputtimes<-cumsum(outputs)
probs<-vector(length=numberofoutputs)
entropy<-vector(length=numberofoutputs)
delays<-vector(length=numberofoutputs)
avgmessages<-0
entropycount<-1
j<-1
for(i in outputstoskip:numberofoutputs) {
  while(inputtimes[j]<outputtimes[i]){ j<-j+1 }
  #prob that input hasn't been evicted by firing
  probs = 1-pexp(outputtimes[i]-inputtimes[1:(j-1)], lambda)
  normalizedprobs<-probs[1:(j-1)]/sum(probs[1:(j-1)])
  normalizedprobs<-normalizedprobs[normalizedprobs != 0]
  #remove all zeros for entropy calc, none of them should be zero
  #but they are because of floating point precision
  entropy[entropycount]<--sum(normalizedprobs*log2(normalizedprobs))
  currentavgdelay <- sum(probs*(outputtimes[i]-inputtimes[1:(j-1)]))
  currentavgdelay <- currentavgdelay / sum(probs[1:(j-1)])
  avgmessages<-avgmessages+sum(probs[1:(j-1)])/(numberofoutputs-
     outputstoskip)
  delays[entropycount] <- currentavgdelay
  entropycount<-entropycount+1
}
avgentropy<-sum(entropy[1:entropycount])/(entropycount)
avgdelay<-sum(delays[1:entropycount])/(entropycount)
#print(sprintf("Average entropy: %f, Expected delay: %f Average delay: %
    f Average messages in pool: %f Inputs Processed: %d, Total Outputs:
    %d", avgentropy, 1/lambda, avgdelay, avgmessages, j, numberofoutputs
    ))
return(avgentropy)
}
```

Finally, we can use this simulation to compare the exponential mix to other mix designs, as we could

[11] The arrival time of m.

reduce the previous designs to a simple constant fraction timed mix under the assumption of a constant traffic rate r.

To make a concrete comparison we use the exponential distribution with our previously chosen $\lambda = 0.2850218$ with $ED_\lambda = 1/\lambda = 3.5085035$ and the variance of the exponential distribution $\frac{1}{\lambda^2} = 12.3096$. We can find a constant fraction timed mix with equal ED and $Var(D)$ using the previously given formulas and get $f = 0.2465$ and $p = 0.986729$. [12] Further, we include another constant fraction timed mix with $f = 0.5$ and $p = 2.339002$ in the comparison, which has the same ED but $Var(D) = 5.133916$.

r	*Entropy Exponential*	*Entropy Timed*	*Entropy Timed* 0.5
10	6.574	6.66	6.58
20	7.573	7.623	7.563
50	8.89	8.909	8.876
100	9.881	9.9	9.87
200	10.86	10.89	10.87
500	12.14	12.21	12.19
1000	13.05	13.2	13.19

Figure 3.7: Comparison of entropy for the exponential mix and timed mix with equal ED and equal variance and a timed mix with $f = 0.5$ but different variance. Simulation for 100000 inputs, average over 100 trials.

Looking at figure 3.7, delay variance seems to have little influence, if at all, on expected entropy. [13] [14] In fact comparing different results with many simulations, the relation from entropy and expected delay was surpisingly constant. We found that only in cases with extreme parameters, where the simulation behaved otherwise abnormally (e.g. the rate is too low to fulfill a proper average of messages fired), there was a relation between expected delay and average entropy that did not deviate significantly and other factors did not have any influence. A sound reasoning or even a closed form for this relation seems hard to find, but to document our results formally we found that:

$$\frac{2^{Entropy}}{ED * r} \approx 2.7.$$

3.3 Conclusion

3.3.1 Shortcomings of the Model Assumptions

The most obvious shortcoming is the one already extensively discussed problem of the Poisson process. The differences of mixes vanished under the assumption of a steady arrival rate. Clearly the mixes behave very differently under changing traffic conditions. Consider that under lower traffic, the exponential mix will sacrifice entropy (since the exponential distribution parameter isn't changed) but the pool mixes (that fire some fraction) will naturally throttle their throughput.

Another shortcoming, which we consider less severe, is that pool size is unlimited and messages are never dropped. The first one isn't a problem with an assumed steady arrival rate but might be in the real world. The latter is definitely an oversight. Any network component will have to drop messages, under some conditions even a nonsignificant amount. This was not taken into consideration for the model. Related to this, also the parallelization and computational power is unlimited in the model. For the exponential mix, which has spread out inputs and outputs, this is only problematic if many hit the same moment. But for pool mixes in theory all messages leaving the round should be fired at the same time and therefore reach their next hop at the same time. If the next hop is another mix they should be counted as coming in at the same time even if spread out. This would require a high degree of synchronization which is definitely an engineering challenge.

[12] Note that the expected delay and its variance do not uniquely identify one such mix. In fact, there is another one with a very large p and f close to 1. We have chosen this one to not have significant floating point precision errors.

[13] At higher rates the exponential mix does run out of inputs that contribute to delay and entropy. Its actual expected delay to entropy relation at $r = 500$ and $r = 1000$ was still equal.

[14] The error at lower rates where timed mixes seem to yield higher entropy are because the fraction of messages per round is rounded down and therefore $f = 0.2466$ will only output 2 most of the time for $p * r \approx 10$.

3.3.2 Shortcomings of the Results

For comparison, we fixed the expected delay and sometimes the variance to match the Instant Messaging use case. Thus strictly speaking, the results are only applicable for these parameters. While we were not able to find a proof for our average entropy to expected delay relation, the fact that it remained true throughout our experiments indicates that the IM parameters have no major influence on the results. Therefore, the results should translate to mix networks with all use cases easily, although they are out of scope for this work.

A reasoning for the average entropy to expected delay relation seems unlikely to be found easily because of the construction of entropy and the exponential distribution.

Precision

As mentioned in section 3.2.6, the calculated expected values will differ from the simulation if r or the number of inputs simulated is too low. For example, $f = \frac{3}{4}$ will behave almost like $f = 0.7$ for $r = 10$. To mitigate overseeing such errors we additionally implemented calculations for the actual delay of the simulation to compare to the expected delay. If these do not match, the result will be skewed, too.

Additionally, it should be ensured that probabilities added together matches expected pool size throughout the simulation roughly. For this we also implemented calculations for the average pool sizes. Mismatches here arise from floating-point precision elimininating some probabilities, but those errors were not a problem in our results.

If one wishes to replicate the results, these countermeasures should be taken into consideration.

Continuous Mix

The continuous mixes fundamentally rely on a steady arrival rate. Thus, it can be argued that this mix is in some way favored by the results, as the continuous mix is highly vulnerable to non-steady arrival. Even worse there is no analysis for such scenarios, so a continuous mix cannot be thought of as providing any anonymity in those scenarios. Even for Poisson distributed input the anonymity provided is probabilistic, that is, a message can still leave the mix unmixed (if no other messages arrived or are residing in the mix currently). A pool mix will guarantee some anonymity even in those cases.

3.3.3 Assessment of the Results

The comparison of strategies did not favor any design, in fact it provided some evidence that there is very little difference, if at all. Overall, the anonymity provided is most likely good enough even for lower traffic conditions. While 8 bits of entropy might not seem much from a cryptographer's perspective, in terms of anonymity this can be seen as good. [15] Therefore, anonymity on its most basic terms should not be the obstacle for an Instant Messaging deployment. The results of this chapter provide some evidence that entropy-based anonymity will not be the weakest link in providing anonymous communication through mix networks.

[15] The important difference is that in cryptography the adversary has a way to check their guess relatively efficiently. There is no technical way to check if the message was sent out by some sender that the adversary guesses.

Chapter 4

Attacks & Defenses

We summarize common attacks found in ACNs, their proposed defenses and their applicability to a mix network for the IM case. Here we are only concerned with the research on active attacks, since passive attacks lead to statistical disclosure, end-to-end confirmation and others. But to defend these is the mix network's main goal and therefore the defense is the design of the mix network. We review passive attacks more thoroughly in section 5.3.

First, we describe two practical attacks and then move on to the $(n-1)$ attack, which covers many attack scenarios because the adversary is given strong capabilities.

4.1 Replay Attack

The replay attack tries to reinsert a previously recorded or otherwise known message into the mix which processes it as a legit message. While this attack is similar to the replay attack often described in cryptographic authentication scenarios it has vastly different implications. The adversary's goal is to be able to follow the path by repeatedly letting the message be routed through the network and therefore revealing its designated receiver. Tracing the message is thought of through statistically inferring the path, as it becomes much easier if multiple sets of data (through multiple replays) were available to the adversary.

The adversary therefore has the power of a GPA, additionally can be considered to have some (but not all) of the mixes compromised and has gained access to a message. [1]

While a straightforward defense might be including some allowed time interval in the routing information when this message should arrive at the mix, this poses practical problems. Firstly, this time might not be known and would have to include some tolerance (see section 4.3.1), during which the replay attack can be successfully performed. Further, mix networks allow for so-called reply blocks, pre-encrypted packet headers including routing information, that is supplied with a message. A receiver can then append an answer message to the packet header and send this to the first mix, which is the only routing information available to them. The reply block can then be routed through to the original sender without the receiver knowing where it was sent to. [2]

Scenarios in which one of the communicating parties is in control of the adversary is referred to as an adversarial Bob. The replay attack can be performed by such a Bob, who can resend the reply block. Replies are asynchronous by nature and their sending and therefore intermediate arrival times cannot be known beforehand. Defenses that do not impose restrictions on the mix network application therefore cannot use the often applied method of timestamps or windows alone to counter the replay attack in authentication scenarios.

[1] Controlling the first mix in the message's path is one realistic scenario of how an adversary gets ahold of such a message of interest. The first mix knows the senders and the message associated with them, but would want to know whom it is routed to.

[2] More indepth explanation about reply blocks can be found in the mix network packet format research. [DG09]

4.1.1 Defenses

Mix Link Encryption

Mixes can encrypt the messages exchanged between other mixes. Having the mixes encrypt their communication does not close the attack vector, but requires the adversary to be an adversarial Bob or be in control of the first mix. A GPA itself cannot gain access to a valid message from a sender by eavesdropping.

This simple defense can be implemented regardless of the delay requirements.

Replay Cache

A sound solution will have to implement a replay cache which determines if a message has been seen already and drop it accordingly. Such a lookup can be implemented efficiently with a Bloomfilter so that memory does not run out.

Even with a Bloomfilter, messages have to be retired from the lookup at some point. [3] Mixmaster deleted several days old message from the replay cache. [DDM03] Of course, such an approach is insufficient, as replay attacks with the same message can be launched several days apart.

Expiring Messages

Combining a replay cache with expiring messages closes the attack vector for replay attacks. Messages must include an indication of what time interval it is designated for. If the possible time intervals are synchronized network-globally, the mixes can keep a seperate replay cache for each interval and delete old caches when they are considered to be expired.

Modern mix network implementations such as Katzenpost have network-wide *epochs* with mixes having a seperate public key for each epoch. Messages are encrypted with a public key for that epoch and the mix only keeps the current and the previous epochs key pair and replay cache (see section 4.2.1). [Proa]

For IM messages can be expired sooner, as messages that have stayed in the network for too long, are considered as dropped anyway.

4.2 Compulsion Attack

Another attack that is more of practical nature of operating mix nodes in the real world is the *Compulsion Attack*. While traditionally mix nodes are considered to be either honest or dishonest, colluding or non-colluding with an attacker, honest mix nodes might be (legally or otherwise) forced to hand over data about their operation after it happened. This can include decrypting individual messages or disclosing private keys.

The adversary can have the capabilities of a GPA, have some of the mixes collude and can be an adversarial Bob (i.e. be sender or receiver) and tries to find the origin or destination of a message of interest. Additionally the adversary can force mixes to decrypt a message or even hand over all keying material it possesses at the time of compulsion. This compulsion is considered to be expensive and takes some time until the request is fulfilled by the mix. Therefore, the adversary needs to narrow down their selection of mixes for the attack to be feasible. [4]

4.2.1 Defenses

Mix Link Encryption

The mixes can transmit messages with (forward-secure) link encryption. This does not close the attack vector but rules out the purely GPA scenario. With link encryption in place, a message of interest cannot

[3] A Bloomfilter running over a long time will approach always outputting an already seen message.

[4] If such a compulsion was not expensive, the adversary could target them all at the same time. Real-world scenarios justify this cost by a legal authority either needing a court order, locating a potential mix provider and perhaps confiscating their equipment.

be aquired through passively observing the network links, therefore the adversary needs to find another way of finding a message they aim to trace.

The adversary then needs to have a mix collude that is the first hop, or be a communication partner and having received a valid reply block for a message.

Forward-secure mix link encryption is compatible with IM constraints.

Rotating Mix Keys

As introduced in section 4.1, mixes can restrict messages to intervals in which they are valid. To enforce this, mixes can rotate their public key under which the messages must be encrypted for each interval. Intervals that are in the past and its messages should no longer be accepted, thus don't have to be decrypted by a mix. Therefore the mix can delete the keying material once it considers an interval to be expired.

The compulsion attack can only force revelation of keying material that the mixes possess at the time of compulsion. This limits the attack window for the adversary, as they do not know which mix is the next in the path until the predecessor has decrypted the message, they cannot attack them in parallel. For a standard 3-hop path the adversary then has to perform 3 compulsion attacks during the time frame in which the intervals keys have not been deleted by the mixes. Up to the point of compulsion mixes are considered honest, that is, they safely delete their keying material.

In an IM setting where the delivery is expected within a minute, intervals can be set accordingly. If the compulsion attack is a priority to defend against, the mixes can delete the corresponding keys after a few minutes. This will make the compulsion attack arguably impossible to perform in practice.

Multicast Routing

Danezis et al. propose a multicast routing solution in which not one mix is included in the routing information as the next hop but several. [DC05] But the message is encrypted with only one of the mixes public key, the rest cannot decrypt the message and process the message, therefore discard it. [5] Where a multicast is given at the routing information, the adversary will have to launch the compulsion attack at all the potential mixes (or guess one with a chance of failure), thus either making the attack more expensive or less likely to succeed.

Multicast routing does introduce computational and bandwidth overhead by transmitting and decrypting messages that will not be further processed at the mix. The overhead is not worse for an IM use case, but such a mix network might have tight bandwidth limits to begin with.

Forward Secure Mixes

One more involved approach at countering a compulsion attack has been proposed by Danezis. [Dan02] The idea is to also cycle through keys (symmetric to be precise) for each user-mix pair, similar to the idea of a ratchet in the well-known Double-Ratchet Algorithm. Danezis argues that the number of mixes is small enough for this to add not too much (memory) overhead at the mixes. While that may be true, depending on the setup, another problem we identified is that mixes will be able to link packets coming from the same user as they make use of the same chain of keys. The mix and the user know their current key as the message arrives, but the mix will delete the old key as soon as it is used, it is therefore not able to comply to compulsion attacks once it processed the message. [6]

Apart from the memory overhead that grows with the size of the network, for IM specifically a problem is out-of-order delivery of messages. Due to the high frequency of messages and their variance in delay a mix can receive a message which the ratchet has not advanced to yet, because two messages arrive in a different order than they were sent in. The mix can either save messages it cannot decrypt (yet) which results in memory overhead or discard such a packet. If out-of-order delivery is common enough discarding such packets would lead to a high packet loss.

[5] An additional burden to the adversary is that mixes cannot prove they do not know the keying material. For simplicity we leave this scenario of a lying compulsion attack out and consider all of them cooperating with the adversary after the compulsion.

[6] Note that this does not close the attack vector for an adversarial Bob, where the compulsion attack starts before the message has been processed.

4.2.2 Conclusion

The first two defenses limit the attack scenarios and the time at which it must be performed. They occur at little additional cost for the mix network.

For multicast routing and forward secure mixes a tradeoff must be made. Both countermeasures are likely to have negative effect on the functioning of the network. In practice limiting the interval to minutes or hours is likely to eliminate the compulsion attack.

4.3 $(n-1)$ Attack

An attacker performing the $(n-1)$ attack considers a particular mix M and a desired incoming message m. The goal of the attack is to be able to tell which of the outgoing messages corresponds to m. Ideally, the attacker can get the mix to shuffle m with $(n-1)$ of the attacker's messages. Then the outgoing message which is not recognized by the attacker must be m. The attack can be achieved by dropping or delaying all other legitimate messages to M and inserting the attacker's messages. If the mix is a pool mix the attacker first needs to *flush* the pool of legitimate messages by delaying legitimate messages and inserting their own. If the mix is an exponential mix in it's pure form (one that will continue operation even in changing traffic conditions) the attacker has no reason to insert own messages, it suffices to delay or drop the messages routed to the mix under attack. Generally, the adversary does need the ability to delay or drop and insert messages to launch this attack. Furthermore, the adversary needs to be able to observe all network links, thus making it a strictly stronger adversary than a GPA.

There are many more caveats to this attack depending on the particular mix network design.

4.3.1 Defenses

Defenses against the attack can be categorized as attack prevention and attack detection. Since the attack is active detecting an ongoing attack should also prevent or at least drastically decrease the probability of the attack's success. If an active attack is detected and the defense does not specify otherwise, the mix simply stops forwarding messages and effectively shuts down. This theoretically turns the $(n-1)$ attack from a deanonymization to a Denial-of-Service attack, but since the attacker is usually given the ability to delay arbitrary messages already, the attack then is of no advantage.

Link Encryption

A practical defense is link encryption, although it does not close the attack vector it makes the attack harder to perform. The attacker will not recognize their own messages when they are only transmitted through link encryption (to the next mix). Thus, for the $(n-1)$ attack to succeed the attacker will have to link messages at every hop, either by performing the $(n-1)$ attack or having the mix collude. There is no good reason not to implement this countermeasure. It also shows how the attack which has been studied extensively in the literature will be harder to perform in practice than it seems.

Time Windows

The stop-and-go mix design paper by Kesdogan et al. [KEB98] proposes time windows. For each message two timestamps TS^{min} and TS^{max} are provided to the mix. If the arrival time of the message is not in this time window, the mix can assume the message has been delayed and should drop it. The active attack is then detected and unsuccessful.

The timestamps of arrival at the mix are calculated by the sender. The calculation considers the following parameters:

1. syn : maximum clock deviation of two clocks
2. t_S : local time of the sender
3. T_i : delay time at mix i
4. d_{ij} : unidirectional transmission delays

It then calculates the timestamps for mix i as follows:

$$TS_i^{min} = t_S + \sum_{j=1}^{i-1} T_j + \sum_{j=1}^{i} d_{j-1,j}^{min} - syn$$

$$TS_i^{max} = t_S + \sum_{j=1}^{i-1} T_j + \sum_{j=1}^{i} d_{j-1,j}^{max} + syn.$$

For the defense we are interested in the length of the time window:

$$\Delta t = TS_i^{max} - TS_i^{min} = 2syn + \sum_{j=1}^{i} \Delta d_{j-1,j}.$$

According to the authors the window should be smaller than the time it takes the attacker to empty the mix. Thus, the target message cannot be delayed long enough for the mix to be empty without the mix noticing. The authors already acknowledge the weakness that the more nodes are chained along a path, the longer the window gets.

Unfortunately, we consider this defense to be unpractical since the clock deviation greatly outweighs the delays for any real-world network.

For example the Tor network, which requires the time synchronization service NTP, gives clients a tolerance of 30 minutes. [7]

