Analyzing the Feasibility of Integrating a Mix Network into Tor

Theresa Tratzmüller

380177

t.tratzmueller@campus.tu-berlin.de

September 30, 2023

MASTER'S THESIS

Distributed Security Infrastructures Chair

Institut für Kommunikationssysteme

Fakultät IV

Technische Universität Berlin

Examiner 1: Prof. Dr. Florian Tschorsch

Examiner 2: Prof. Dr. Stefan Schmid

Advisor: Christoph Döpmann

Sworn Affidavit

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Berlin, September 30, 2023

[illegible]

Theresa Tratzmüller

Abstract

Tor is the largest deployed anonymous communication system. The privacy it can provide relies on adversaries not being able to observe traffic in the entire network. Unfortunately, adversaries with a global network perspective are increasingly realistic. Mix networks are a family of anonymous communication systems that are able to resist adversaries with this capability. However, these systems generally sacrifice performance in terms of latency and/or bandwidth in order to provide strong anonymity. The Loopix Anonymity System is a mix network that achieves a particularly advantageous trade-off between performance and anonymity. In the present work, we explore whether Tor can evolve to resist ever-more-powerful adversaries by adopting the key characteristics of the Loopix design. Loopix' privacy guarantees build on a mathematical model that requires a precise set of circumstances to apply. We provide a comprehensive discussion of the main challenges that make it difficult to recreate these circumstances under the constraints of Tor's design. In an empirical investigation, we find that shaping the kind of traffic typical of the Tor network in the manner prescribed by Loopix' protocol incurs a prohibitive overhead.

Zusammenfassung

Tor ist das größte real existierende System, das Nutzer:innen anonyme Internetkommunikation ermöglicht. Tors Anonymitätsgarantien setzen voraus, dass gegenerische Entitäten nicht dazu in der Lage sind, den Datenverkehr im gesamten Netzwerk zu überwachen. Leider verfügen beispielsweise führende Nachrichtendienste zunehmend über diese Fähigkeit. Mix-Netzwerke sind eine Familie von Systemen, die anonyme Kommunkation im Angesicht von Parteien mit globaler Perspektive auf das Kommunikationsnetzwerk bieten. Im Allgemeinen erreichen derartige Systeme ihre hohen Anonymitätsgarantien um den Preis von Performanzeinbußen in Form von Latenz oder Bandbreite. Das Loopix-Anonymitätssystem ist ein Mix-Netzwerk, welches einen besonders vorteilhaften Trade-Off zwischen Anonymität und Performanz erzielt. In der vorliegenden Arbeit widmen wir uns der Frage, ob das Tor-Netzwerk sich durch die Übernahme der Eigenschaften von Loopix angesichts zunehmend mächtigerer Gegner weiterentwickeln kann. Die Anonymitätsgarantien von Loopix beruhen auf einem mathematischen Modell, welches nur unter genauen Bedingungen anwendbar ist. Wir bieten eine umfassende Diskussion der wichtigsten Herausforderungen, welche der Herstellung dieser Bedingungen unter den Gegebenheiten des Tor-Designs entgegenstehen. In einer empirischen Untersuchung beobachten wir, dass das Formen des für Tor typischen Datenverkehrs entsprechend der Anforderungen des Loopix-Protokolls äußerst kostspielig ist.

Table of Contents

List of Figures

Chapter 1

Introduction

In digital communications, cryptography ensures that application data is hidden from eavesdroppers. Unfortunately, encrypting communication content does not suffice to provide privacy on the internet as network-level metadata can be used to de-anonymize users. A plethora of anonymous communication systems that aim to protect this metadata have been proposed. However, only few of them have been deployed in a real-world setting, and even less have succeeded in attracting a large user base. The largest and most popular deployed anonymous communication system is Tor [35, 100]. The adversary anticipated by Tor's threat model is limited to a *partial* view of the network. Unfortunately, large intelligence agencies' capabilities increasingly include the ability to observe communications in the *entire* network [78].

Tor will hence have to adapt to match ever more powerful adversaries, or be replaced by another anonymous communication system taking off at scale. Considering how difficult it is for such a system to achieve critical mass [32], and given that only Tor has completed this feat thus far, there seems to be value in trying to enhance Tor rather than attempting to build a new system from scratch.

Mix networks are systems that provide metadata-private communications and resist a global adversary by obfuscating traffic patterns. Traffic pattern obfuscation is therein commonly achieved by delaying traffic, inserting fake traffic, or both. These systems have long been deemed impractical due to the overhead incurred by their traffic obfuscation strategies. The *Loopix Anonymity System* [78] is a modern mix network design that has garnered particular attention as it is able to offer strong anonymity guarantees while supporting low-latency traffic. Can we advance Tor's security guarantees by approximating the behavior of Loopix under the constraints of Tor's design? In the present work, we explore whether Tor is able to integrate Loopix' design, or whether the heterogeneities in between the two systems are too vast to be bridged.

Our main contributions are as follows:

1. We provide a comprehensive theoretical discussion of the main challenges regarding the integration of Loopix with Tor.
2. We empirically investigate the overhead of Loopix' traffic shaping strategy when applied to traffic typical of the Tor network. We contrast this overhead with the overhead of a traffic shaping strategy supported by Tor's official traffic obfuscation framework.
3. We develop an approach to generating traffic representative of Tor in a highly abstract simulation setting and evaluate its validity.

Chapter 2

Background

In this section, we provide an overview of anonymous communications. While our discussion of the field is far from exhaustive (for comprehensive surveys see Danezis and Diaz [23], Edman and Yener [35], and Sasy and Goldberg [90]), we touch on a variety of aspects since exploring whether and how two rather different anonymous communication systems may be integrated with each other requires contrasting many facets of their respective designs.

2.1 Anonymous Communication Systems: Terminology and Adversary Models

Anonymous communication systems are infrastructures on top of existing Internet protocols whose goal consists in protecting communication meta-data [35]. Individual systems differ not only in how they approach this overarching goal, but also in terms of the specific facets of anonymity they focus on as well as the assumed capabilities of the adversaries they intend to protect against. We therefore give definitions of relevant concepts in the realm of anonymity and describe the type of adversary we are concerned with in the present work.

Anonymity and