```
/** How far in the future do we allow a directory server to tell us it
    is
* before deciding that one of us has the wrong time? */
#define ALLOW_DIRECTORY_TIME_SKEW (30*60)
```

While this restriction is surely not strict, it illustrates how far off the estimation of the authors about clock skew is. [8]

Detours

One of the earliest defenses was proposed by Gülcü et al. [GT96] A mix handling a message can decide, instead of routing to the next hop directly, to introduce a detour. That is, routing the message through the mix network itself to the next hop. The sender will have no control whether a detour is taken or what its path is. Therefore, an adversary that floods the mix with messages will have some fractions of messages detoured by the mix under attack, making them not recognizable to the adversary. The target message which is not recognized by the adversary is then indistuingishable from the detours of their own message.

For this defense to work the chance of detouring must be high enough. Detours will clearly influence the delay of the message, thus if detours are implemented for IM, the parameters need to be adjusted accordingly. Otherwise inter-mix detours are an applicable defense even under latency requirements.

Regroup-and-Go Mixes

Another proposal to counter $(n-1)$ attacks was made by Shi et al. [SFS06] Here the sender will wait for several messages to the same receiver have accumulated. [9] For each hop on the mix the messages are then split up into groups of at least two. The grouping information is then encrypted together with the routing information for that mix. The mix will only forward the messages if all of its group have

[7] Tor Project, tor main repository:src/feature/dirclient/dirclient.c

[8] Note that clock skew refers to inside a machine different components and on a network different nodes. We are only concerned with the latter, although the first has been shown to have implications on ACNs too. [Mur06]

[9] In the later paper they describe a file transmission where the file (i.e. the message) is split up into multiple messages. [SF08] The additional delay of waiting for messages to accumulate would likely disqualify this method, but we overlook this, since any message could be split up like that or dummy messages can be added

arrived. The authors argue then, that since the attacker cannot let many other messages into the mix, they would have to guess exactly the group structure at this mix. This is of course unlikely for even a small number of messages.

Several weaknesses are outlined in the paper itself, like having an avalanche effect where packet loss is amplified or introducing more state into the mix would lead to a Denial-of-Service vulnerability, where incomplete groups are sent to mixes. Another weakness we identified is the increased potential for correlation, instead of having just one message going along a path through mixes, there are several along the same path. This traffic pattern will stand out significantly from a passive observer. Additionally, the groups use relative timestamps to detect an ongoing attack, therefore likely suffering from a similar unusability as time windows described earlier.

The argument illustrates well something overlooked in research about the $(n-1)$ attack. The attack targets a single message, usually assumed to be along its path through the mix network. But for an adversary to target a single message it needs to be interesting or different in some way. Since messages inflight are equally-sized and non-decryptable (effectively random) to anyone but the receiver, the only interesting information must be the sender it originated from. To be able to know the origin of a message the $(n-1)$ attack must have started on the first hop (and then later repeated along the path). Therefore, an adversary can isolate all messages packaged together, as they all follow the same path reducing the defense to nothing.

RGB Mixes

The Red-Green-Black mix (RGB mix) uses a little amount of dummy traffic to detect when it is potentially under attack and then increases its dummy traffic accordingly. [DS03a] More precisely the mix routinely send green messages (so called heartbeat messages) that are normal mix network messages but their destination is the sending mix itself. The rate at which the mix sends is adjusted to the expected real traffic load of the mix. Recognizing the messages destined for the mix by itself, it knows what fraction of messages are currently the heartbeat messages. This achieves two things. First, the adversary cannot distinguish the mix's red traffic from real traffic, thus either needs to guess which traffic not to delay, or delay all of it. Secondly, if traffic is delayed, the mix will realize the fraction of red traffic dropping, since the adversary can insert real-seeming generic traffic (black traffic) but cannot deceive the mix to recognize them as their own (red traffic).

The mix then knows that it is either under attack or the traffic load is changing. As to not disrupt the functioning of the mix when a false alarm is given due to normal traffic changes the mix will not stop its operation, but add many additional dummy messages (terminating anywhere) in the output until the fraction of incoming red traffic recovers to normal. The adversary then cannot distinguish their targeted message from the green messages generated by the mix. RGB mixes constitute a sophisticated dummy traffic strategy in which overhead is low at normal operation but security is still provided under attack.

RGB mixes can be reconciled with IM requirements if the parameters are adjusted to small intervals. Smaller intervals can lead to more traffic load changes depending on the scenario. [10] Therefore, there will be bandwidth and computational overhead due to generating dummy traffic.

Loops

In the same vein as RGB mixes, Piotrowska et al. propose Loopix, a complete mix network design that is based on dummy traffic loops. [PHE+17]

The work describes several types of dummy traffic, the one interesting for $(n-1)$ attacks is called loop traffic. It generates loop messages Poisson distributed with some parameter λ_L which can be changed according to traffic conditions. One technical difference to RGB mixes is in routing, since mixes are set up in a stratified topology they might not be able to forward messages to another mix due to them being in the last position. The loop message will take its path to the layers and end in a provider, which is then asked to forward this message back to the mix. The addressed provider thus can recognize the message as a loop and the mix it originated from. They claim this does not open up a substantial attack vector.

[10] The only scenario where this is not the case is the input being a Poisson process as described earlier.

Mixing strategy design

Protection against a potential $(n-1)$ attack can also be gained by carefully designing the mixing strategy as already seen with the binomial mix which aims to hide the number of messages residing in the pool thus.

O'Connor has done more indepth research into behavior of mixes under a $(n-1)$ attack and derives strategies of choosing parameters to optimize protection.[O'C05] Since this has no direct relevance to the IM use case, we will not review this work. One of the additional countermeasures proposed is a maximum number of messages accepted per interval to increase the length of the attack.

Adversary Discussion

The defenses were briefly discussed and their applicability with IM requirements evaluated. For a more indepth discussion a finer adversary model is needed. The research in the literature is limited to a global active adversary that can insert, delay and drop messages arbitrarily. Against such an adversary defenses are hard to evaluate. Therefore, we could only check their applicability and if their security argument is sound. None of the defenses close the attack vector completely, but focus on making the attack harder to perform. For this to be evaluated properly an almost almighty adversary does not seem helpful. One intuitive way of weakening such an adversary is by associating their actions with some cost, then the defenses could be compared how much more costly they make the attack. Unfortunately such an adversary has not been proposed and formalized, which could help extending the attempt of this chapter.

Chapter 5

Constraints of Anonymity

In this chapter properties of anonymous systems are studied. We introduce selected topics of research and its relation to Instant Messaging. More theoretical aspects are explored, most of which do not constrict itself to mix networks but anonymous communication networks in general.

5.1 Unobservability

The focus of study in ACN literature is often (Message-)Sender-Unlinkability like introduced in section 1.2.4. Another privacy notion that has also been of academic and practical interest is that of *Unobservability*, that is informally speaking, can it be observed when a user performs some action (e.g. sends a message).

5.1.1 Definition

The canonical definition for unobservability comes from Pfitzmann et al. [PH00]

> Sender unobservability then means that it is sufficiently undetectable whether any sender within the unobservability set sends. Sender unobservability is perfect iff it is completely undetectable whether any sender within the unobservability set sends.
>
> Recipient unobservability then means that it is sufficiently undetectable whether any recipient within the unobservability set receives. Recipient unobservability is perfect iff it is completely undetectable whether any recipient within the unobservability set receives.

More formal systems of privacy notions, including sender and receiver unobservability have been proposed by Kuhn et al. [KBS+19]

Since scenarios in which achieving one but not the other is beneficial are hard to imagine, the term unobservability usually refers to both. An important distinction is that unobservability does not include undetectability of the containment in the unobservability set. That is, the unobservability set is considered to be known in full detail to an adversary, whether any user is part of that set is not hidden.[1]

5.1.2 Implementation of Sender Unobservability

Dummy Traffic and Constraining Throughput

Achieving sender unobservability is intuitive, some interval is defined in which messages are expected to be sent. If there is no real message to be sent, the client needs to generate an indistinguishable message to the adversary. Therefore, the dummy message must also traverse through the mix network and only after that can be discarded at some destination. As messages that cannot be decrypted at the receiver

[1] This was touched upon in the introduction, as this would be equal to hiding the usage of an anonymous communication network altogether.

are often discarded without error, many systems create a dummy message to a random recipient that either cannot be decrypted or contains some flag indicating that it is a dummy message to the receiver.[2]

This design must also restrict sending multiple messages during such an interval, as this would be revealed. Therefore, the Throughput is limited to a predefined number of messages per predefined interval.

Loopix

Loopix does not use a fixed interval, but an exponential distribution in which messages are sent out. This is not fundamentally different, but an adversary tackling unobservability would have to decide if the pattern of outgoing messages is a series of exponential distribution draws. Thus, some uncertainty is introduced if a client does not follow the protocol of limiting outgoing messages per interval, but we will ignore this for now.

5.1.3 Implementation of Receiver Unobservability

The approach of padding traffic patterns does not work for receiver unobservability. While the receiver could generate a message that is routed through the mix network back to themselves, this strategy is insufficient. Since a receiver does not know if they will receive a message during an interval, they do not know if they should generate an additional message. Furthermore, without coordination, a receiver cannot influence how many messages they receive in an interval by real senders. This could be an unlimited amount, clearly revealing the recipients' incoming traffic pattern.

Two systems claiming receiver unobservability utilize trusted remote inboxes from where messages are pulled, instead of directly receiving messages from the mix network.

Loopix

Loopix proposes special providers with host inboxes. A client will periodically send a request for messages in its inbox to the provider. If there is an undelivered message in the inbox, it is returned. Otherwise an indistinguishable sized response is crafted. As the authors note, this only achieves receiver unobservability to a third party and requires providers to be outside of control of the adversary.

AnonPoP

AnonPoP proposes an inbox hosted somewhere in the mix network and changing pull-mixes through which receiving messages from the inbox is done. A client keeps up a request-pool at the pull-mix that receives a message (or a dummy message generated by the inbox) every interval. Having a pool of requests aims to not reveal a client disconnecting to an eavesdropper. This is of course limited by the time for which a client can generate valid mix messages into the future, which for IM is likely to be very short.

5.1.4 Constraints

Sender unobservability is achieved at some cost. Limiting user output and forcing dummy traffic affects maximum throughput, bandwidth waste and message frequency. While these may be obvious they, to our knowledge, have not been stated previously.

Lemma 5.1. *Unobservability for an expected throughput of one message for every interval of length t implies a message frequency of at least $\frac{1}{t}$ on average.*

Proof. Assume sender unobservability is provided with a lower message frequency. There exists one user with an average frequency $< \frac{1}{t}$. This user then has t consecutive intervals, where the total amount of messages sent is $\leq t - 1$. Therefore one of the intervals does not contain a message. In this interval the user cannot have sent a message originating from that time window and the user can be excluded from the corresponding unobservability set. □

[2]Since the receiver does not know where this message originated from, no information should be gained by them knowing that they received a dummy message.

Corollary 5.2. *The maximum throughput of a mix network providing sender unobservability is equal to the message frequency.*

As the message size must be fixed (or limited) for them to be indistinguishable, this determines the exact maximum throughput.

Additionally, the bandwidth waste could be determined if the real message frequency is known.

5.1.5 Implications for Anonymity

Sender and receiver unobservability seems abstract, but has significant impact on an anonymous communication network. Whenever messages can be grouped to sessions, or sessions be associated with users, long-term attacks can be launched.

The *long-term intersection attack* starts with the entire anonymity set and then excludes (or rather decrease assigned probabilities) to any user (or user pair) that have a mismatching communication pattern. To deanonymize a users the adversary might have knowledge about when messages were received. [3] The adversary can then rule out users that were not sending messages in some time frame before that. Similarly this will work for receiver observability.

So clearly sender unobservability has an effect on anonymity. While the possibility of long-term intersection attacks has been studied [BL03], quantifying this in practical systems that do not guarantee perfect unobservability has not been done. The attack will likely be much more effective when there are more distinct communication patterns and the time frames in which messages could be sent are small. For an Instant Messaging case where communication is frequent, short-windowed and likely recurring between users, this could prove to render anonymity impossible. [4] Unfortunately, the formal tools to quantify such anonymity loss are not available.

Long-term Intersection despite Unobservability

In theory unobservability must be provided at any cost to provide anonymity. Accepting the cost however might actually lead to long-term intersection attacks, too. Any practical messaging system will have to account for user churn (that is, users going offline). Since unobservability leads to higher message frequency and bandwidth usage, users pay a cost, even when they are not actively using the system to communicate. Users are then incentivized to go offline after their intentional messaging activity ends. Users going offline is clearly observable by a GPA and therefore can be used to intersect the anonymity set in a similar fashion.

Any anonymous communication system will be open to such intersection attacks, determined by their users behaviour. Whether providing unobservability is a good tradeoff cannot be said without quantifying anonymity loss through these attacks and in turn taking complex user behaviour into account.

5.1.6 Instant Messaging

We can take a further look at providing (sender) unobservability for the IM case. The messaging pattern in an IM setting is vastly different from a email like setting. Messages might only be seconds apart from each other with an expected delivery of seconds, too. Providing sender unobservability with traditional usage patterns will thus come at a high cost.

Providing unobservability for low-latency applications has been claimed before, but the only justification for this was that parameters (e.g. frequency) are tunable to the appropriate requirements, without going more-indepth at the cost. [PHE+17][BFK01]

Throughput & Bandwidth

Unobservable message sending would require an average throughput without any user-controlled bursts. With this in mind, allowing media transport (with a sudden high throughput compared to the usual text messaging) is unlikely to be feasible.

[3] Consider the user to be a whistleblower posting publicly.

[4] The whistleblower can probably send their message to the public with a delay of days. This is in contrast to Instant Messaging where replies are set in context and are expected to arrive within seconds.

Total bandwidth requirements without media transport is acceptable. Mixminion uses 32KB per message including encryption and routing information. [DDM03] Therefore, allowing a message every 10 seconds would result in 276480 KB bandwidth usage daily.

Message Frequency

Message frequency will be problematic, especially for mobile clients, where modern messaging takes place. Sending a message every 10 seconds there means waking up the device and network. Booknong et al. quantify the battery consumption of a typical application that synchronized with the network in 15 minute intervals to be 10-15% of the daily total consumption of that phone. [BPC13]

This clearly makes unobservability infeasible if mobile devices are to be considered.

5.1.7 Alternatives

We showed that unobservability is formally imcompatible with user churn. Mitigations for this have been proposed that can be grouped into two categories.

Action Queues

When the interactions with the mix network for messaging is not done directly by the user, but a provider (either within the mix network like AnonPoP or outside like Loopix) is used, the user can feed them actions to be queued for some time. This aims to mask user behaviour when they are in fact offline.

Studying these mitigations formally does not yield satisfying results. At best these can hide offline periods for some finite consecutive time. The window is limited by how far in the future the user can generate valid mix network messages. If one deploys countermeasures against the compulsion attack as described in section 4.1.1, this is unlikely to be significantly more than a few hours.

Additionally, the unobservability then is only provided assuming the provider is to be trusted.

Steganography

Steganographic techniques aim to hide whether any network interaction is related to the mix network or *normal* use of other applications.

Nonesuch hides postings to the mix network in Usenet images. [HBSD06] Entry nodes to the mix network try to decrypt all publicly posted Usenet images.

The HTTP protocol also allows for some covert channels where data can be transmitted. [Bau03]

If these techniques are effective in masking interaction is hard to study formally and depends on user behaviour.

5.2 Colluding mixes

Attacks on anonymity where the adversary uses their capabilities as a passive eavesdropper have been studied extensively, and so have attacks where adversaries become active with their network capabilities by delaying and inserting messages into mixes. In the literature it is often only briefly acknowledged that the adversary can also run mixes themselves or have mixes collude with them. This is neither unrealistic nor expensive per se, considering Tor, while not a mix network, the most widely adopted ACN allows anyone to run nodes and join the network within hours.

The Tor network is currently comprised of 6000 such relays. [Proc] For Tor colluding nodes is not a major issue:

> [...] It doesn't make any sense for the NSA to run the relays [...] they're already watching AT&T and Deutsche Telekom and the cables underneath the oceans so they don't have to run the relays to attack the Tor network. (Roger Dingledine, Tor co-founder [Din17])

Mix networks on the other hand aim to protect against such a GPA (what he is referring to, more formally), hence for an adversary it isn't sufficient to only eavesdrop all network links. Therefore colluding mixes becomes a more important concern for mix networks.

5.2.1 Mix Selection

For this section, we assume the route in the mix network to be of length 3. That is 3 mixes are chained before a message is routed to its destination. This is consistent with what ACNs like Tor or the Katzenpost project use.

Further, we rule out fixed mix cascades, with which clients will reuse a chain of mixes multiple times. Because of the potential message frequency in an IM use case, the used chain would be a highly valued target and therefore would have to be changed often. The situation would then approach one where the chain is changed every time.

Concluding our assumptions then, a client will choose 3 mixes uniformly at random from all n mixes. [5] The adversary controls d of these.

Lemma 5.3. *With n mixes of which m collude the probability that a chain of mixes is entirely colluding is*

$$\frac{d(d-1)(d-2)}{n(n-1)(n-2)}.$$

The lemma follows directly from combinatorics.

As a practical limit on the amount of mixes we choose 10000, considering Tor managed to achieve 6000 relays with its widespread adoption, this is a realistic maximum estimate.

n	$d=3$	$d=n/100$	$d=n/10$	$d=n/2$
10	0.008	0	0	0.083
100	0.0000006	0	0.00008	0.121
1000	≈ 0	$7.221651e-07$	0.001	0.1247
10000	≈ 0	$9.704911e-07$	0.001	0.125

Figure 5.1: Probabilities of a colluding chain getting chosen with n total mixes and d colluding mixes.

5.2.2 Long-term Disclosure

We can also express figure 5.1 as messages that must be sent, so that a colluding chain is expected.

n	$d=3$	$d=n/100$	$d=n/10$	$d=n/2$
10	120	NA	NA	12
100	161700	NA	1347.5	8.25
1000	166167000	1384725	1027.6	8.024
10000	166616670000	1030406	1002.7	8.002

Figure 5.2: Expected number of messages until a colluding chain will be chosen with n total mixes and d colluding mixes.

Putting these number in perspective requires usage data of Instant Messaging. In a study for WhatsApp usage, the average number of messages exchanged (sent and received) per user per day were found to be 145.22. [RSS+18]

Before proceeding with the numbers an important distinction should be noted from the attackers perspective: Is the goal deanonymizing the communication from a particular user or deanonymizing usage of as many users as possible (and maybe even use this information to exclude deanonymized users from target)?

Deanonymizing One User

Assuming this daily message volume table 5.2 can be shown as the expected time an adversary needs to wait until they can deanonymize a message from one targeted user. [6]

[5] This isn't strictly true for all cases, as mix topology influences the combinations. See [BPS01]. But this does not change the attack results significantly.

n	$d = 3$	$d = n/100$	$d = n/10$	$d = n/2$
10	20.2 hours	NA	NA	1.888 hours
100	1134 days	NA	9.453 days	1.39 hours
1000	$\approx$ 3200 years	9714 days	7.209 days	1.351 hours
10000	$\approx$ 3211000 years	7229 days	7.034 days	1.347 hours

Figure 5.3: Expected time until a user with 142.44 messages daily chooses a colluding chain with n total mixes and d colluding mixes.

For a mix network that is of the size as the Tor network currently (6000 nodes) it would take about 56 days with just 5% (300) nodes colluding with the adversary. The Noisebridge project runs Tor relays with donation funds and reports that one of their nodes with $320 Mbps$ costs \$800 per month do operate. [Prob] Currently this would equal 1.28% of the average consumed bandwidth of the Tor network. [7] [Prod]

Deanonymizing Anyone

The problem is enhanced significantly if the interest of the adversary is just deanonymizing any communication happening over the mix network. To stay with WhatsApp as our example, the app is reported to transport 100 billion messages daily. [Sin]

n	$m = 3$	$d = n/100$	$d = n/10$	$d = n/2$
10	$\approx$ 83.3 million	NA	NA	$\approx$ 833.3 million
100	61842.92	NA	1347.5	8.25
1000	60.18	7222	9731176	1246246306
10000	0.06	9705	9973013	1249624956

Figure 5.4: Expected number of communications per day with a colluding chain, with n total mixes and d colluding mixes.

It should be noted that this kind of attack is not only limited to attacking the ACN as a whole. By revealing communication links of all users (which are known to the adversary), they may be excluded in the set of potential targets, therefore narrowing down the anonymity set of interest.

Varying Chain Lengths

One assumption was that mix chains of 3 are chosen. An obvious mitigation then could be to have a varying length of mix chains, depending on the fraction of dishonest mixes $\frac{d}{n}$. [8]

To determine the effectiveness we take a standard security parameter of 128 bits.

Definiton 5.4. *A guessing probability p is called negligible if $p \leq 2^{-128}$.*

For large d and n the probability of choosing an all-colluding k-length chain approaches $\frac{d}{n}^k$. We simplify this to hold for all our scenarios to state the following lemma:

Lemma 5.5. *For the fraction of colluding mixes $\frac{d}{n}$ the probability of choosing a chain of length k where all mixes collude is negligible for*

$$k > log_{\frac{d}{n}}(2^{-128}).$$

[6] The data for WhatsApp usage only includes actual messages, IM protocols usually also include presence updates (e.g. online status) and even a status update whenever the communication partner is typing. According to [XGT07] these non-chat messages make up the vast majority of traffic. If such features were to be implemented on a mix network, the message volume will increase significantly.

[7] A Tor node and a mix network node will not have equal running cost, but since they both route packets or messages respectively and perform onion-decryption on them they should be rougly comparable.

[8] We simplify here and assume this fraction is roughly known.

Proof.

$$2^{-128} \geq \frac{d}{n}^k$$

Since $0 < \frac{d}{n} < 1$ taking the $log_{\frac{d}{n}}$ flips the sign of the inequality.

$$k > log_{\frac{d}{n}}(2^{-128})$$

□

Using this lemma we can take a look at the required chain lengths for different colluding mix fractions. For just 5% of colluding mixes a chain of 30 mixes is required for each message. The delay of this is

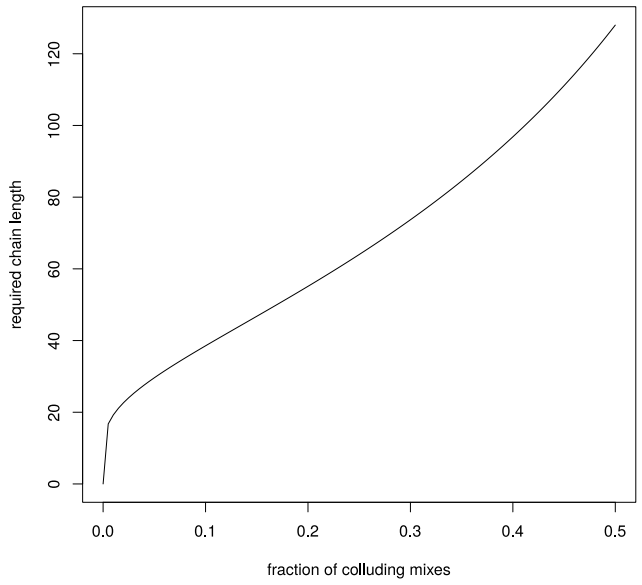

Figure 5.5: Required chain length to ensure a security parameter of 128 bits with a fraction of $\frac{d}{n}$ of colluding mixes.

likely to be infeasible for any low latency application, including IM. Apart from that the rate of dropped messages because of normal operation will most likely be close to one, as mixes are allowed to drop messages they can't process (due to load or other reasons), like most other networking equipment.

Summary

Running mixes proves to be extremely effective for an adversary. For both, attacking individual users and the ACN altogether, the attack deanonymizes cost-effectively. Any adversary that has the capabilities of a GPA will realistically also have enough funds to run a substantial fraction of the network nodes themselves. Detecting such colluding mixes is hard, if not impossible, and cannot be solved with purely technical means.

Not allowing nodes to join such an ACN easily, as is the case with Tor, might seem like a remedy. But such an ACN would still have to provide significant traffic capabilities to provide any significant anonymity at all, as detailed in chapter 3.

Another mitigation might be a biased-route selection that include some notion of user-defined trust for certain mixes. These are hard to study formally without restricting the bias significantly. Such a trust-based system would impede usability, as such trust would have to be decided or at least be accessible to the user. Additionally, biased routes enable statistical inference about user based on their

route selection. Even then, it is unclear if such a system would provide significant improvement against colluding mixes.

Finally, sending decoy dummy messages to users is another idea to introduce uncertainty in the attack. The real (often-repeated) communication links would still stand out statistically. If enough dummy traffic is introduced that they do not, the network would reduce to a constant-bandwidth, always-on link for every user combination. This is clearly impractical.

All of these considerations, if effective at all, will provide only some factor of protection against colluding mixes. Given the highly-effective deanonymization the attack yields the security margin would be unlikely to be acceptable for users.

Note that the same argument applies to not just mix networks but any so-called *anytrust* systems, where the security goal depends on at least one of the selected servers not colluding.[9]

5.3 Anonymity Trilemma

Das et al. [DMMK18] were able to prove a previously conjectured and intuitive hypothesis. In what they describe as the *Anonymity Trilemma*, they could show lower bounds for bandwidth and latency with respect to the desired anonymity. While we consider their work less applicable for the evaluation against the IM use case, we still introduce their work and model in this section.

5.3.1 Modelling Anonymous Communication Protocols

In their work they abstract different properties from ACNs to a model they study. This model contains dummy traffic and real traffic that is sent in rounds, and define a maximum number of rounds during which a message can be active. That is, for a message received they only need to consider messages sent in the previous ℓ communication rounds. [10]

The bandwidth overhead is given as a fraction $\beta \in [0, 1]$ of users sending dummy messages in each round.

The analysis is then divided into whether parties are compromised or not and whether users coordinate with each other (e.g. whether to send dummy traffic in the current round). Clearly messages that do not meet in the network at at least one honest node are not indistinguishable to an adversary.

5.3.2 Problems with Evaluation against Instant Messaging

As applying the bounds requires fixing a security parameter (specified as η in the paper), a mix chain length, number of users, their payload message sending rate and number of mixes in the system, we do not further evaluate these bounds here. Doing so would impose unrealistic assumptions on the ACN. Given the number of unknown parameters, we also could not gain meaningful lower bounds for bandwidth when trying different scenarios.

In theory, for appropriately chosen parameters, the authors don't rule out anonymity for Loopix, the exponential mix based system.

[9] This includes all ACNs presented in the introduction.

[10] A consequence of this is that the exponential mix maximum delay, which in theory is unlimited, is cut off after some time in this model.

Chapter 6

Practical Limitations

6.1 User Inboxes

Focusing on sender anonymity like we did up until now, neglects the challenges of building a bidirectional messaging channel out of a mix network. Historically, the use case for mix network was that of remailers, where ideally the message was sent to a public mailing list, where only sender anonymity was an issue. Designs, that allowed for replies based on reply blocks that are held by remailers (so-called nymservers) have been shown to be broken by Sassaman et al. [SCM05]

What an Instant Messaging application would need to provide is not just replies, but a scalable point-to-point communication.

6.1.1 Considerations

We discuss some considerations before introducing the proposed designs from the literature.

Adversarial Senders & Inbox Trust

A communication partner must know the associated routing information to reach the user inbox. This can potentially give them an advantage of coupling an inbox to a particular user of an ACN.

If the ACN is only used for metadata protection against a network operator or provider, allowing an advantage for an adversarial sender may be gotten away with. [1]

Reachability

Inbox designs differ in their capability of a user to be messaged when they are not online and able to participate in the network at the time of sending. For any ACN design that allows users going offline, the messages need to be stored at some (even distributed) location, we call this the inbox.

Although there are designs where users cannot be contacted when offline, there is no practical contemporary mix network that allows for end users participating in the mix network directly. They all have some form of an inbox.

Scalability

For different designs scalability must be examined. One motivating example to build a bidirectional channel from a sender-anonymity mix network is applying end-to-end encryption to all communication. After the messages have traversed the network, they can then be published for all participants of the network, like on a public bulletin board. A participant would then try to decrypt all published messages and post their end-to-end encrypted replies themselves to the public after traversing the mix network.

Clearly this approach does not scale for one-to-one messaging. Bandwidth and computation overhead would be linear in the number of messages published by the mix network, that is, dummy messages and real ones combined.

[1] Strictly speaking, the adversarial sender could still gain an advantage over how active the other communications of the recipient are.

6.1.2 User Inbox

Loopix [PHE+17] proposed special servers that provide inboxes per-user. If someone wants to reach a user they must know at which provider the inbox is located. Providers interact with the mix network on behalf of the clients and the connections from clients to providers happends directly. Thus, if the adversary knows the location of the inbox, the anonymity set is reduced significantly, as the adversary can rule out any user not connected to the provider.

As mentioned, this may be sufficient if the ACN is only leveraged for metadata protection. A practical use case is communication among academics, who know each other but their cooperation should remain secret (until they publish something together). For a whistleblowing scenario where the whistleblower does not fully trust a potential journalist, this inbox design is insufficient.

Additionally, the server that hosts the inbox gains some knowledge on the number of messages received by the user.

The per-user inbox scales well as users will only use bandwidth for the messages addressed to them. Apart from the dummy traffic between client and provider to provide sender and receiver unobservability to a third party, this inbox design is even optimal as exactly the messages intended for the receiver are transmitted to them.

6.1.3 User Deaddrop

The inbox design can also be implemented as deaddrop, where the provider of the deaddrop should not be able to associate it with a user. The user then needs to poll the deaddrop frequently to retrieve messages (the response of the deaddrop can be routed back for every request with a SURB).

The bandwidth of the user is not dependent on the messages or the number of users of the system but their own, making this reasonably scalable.

User's connectivity and requests to the deaddrop are obviously correlated. To prevent a long-term statistical inference of the two, Gelernter et al. [GHL18] propose *Per-Epoch Mailboxes* such that the mailbox resides at a different location for every epoch (i.e. fixed interval). Although they do not provide specifics on how such an inbox can then be found for a potential communication partner, it could be by using a globally-agreed-upon scheme for the inbox location this epoch. [2]

Using adversarial senders, the sender will never hold a valid message (i.e. SURB) to the receiver to launch a compulsion or confirmation attack.

6.1.4 Communication Pair Deaddrops

A deaddrop for messages has also been proposed to be just for one communication pair, e.g. in Vuvuzela. [vdHLZZ15] Agreeing on such a deaddrop location then is another design problem. In Vuvuzela they propose running a seperate *invitation* high-bandwidth channel, that is essentially a public broadcast for conversation requests. Such a channel could then be turned off (not monitored) when no new conversations are expected.

Another proposal is bootstrapping such a location from a shared secret that was transmitted out-of-band. [App14]

The provider of such a deaddrop cannot easily associate the deaddrop with users. To even mask the message volume within a deaddrop to the provider, Vuvuzela proposes a high-bandwidth dummy traffic that masks access patterns with dummy messages.

Apart from the access pattern of the deaddrop no information about the communication partners is revealed to the provider. Inferring users from the access pattern with a confirmation attack is a long-term attack vector. Changing the deaddrop location often is therefore recommended.

The deaddrop per communication pair design impacts message throughput significantly compared to the per-user deaddrop. If the user has multiple simultaneous conversations, they can either poll more frequently, which would be noticed by a passive adversary, or they divide their constant polling rate

[2] Consider a globally known list of providers of length n, the user's inbox at epoch i will reside at the $i\%n$-th entry of the list.

To prevent knowing inbox locations of the future, the epoch could have a random value associated with it instead of i. In Tor such a design even integrates a shared random value agreement, so users do not need to trust a single entity to generate this random value. [LW17]

among all their active deaddrops. The latter option would still reveal this to the deaddrop provider, as the deaddrop would be polled less often.

Vuvuzela thus only allows one simultaneous conversation per user, impacting usability greatly.

6.2 Inbox & Invitation Denial-of-Service

Now that inbox designs proposed in the literature were introduced, we can investigate a simple attack that has not seen any academic interest, but have found it to be critical for an IM setting.

6.2.1 Denial-of-Service

The attack can be performed by anyone knowing the location of the inbox by simply sending messages to it. In fact, the adversary does not need to know whom the inbox belongs to, as any message that must be transmitted from the inbox to the client is sufficient for the Denial-of-Service.

The bottleneck that is exploited is between inbox and client. When unobservability is desired, this channel must be used at a constant traffic rate, as described in 5.1.

The attack is cost-effective, if the traffic throughput is reasonably set. To save bandwidth this is likely set with little extra capacity over the expected communication pattern. Therefore, the attack will be in the same order of magnitude as normal messaging.

6.2.2 Impact

The user inbox and user deaddrop design are both affected as the location is known publically in both cases. Since the same inboxes are used for all communications, Denial-of-Service of that inbox leads to a Denial-of-Service of that user.

Only the communication pair deaddrop design is not affected for two reasons: It's location is only known to the communication participants (and the provider, who can Denial-of-Service anyway) and the deaddrop is distinct for each pair. If the inbox cannot be used for communication anymore, a client can simply stop querying messages from it and the other communications are unaffected.

While a Denial-of-Service attack does not lead to a direct deanonymization, it is more severe for ACNs than other messaging systems. Preventing communication over the ACN can lead to users switching to less secure systems, where they are deanonymized (more easily).

If messagging over the ACN is not completely prevented, the attack still has an impact. Dropped messages lead to degraded anonymity when using a mix network since retransmissions increase the potential of statistical disclosure as shown in [BDMT07]. [3]

6.2.3 Mitigations

Increasing Throughput

The most obvious mitigation would be to increase throughput of the channel from client to deaddrop. This would increase overall system bandwidth use for all users, even if no attack is launched. As the bandwidth usage and frequency of polling is already likely a limiting factor for adoption, increasing throughput, which would have to be several order of magnitudes above normal messaging behavior to be effective, is not a satisfying mitigation

Increasing Attack Cost

The attack cost is that of sending a message to the deaddrop provider that is indistinguishable from a valid message to them. [War] employs a Proof-of-Work scheme, increasing the computational cost for sending all messages. For Instant Messaging specifically, this would be less applicable as messages can't require a Proof-of-Work that takes users longer than a few seconds without experiencing major usability issues.

[3] Their work focuses on a Denial-of-Service of mixes, but depending on the implementation, an inbox provider stacking up unretrieved messages could drop future messages at some point. A similar analysis would then apply.

Anonymous Blacklisting

Anonymous Blacklisting was proposed in Tor to differentiate between honest and malicious users. [LH11] Traditionally, these required a trusted third party to authenticate the user via blinded signature schemes, if they aren't blacklisted.

Later schemes such as BLAC [TAKS10] have relaxed the trust to a third party not to deanonymize a user, but they come with significant computational overhead.

6.2.4 Summary

Transforming a mix network into a IM system that can be used as a bidirectional channel poses additional challenges. If the system wants to use globally-known inboxes, the Inbox-Denial-of-Service is problematic. Although we could borrow mitigations from other ACNs and messaging systems, they bring downsides to the usability and feasibility. Especially for the IM case these mitigations seems even more unlikely, as messages are frequent and time-critical.

On the other hand, the communication pair inbox opens up different challenges. Any such system will reveal some information long-term about how many communication pairs they currently maintain actively. This information is leaked to the provider, because the polling frequency decreases, or the network edge on the client side, as the polling frequency multiplies, if the frequency is kept up at each deaddrop individually.

To our knowledge there has been no discussion on how to balance these issues for mix network design.

Chapter 7

Discussion & Open Problems

We highlight some open problems related to IM and mix networks that we have encountered during this work. Further, we outline what direction future research and design might take.

7.1 Open Problems

7.1.1 Exponential Mixes

Delay Parameter

The anonymity of exponential mixes fundamentally relies on there being enough traffic during the window of which a message is delayed. All proposals let the delay be determined by the sender in advance, but no design has been proposed how the sender could know the currently appropriate exponential distribution parameter themselves.

For IM the tradeoff between anonymity and delay is even more critical, so a dynamically changing parameter could be considered that takes the systems current traffic load into consideration. How this traffic load could be reported in a secure manner is a fundamental problem for this to work.

Without such a parameter changing scheme the exponential mix does not seem useful for an IM setting. A timed mix, where the delay is not known in advance, will have an increased delay if traffic is low, and thus less likely to compromise anonymity. As shown in chapter 3, their expected delay and anonymity tradeoff is equal, but a user would probably prefer unusually long delays than an occasional anonymity compromise.

Adversary Model

Assuming a dynamically changing delay parameter depending on the traffic load exists, this would have to be secure against an adversary. Current models can trivially break any such scheme, as they are allowed to insert unlimited own messages into the system. Therefore, the total traffic of the system could at any time be comprised of 99.99% the adversary's own messages. As soon as the delay parameter is adjusted to this load, the adversary could stop their traffic and thus reduce the anonymity by that fraction. Any real-world system will have some transitional period until the parameter is adjusted again and until then, all traffic is deanonymized.

In an IM setting the transitional period is likely to be at least as long as the message delay, making this attack very effective.

Any proposal for a dynamic parameter scheme will have to also propose a more fine-grained and weaker adversary model, that considers the cost of such active attacks. This cost then needs to be proven to be too high for the dynamic parameter scheme.

7.1.2 Anonymity Metrics

When comparing different mixing strategies with respect to their entropy-based anonymity, they showed very little difference. While our findings did not achieve the goal of deciding which mixing strategy is

most suitable for IM applications, they were surprising nonetheless and open up many questions about anonymity metrics.

The fact that they showed little difference under normal operation, could have many origins. We were unable to prove any hypothesis about the behavior of the metric, but questions that remain open are:

- Can the seemingly constant entropy to expected delay relation be explained or even be proven?
- Is the constant traffic load the culprit of the analysis? Are there other assumptions that lead to meaningful results?
- Is the entropy-based anonymity metric unfit and meaningless for the evaluation in the design phase?
- Are there other metrics to evaluate mixing strategies without strict assumptions on the input?

7.1.3 Attacks

Attacks for mix networks have been widely studied, still our work opened up many questions.

Threat Models

The mix network research always focused on the strongest conceivable adversary. A GPA that can additionally perform any active attack globally and spares no expense when launching an attack.

This fixation on such a strong threat model seemed limiting during our work. It lead to an abundance of research of the arguably most powerful of all attacks, the $(n-1)$-attack, while more simple attacks have been overlooked.

How often these attacks can and will be launched in practice has not been of much concern. For this more refined threat models are needed.

Syverson [Syv13] criticized this fixation on such a strong threat model, that is too strong for formal analysis, but in some cases even too weak for real-world attacks. Apart from the colluding mixes scenario the compulsion attack, where the adversary can compromise any node with sufficient effort also highlights this problem.

An evaluation is needed here. The current threat model cannot compare what is more cost-effective and therefore should be protected against: Colluding half the mixes or delaying 99% of the systems traffic?

Denial-of-Service

Denial-of-Service attacks have seen little acedemic attention. We showed that for messaging (not just instant) a Denial-of-Service inbox attack is critical to the system. Design proposals in the literature either miss the potential for these attacks completely, or leave them as future work.

We think any design that does not take Denial-of-Service attacks and their impact into consideration cannot be taken seriously. Even if the otherwise perfect anonymity system is deployed, it is worthless, if an adversary can render it unusable for little cost.

Open Systems

Anonymity systems that require nodes or servers (i.e. are not peer-to-peer) have always been considered as being run by volunteers and organizations, who could join the network in an open manner. For Tor this proved to be a good approach. It is inconceivable how it would have gained such widespread adoption and provide the bandwidth capacities it has today, if relays were not allowed to join the network without much verification.

Our analysis of colluding mixes showed that this approach cannot simply be translated to mix networks. With the stronger threat model it becomes very effective for an adversary to join the network.

Detecting colluding mixes does not seem possible, as they behave just like normal mixes until a communication happens to be on a completely colluding path. This can even be implemented efficiently as an extension, if one mix has a list of the other colluding mixes it will only save connections that are routed to one of the other colluding mixes.

Therefore, preventing colluding mixes from joining the system might be the only effective countermeasure. How an ACN can still be bootstrapped to provide the bandwidth capacity to ensure anonymity is a practical problem to be solved.

7.2 Proposals

Instant Messaging proved to bring additional obstacles to enable private messaging. We argue for some proposals that should be worked on first, considering that even Email-like asynchronous communication has no satisfying design that uses mix networks yet.

7.2.1 Relaxing IM Requirements

Private Messaging using mix networks is unlikely to be used as a drop-in replacement for any messaging. It will come at a significant cost to anonymize communications. Compare this to end-to-end encryption, that is widely deployed today, because the additional overhead is negligible.

One way of allowing mix networks more leeway is to relax the requirements a system would need to meet. A messaging application that does not guarantee reliability or even a maximum delay of messages is more realistic.

7.2.2 Usability

When a private messaging application is neither comparable to Email, nor Instant Messaging it can pose additional challenges to usability. A user should be informed of the state of the system, the current expected delay and even have a say in practical problems like when to switch inbox locations as this affects their messaging usability.

A session-based messaging approach like Off-The-Record messaging relied upon seems useful to tackle many of the problems. For each session a new inbox could be used. If the sessions are short enough, the status of the system can be considered to be constant and therefore be made transparent for the users of the session.

Users should be made aware of the underlying conditions of the system. For example, problems can be reduced if the messages in a several seconds window are bundled as one message to decrease message sending frequency. Further, if a message is forced to refer to other messages, problems with reliable message delivery could be less severe to the user.

For all these considerations usability is critical of the system, as the anonymity set is at most all users of the system. If the system's usability proves to be a high entry barrier for users, it might not provide any privacy after all.

Chapter 8

Conclusion

In this work we looked at many aspects of mix networks with a focus on Instant Messaging. Previous research was evaluated and extended, possibilities of future research was presented. Finally, we would like to give our assessment of the future of private messaging based on this work.

8.1 Evaluation of Research

Our work on anonymity metrics posed some interesting empirical findings, namely that mixing strategies could not be ruled out based on their entropy. More interesting even is the fact that the model does not seem to provide meaningful results at all.

Of more practical nature was our analysis on commonly studied attacks, their defenses and if they can be applied in a mix network to be used for IM. We could show that some of the defenses are not practical. In other cases, the IM use case was even beneficial like for the implementation of a replay cache, as message can lose validity much faster.

Further, we looked at the constraints of anonymity and possible tradeoffs that could be made.

Our focus on colluding mixes showed that research claiming low-latency applications can used over mix networks are naive and overlook the specifics such use cases would bring. For IM we showed that the already cost-effective attack of colluding mixes is even more devastating.

Further, we showed that practical design choices like inboxes have significant impact on the anonymity of such systems. Research needs to take these into consideration if their goal is to pave the way of a private messaging system.

8.2 Future Research

We outlined directions in which future research on this topic could be developed. We advocate for a shift in focus of certain attacks and of threat models.

More precise and concrete adversaries are needed to further develop many of the aspects in mix network research.

8.3 Instant Messaging

Some designs of messaging systems were described in the introduction. For a scalable one-to-one messaging system, mix networks seem like the only practical candidate. This work can be seen as a founded argument why mix networks will not succeed in achieving this goal. Unfortunately, they still seem like the most hopeful technique of providing private messaging. Privacy as a drop-in replacement for current IM is clearly wishful thinking, as shown by this work. If it is ever successful of providing privacy in practice, it will not replace traditional non-private IM, due to the high cost.

Bibliography

[AL08] Christer Andersson and Reine Lundin. On the fundamentals of anonymity metrics. In Simone Fischer-Hübner, Penny Duquenoy, Albin Zuccato, and Leonardo Martucci, editors, *The Future of Identity in the Information Society*, pages 325–341. Springer US, 2008.

[App14] Jacob Appelbaum. Going Dark: Phrase Automated Nym Discovery Authentication. `https://github.com/agl/pond/blob/master/papers/panda`, 2014. Accessed: 2020-12-08.

[Bau03] Matthias Bauer. New covert channels in HTTP: Adding unwitting web browsers to anonymity sets. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, WPES '03, page 72–78. Association for Computing Machinery, 2003.

[BCC+15] Stevens Blond, David Choffnes, William Caldwell, Peter Druschel, and Nicholas Merritt. Herd: A scalable, traffic analysis resistant anonymity network for VoIP systems. pages 639–652. Association for Computing Machinery, 08 2015.

[BDG15] Nikita Borisov, George Danezis, and Ian Goldberg. DP5: A private presence service. *Proceedings on Privacy Enhancing Technologies*, 2015(2), June 2015.

[BDMT07] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, page 92–102. Association for Computing Machinery, 2007.

[BFK01] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web mixes: A system for anonymous and unobservable internet access. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings*, pages 115–129. Springer Berlin Heidelberg, 2001.

[BL03] Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies*, pages 110–128. Springer Berlin Heidelberg, 2003.

[BMS01] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, *Information Hiding*, pages 245–257. Springer Berlin Heidelberg, 2001.

[BPC13] Sirapat Boonkrong and Dinh Pham Cao. The comparison of impacts to android phone battery between polling data and pushing data. In *International Conference on Computer Networks and Information Technology (ICCNIT 2013)*, 06 2013.

[BPS01] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free mix routes and how to overcome them. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, page 30–45, Berlin, Heidelberg, 2001. Springer-Verlag.

[CGBM15] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (S&P 2015)*, pages 321–338. IEEE Computer Society, May 2015.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, 1981.

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

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

[Dan03] George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Privacy Enhancing Technologies*, pages 1–17. Springer Berlin Heidelberg, 2003.

[Dan05] George Danezis. The traffic analysis of continuous-time mixes. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies*, pages 35–50. Springer Berlin Heidelberg, 2005.

[Dan07] George Danezis. Breaking four mix-related schemes based on universal re-encryption. *International Journal of Information Security*, 6:393–402, 01 2007.

[DC05] George Danezis and Jolyon Clulow. Compulsion resistant anonymous communications. In Mauro Barni, Jordi Herrera-Joancomartí, Stefan Katzenbeisser, and Fernando Pérez-González, editors, *Information Hiding*, pages 11–25. Springer Berlin Heidelberg, 2005.

[DC07] George Danezis and Richard Clayton. Introducing traffic analysis. `https://www.cl.cam.ac.uk/~rnc1/TAIntro-book.pdf`, 2007.

[DDM03] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: design of a type III anonymous remailer protocol. In *2003 Symposium on Security and Privacy, 2003.*, pages 2–15, 2003.

[DDTL10] George Danezis, Claudia Díaz, Carmela Troncoso, and Ben Laurie. Drac: An architecture for anonymous low-volume communications. In *Privacy Enhancing Technologies*, 2010.

[DG09] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (S&P 2009)*, pages 269–282. IEEE Computer Society, May 2009.

[Din] Roger Dingledine. One Cell Is Enough to Break Tor's Anonymity. `https://blog.torproject.org/one-cell-enough-break-tors-anonymity`. Accessed: 2020-03-16.

[Din17] Roger Dingledine. Next generation tor onion services. DEFCON 25, 2017.

[DL07] Whitfield Diffie and Susan Landau. *Privacy on the Line: The Politics of Wiretapping and Encryption, Updated and Expanded Edition.* The MIT Press, 2007.

[DMMK18] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency — choose two. In *Proceedings of the 39th IEEE Symposium on Security and Privacy 2018*. IEEE Computer Society, May 2018.

[DMT10] Claudia Diaz, Steven J. Murdoch, and Carmela Troncoso. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In Mikhail J. Atallah and Nicholas J. Hopper, editors, *Privacy Enhancing Technologies*, pages 184–201. Springer Berlin Heidelberg, 2010.

[DP04] Claudia Diaz and Bart Preneel. Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, LNCS, May 2004.

[DS03a] George Danezis and Len Sassaman. Heartbeat traffic to counter (n-1) attacks: Red-green-black mixes. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, WPES '03, page 89–93. Association for Computing Machinery, 2003.

[DS03b] Claudia Diaz and Andrei Serjantov. Generalising mixes. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, pages 18–31. Springer-Verlag, LNCS 2760, March 2003.

[DSCP02] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[DSD] Claudia Diaz, Len Sassaman, and Evelyne Dewitte. Comparison between two practical mix designs. In *Proceedings of ESORICS 2004*, volume 3193, pages 141–159.

[Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, January 1968.

[Fou] XMPP Standards Foundation. Projects using XMPP-based Instant Messaging. `https://xmpp.org/uses/instant-messaging.html`. Accessed: 2020-03-15.

[GCM+16] Trinabh Gupta, Natacha Crooks, Whitney Mulhern, Srinath Setty, Lorenzo Alvisi, and Michael Walfish. Scalable and private media consumption with popcorn. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 91–107. USENIX Association, March 2016.

[GHL18] Nethanel Gelernter, Amir Herzberg, and Hemi Leibowitz. *Two Cents for Strong Anonymity: The Anonymous Post-office Protocol: 16th International Conference, CANS 2017, Hong Kong, China, November 30—December 2, 2017, Revised Selected Papers*, pages 390–412. 01 2018.

[GJ03] Philippe Golle and Markus Jakobsson. Reusable anonymous return channels. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, October 2003.

[GJ04] Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Proceedings of Eurocrypt 2004*, May 2004.

[GJJS04] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal re-encryption for mixnets. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, pages 163–178. Springer Berlin Heidelberg, 2004.

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

[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, February 1996.

[HBS13] A. Houmansadr, C. Brubaker, and V. Shmatikov. The parrot is dead: Observing unobservable network communications. In *2013 IEEE Symposium on Security and Privacy*, pages 65–79, 2013.

[HBSD06] Thomas S. Heydt-Benjamin, Andrei Serjantov, and Benessa Defend. Nonesuch: A mix network with sender unobservability. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, WPES '06, page 1–8, New York, NY, USA, 2006. Association for Computing Machinery.

[HHG19] Y. He, L. Hu, and R. Gao. Detection of Tor traffic hiding under obfs4 protocol based on two-level filtering. In *2019 2nd International Conference on Data Intelligence and Security (ICDIS)*, pages 195–200, 2019.

[Jak99] Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In *Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, PKC '99, page 112–121, Berlin, Heidelberg, 1999. Springer-Verlag.

[JWJ+13] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 337–348. Association for Computing Machinery, 2013.

[KBS+19] Christiane Kuhn, Martin Beck, Stefan Schiffner, Eduard Jorswieck, and Thorsten Strufe. On privacy notions in anonymous communication. *Proceedings on Privacy Enhancing Technologies*, 2019:105–125, 04 2019.

[KCGDF17] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Horizontally scaling strong anonymity. *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017.

[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)*. Springer-Verlag, LNCS 1525, 1998.

[KLD20] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable messaging system with cryptographic privacy. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 759–776. USENIX Association, February 2020.

[LBCZ+13] Stevens Le Blond, David Choffnes, Wenxuan Zhou, Peter Druschel, Hitesh Ballani, and Paul Francis. Towards efficient traffic-analysis resistant anonymity networks. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 303–314. Association for Computing Machinery, 2013.

[LH11] Peter Lofgren and Nicholas Hopper. Bnymble: More anonymous blacklisting at almost no cost. In *Proceedings of Financial Cryptography and Data Security (FC'11)*, February 2011.

[LW17] Arjen K. Lenstra and Benjamin Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *International Journal of Applied Cryptography*, 3(4):330–343, 2017.

[Mad] Shaahin Madani. *Improving security and efficiency of mix-based anonymous communication systems*. PhD thesis, RMIT University, Melbourne Australia.

[MCPS04] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol version 2. Technical report, 2004.

[MHJT14] Brad Miller, Ling Huang, A. D. Joseph, and J. D. Tygar. I know why you went to the clinic: Risks and realization of https traffic analysis. In *Proceedings of the 14th Privacy Enhancing Technologies Symposium (PETS 2014)*, July 2014.

[Mur06] Steven J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 27–36. Association for Computing Machinery, 2006.

[Mye12] Daniel Myers. CS 547 computer system modeling fundamentals lecture 10: The poisson process, 2012. Lecture Notes University of Wisconsin-Madison.

[NBH18] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1962–1976. Association for Computing Machinery, 2018.

[O'C05] Luke O'Connor. On blending attacks for mixes with memory. In Mauro Barni, Jordi Herrera-Joancomartí, Stefan Katzenbeisser, and Fernando Pérez-González, editors, *Information Hiding*, pages 39–52. Springer Berlin Heidelberg, 2005.

[PH00] Andreas Pfitzmann and Marit Hansen. Anonymity, unobservability, and pseudonymity — a proposal for terminology. *Lecture Notes in Computer Science*, 2009:1–9, 01 2000.

[PHE+17] Ania Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In *Proceedings of the 26th Usenix Security Symposium*, August 2017.

[PHG+17] Ania M. Piotrowska, Jamie Hayes, Nethanel Gelernter, George Danezis, and Amir Herzberg. Annotify: A private notification service. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, WPES '17, page 5–15. Association for Computing Machinery, 2017.

[Proa] Katzenpost Project. Katzenpost. `https://katzenpost.mixnetworks.org/index.html`. Accessed: 2020-03-17.

[Prob] Noisebridge Project. tor.noisebridge.net. `https://noisetor.net/`. Accessed: 2020-12-08.

[Proc] Tor Project. Tor servers. `https://metrics.torproject.org/networksize.html`. Accessed: 2020-12-09.

[Prod] Tor Project. Tor traffic. `https://metrics.torproject.org/bandwidth.html`. Accessed: 2020-12-09.

[RSS+18] Avi Rosenfeld, Sigal Sina, David Sarne, Or Avidov, and Sarit Kraus. Whatsapp usage patterns and prediction of demographic characteristics without access to message content. *Demographic Research*, 39:647–670, 09 2018.

[SCM05] Len Sassaman, Bram Cohen, and Nick Mathewson. The pynchon gate: A secure method of pseudonymous mail retrieval. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, WPES '05, page 1–9. Association for Computing Machinery, 2005.

[SD03] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies*, pages 41–53. Springer Berlin Heidelberg, 2003.

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

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

[Ser07] Andrei Serjantov. A fresh look at the generalised mix framework. In Nikita Borisov and Philippe Golle, editors, *Privacy Enhancing Technologies*, pages 17–29. Springer Berlin Heidelberg, 2007.

[SEV+15] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. Raptor: Routing attacks on privacy in tor. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, page 271–286. USENIX Association, 2015.

[SF08] J. Shi and B. Fang. Security and design issues of regroup-and-go mix. In *2008 Third International Conference on Communications and Networking in China*, pages 944–948, 2008.

[SFS06] Jin-Qiao Shi, Bin-Xing Fang, and Li-Jie Shao. Regroup-and-go mixes to counter the $(n-1)$ attack. *Journal of Internet Research*, 16(2):213–223, 2006.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[Sin] Manish Singh. Whatsapp is now delivering roughly 100 billion messages a day. `https://techcrunch.com/2020/10/29/whatsapp-is-now-delivering-roughly-100-billion-messages-a-day/`. Accessed: 2020-12-09.

[SN03] Andrei Serjantov and Richard E. Newman. On the anonymity of timed pool mixes. In *Proceedings of the Workshop on Privacy and Anonymity Issues in Networked and Distributed Systems*, pages 427–434. Kluwer, May 2003.

[SWT01] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on ssh. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM'01. USENIX Association, 2001.

[Syv13] Paul Syverson. Why I'm not an entropist. In Bruce Christianson, James A. Malcolm, Vashek Matyáš, and Michael Roe, editors, *Proceedings of Security Protocols XVII: 17th International Workshop, April 2009, Revised Selected Papers*, pages 231–239. Springer-Verlag, LNCS 7028, 2013.

[TAKS10] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blac: Revoking repeatedly misbehaving anonymous users without relying on ttps. *ACM Transactions on Information and System Security*, 13:39:1–39:33, December 2010.

[TGL+17] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 423–440. Association for Computing Machinery, 2017.

[TH05] Gergely Tóth and Zoltán Hornák. Measuring anonymity in a non-adaptive, real-time system. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies*, pages 226–241. Springer Berlin Heidelberg, 2005.

[THV04] Gergely Tóth, Zoltán Hornák, and Ferenc Vajda. Measuring anonymity revisited. In Sanna Liimatainen and Teemupekka Virtanen, editors, *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*, pages 85–90, November 2004.

[vdHLZZ15] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, page 137–152. Association for Computing Machinery, 2015.

[War] Jonathan Warren. Bitmessage: A peer-to-peer message authentication and delivery system. `https://bitmessage.org/bitmessage.pdf`.

[WCGFJ12] David Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. *OSDI*, pages 179–192, 10 2012.

[WDA+15] Liang Wang, Kevin P. Dyer, Aditya Akella, Thomas Ristenpart, and Thomas Shrimpton. Seeing through network-protocol obfuscation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 57–69. Association for Computing Machinery, 2015.

[XGT07] Z. Xiao, L. Guo, and J. Tracey. Understanding instant messaging traffic characteristics. In *27th International Conference on Distributed Computing Systems (ICDCS '07)*, pages 51–51, 2007.

[YXB05] Ye Zhu, Xinwen Fu, and R. Bettati. On the effectiveness of continuous-time mixes under flow-correlation based anonymity attacks. In *Fourth IEEE International Symposium on Network Computing and Applications*, pages 215–218, 2005.

Appendix A

Simulations for Calculating Entropy

A.1 Constant Fraction Timed Pool Mix

The constant fraction timed pool mix always fires a constant fraction. The fraction is rounded down. The number of inputs is doubled and the so that the simulation can start counting the messages in the middle of the experiment.

```
#the timed pool mix will always output a fraction of the pool
timedpoolmix <- function(numberofinputs, inputarrivalrate, periodlength,
    firingfraction) {
totalperiods<-numberofinputs/inputarrivalrate/periodlength
numberofinputs<-totalperiods*periodlength*inputarrivalrate*2 #2x inputs
   as expected to not be in the end of the experiment
firingperiods<-seq(periodlength,totalperiods*periodlength,periodlength)
inputs<-rexp(numberofinputs,inputarrivalrate)
inputtimes<-cumsum(inputs)
roundstoskip<-10 #the first n rounds entropy will not be taken into
   account

entropy<-vector(length=numberofinputs)
entropycount<-0
probpool<-vector(length=numberofinputs)
currentpoolsize<-0
messages<-vector(length=numberofinputs)
delays<-vector(length=numberofinputs)
j<-1 #index for inputs processed
k<-1 #index for inputs arrived last round
expectedpoolsize<-periodlength*inputarrivalrate/firingfraction
for(i in 1:totalperiods){
  #inputtimes[k] until inputtimes[j] arrived during this period
  while(inputtimes[j]<firingperiods[i]){
    currentpoolsize<-currentpoolsize+1
    probpool[j]<-1.0
    j<-j+1
  }
  tobefired<-floor(currentpoolsize*firingfraction)
  tobefiredfraction<-tobefired/currentpoolsize
  if(i>roundstoskip) {
    #calculate entropy for tobefired messages
    normalizedprobpool<-probpool[1:(j-1)] / sum(probpool[1:(j-1)]) #
```

```
        needs to be normalized to 1
    normalizedprobpool<-normalizedprobpool[normalizedprobpool != 0]
    currententropy<-—sum(normalizedprobpool * log2(normalizedprobpool))
       #entropy of normalizedprobpool
    if(tobefired>0) {
      entropy[(entropycount+1):(entropycount+tobefired)]<-rep(
         currententropy, tobefired) #assign tobefired messages the
         average entropy
      currentavgdelay<-sum((firingperiods[i]-inputtimes[1:(j-1)])*
         probpool[1:(j-1)])
      currentavgdelay<-currentavgdelay / sum(probpool[1:(j-1)])
      delays[(entropycount+1):(entropycount+tobefired)]<-rep(
         currentavgdelay,tobefired)
      messages[(entropycount+1):(entropycount+tobefired)]<-rep(
         currentpoolsize,tobefired)
      entropycount<-entropycount+tobefired
    }
  }
  currentpoolsize=currentpoolsize-tobefired
  #reduce probability by messages fired
  probpool[1:(j-1)] = probpool[1:(j-1)] * (1-tobefiredfraction)

  k=j
}
if(entropycount==0) print("No rounds counted")
avgentropy<-sum(entropy[1:entropycount])/(entropycount)
avgdelay<-sum(delays[1:entropycount])/(entropycount)
avgmessages<-sum(messages[1:entropycount])/(entropycount)
expecteddelay<-(periodlength/(1-(expectedpoolsize-inputarrivalrate*
    periodlength)/(expectedpoolsize)))-periodlength/2
#print(sprintf("Average entropy: %f, Average delay: %f Expected delay: %
    f Average Pool Size: %f Expected pool size: %f Remaining in pool:
    %1.0f Inputs Processed: %1.0f, Total Inputs: %1.0f, Expected Inputs
    Processed: %1.0f", avgentropy, avgdelay, expecteddelay, avgmessages,
     expectedpoolsize, currentpoolsize, j, numberofinputs, totalperiods*
    periodlength*inputarrivalrate))
return(avgentropy)
}
```

A.2 Timed Pool Mix

The simulation for the timed pool mix works as the previous one. The mix always keeps a constant number of messages. If fewer messages are in the pool, the mix does not fire at all for this round.

```
#The constant timed pool mix keeps a constant number of messages in the
    pool
#If the pool contains less messages it does not fire for that round
#If avgpoolsize(=interarrival*periodlength) > minpoolsize, this reduces
    on avg to f=1-(minpool/avgpoolsize)
constanttimedpoolmix <- function(numberofinputs, inputarrivalrate,
    periodlength, minpoolsize) {
totalperiods<-numberofinputs/inputarrivalrate/periodlength
numberofinputs<-totalperiods*periodlength*inputarrivalrate*2 #2x inputs
```

```
    as expected to not be in the end of the experiment
firingperiods<-seq(periodlength,totalperiods*periodlength,periodlength)
inputs<-rexp(numberofinputs,inputarrivalrate)
inputtimes<-cumsum(inputs)
roundstoskip<-10 #the first n rounds entropy will not be taken into
    account
entropy<-vector(length=numberofinputs)
entropycount<-0
probpool<-vector(length=numberofinputs)
delays<-vector(length=numberofinputs)
currentpoolsize<-0
roundsnotfired<-0
j<-1 #index for inputs processed
k<-1 #index for inputs arrived last round
for(i in 1:totalperiods){
  #inputtimes[k] until inputtimes[j] arrived during this period
  while(inputtimes[j]<firingperiods[i]){
    currentpoolsize<-currentpoolsize+1
    probpool[j]<-1
    j<-j+1
  }
  if(currentpoolsize>minpoolsize) { #if poolsize threshold is not
      reached, do not fire
    tobefired<-currentpoolsize-minpoolsize
    tobefiredfraction<-tobefired/currentpoolsize
    currentpoolsize=currentpoolsize-tobefired

    if(i>roundstoskip) {
      #calculate entropy for tobefired messages
      normalizedprobpool<-probpool[1:(j-1)] / sum(probpool[1:(j-1)]) #
          needs to be normalized to 1
      normalizedprobpool<-normalizedprobpool[normalizedprobpool != 0]
      currententropy<--sum(normalizedprobpool * log2(normalizedprobpool)
          ) #entropy of normalizedprobpool
      entropy[(entropycount+1):(entropycount+tobefired)]<-rep(
          currententropy, tobefired) #assign tobefired messages the
          average entropy
      currentavgdelay<-sum((firingperiods[i]-inputtimes[1:(j-1)])*
          probpool[1:(j-1)])
      currentavgdelay<-currentavgdelay / sum(probpool[1:(j-1)])
      delays[(entropycount+1):(entropycount+tobefired)]<-rep(
          currentavgdelay,tobefired)
      entropycount<-entropycount+tobefired
    }
    #reduce probability by messages fired
    probpool[1:(j-1)] = probpool[1:(j-1)] * (1-tobefiredfraction)
  } else { roundsnotfired=roundsnotfired+1 }
  k=j
}
if(entropycount==0) print("No rounds counted")
avgentropy<-sum(entropy[1:entropycount])/(entropycount)
avgdelay<-sum(delays[1:entropycount])/(entropycount)
 print(sprintf("Average entropy: %f, Average delay: %f Remaining in pool
    : %1.0f Inputs Processed: %1.0f, Total Inputs: %1.0f, Expected
```

```
    Inputs␣Processed:␣%1.0f", avgentropy, avgdelay, currentpoolsize, j,
     numberofinputs, totalperiods*periodlength*inputarrivalrate))
return(avgentropy)
}
```

A.3 Dynamic Pool Mix

The dynamic pool mix will only consider the message above the minimum pool size for the fraction of messages to be fired. When the minimum pool size is not reached, the mix does not fire. The simulation is otherwise equal to the previous ones.

```
#the dynamic timed pool mix determines the fraction of messages to send
   depending on the pool size
dynamictimedpoolmix <- function(numberofinputs, inputarrivalrate,
   periodlength, firingfraction, minpoolsize) {
#periodlength<-10 #t in seconds
totalperiods<-floor(numberofinputs/inputarrivalrate/periodlength)
#inputarrivalrate<-100 #input arrival rate per s
numberofinputs<-totalperiods*periodlength*inputarrivalrate*2 #2x inputs
   as expected to not be in the end of the experiment
firingperiods<-seq(periodlength,totalperiods*periodlength,periodlength)
inputs<-rexp(numberofinputs,inputarrivalrate)
inputtimes<-cumsum(inputs)
roundstoskip<-10 #the first n rounds entropy will not be taken into
   account

entropy<-vector(length=numberofinputs)
entropycount<-0
probpool<-vector(length=numberofinputs)
currentpoolsize<-0
j<-1 #index for inputs processed
k<-1 #index for inputs arrived last round
for(i in 1:totalperiods){
  #inputtimes[k] until inputtimes[j] arrived during this period
  while(inputtimes[j]<firingperiods[i]){
    currentpoolsize<-currentpoolsize+1
    probpool[j]<-1
    j<-j+1
  }

  tobefired<-floor((currentpoolsize-minpoolsize)*firingfraction)
  if(currentpoolsize<=minpoolsize || tobefired<=0) next
  tobefiredfraction<-tobefired/currentpoolsize
  currentpoolsize=currentpoolsize-tobefired

  if(i>roundstoskip) {
    #calculate entropy for tobefired messages
    normalizedprobpool<-probpool[1:(j-1)] / sum(probpool[1:(j-1)]) #
       needs to be normalized to 1
    normalizedprobpool<-normalizedprobpool[normalizedprobpool != 0]
    currententropy<-sum(normalizedprobpool * log2(normalizedprobpool))
       #entropy of normalizedprobpool
    entropy[(entropycount+1):(entropycount+tobefired)]<-rep(
```

```
        currententropy, tobefired) #assign tobefired messages the
        average entropy
    entropycount<-entropycount+tobefired
  }
  #reduce probability by messages fired
  probpool[1:(j-1)] = probpool[1:(j-1)] * (1-tobefiredfraction)
  k=j
}
if (entropycount==0) print("no rounds counted")
avgentropy<-sum(entropy[1:entropycount])/(entropycount)
#print(sprintf("Average entropy: %f, Remaining in pool: %1.0f Inputs
    Processed: %1.0f, Total Inputs: %1.0f, Expected Inputs Processed:
    %1.0f", avgentropy, currentpoolsize, j, numberofinputs, totalperiods
    *periodlength*inputarrivalrate))
return(avgentropy)
}
```

A.4 Binomial Mix

The binomial mix in the literature allows for any distribution $P(n)$ of messages to be fired depending on the pool size. We evaluated only a constant fraction case, the simulation can be adapted accordingly.

The exact number of messages to be fired is drawn from a binomial distribution with a mean of $P(n)$. This binomial distribution is actually drawn during the simulation.

```
#the binomial mix determines the number of messages to be fired by
#a binomial distribution on the normal distribution
binomialmix <- function(numberofinputs, inputarrivalrate, periodlength,
    maxprobfiring, firingdistributionsd) {
totalperiods<-numberofinputs/inputarrivalrate/periodlength
firingdistributionmu<-periodlength*inputarrivalrate # mu of normal
    distribution
numberofinputs<-totalperiods*periodlength*inputarrivalrate*2 #2x inputs
    as expected to not be in the end of the experiment
firingperiods<-seq(periodlength,totalperiods*periodlength,periodlength)
inputs<-rexp(numberofinputs,inputarrivalrate)
inputtimes<-cumsum(inputs)
roundstoskip<-10 #the first n rounds entropy will not be taken into
    account

entropy<-vector(length=numberofinputs)
entropycount<-0
probpool<-vector(length=numberofinputs)
currentpoolsize<-0
messages<-vector(length=numberofinputs)
delays<-vector(length=numberofinputs)
j<-1 #index for inputs processed
k<-1 #index for inputs arrived last round
expectedpoolsize<-periodlength*inputarrivalrate/maxprobfiring
for(i in 1:totalperiods){
  #inputtimes[k] until inputtimes[j] arrived during this period
  while(inputtimes[j]<firingperiods[i]){
    currentpoolsize<-currentpoolsize+1
    probpool[j]<-1
```

```
    j<-j+1
  }

  #Set P(n) here
  probfiring<-maxprobfiring

  tobefired<-rbinom(1, currentpoolsize, probfiring)
  tobefiredfraction<-tobefired/currentpoolsize
  currentpoolsize=currentpoolsize-tobefired

  if(i>roundstoskip) {
    #calculate entropy for tobefired messages
    normalizedprobpool<-probpool[1:(j-1)] / sum(probpool[1:(j-1)]) #
       needs to be normalized to 1
    normalizedprobpool<-normalizedprobpool[normalizedprobpool != 0]
    currententropy<--sum(normalizedprobpool * log2(normalizedprobpool))
       #entropy of normalizedprobpool
    entropy[(entropycount+1):(entropycount+tobefired)]<-rep(
       currententropy, tobefired) #assign tobefired messages the
       average entropy
    currentavgdelay<-sum((firingperiods[i]-inputtimes[1:(j-1)])*probpool
       [1:(j-1)])
    currentavgdelay<-currentavgdelay / sum(probpool[1:(j-1)])
    delays[(entropycount+1):(entropycount+tobefired)]<-rep(
       currentavgdelay,tobefired)
    messages[(entropycount+1):(entropycount+tobefired)]<-rep(
       currentpoolsize,tobefired)
    entropycount<-entropycount+tobefired
  }
  #reduce probability by messages fired
  probpool[1:(j-1)] = probpool[1:(j-1)] * (1-tobefiredfraction)
  k=j
}
avgentropy<-sum(entropy[1:entropycount])/(entropycount)
avgdelay<-sum(delays[1:entropycount])/(entropycount)
avgmessages<-sum(messages[1:entropycount])/(entropycount)
expecteddelay<-(periodlength/(1-(expectedpoolsize-inputarrivalrate*
   periodlength)/(expectedpoolsize)))-periodlength/2
#print(sprintf("Average entropy: %f, Average delay: %f Expected delay: %
   f Average Pool Size: %f Expected pool size: %f Remaining in pool:
   %1.0f Inputs Processed: %1.0f, Total Inputs: %1.0f, Expected Inputs
   Processed: %1.0f", avgentropy, avgdelay, expecteddelay, avgmessages,
    expectedpoolsize, currentpoolsize, j, numberofinputs, totalperiods*
   periodlength*inputarrivalrate))
return(avgentropy)
}
```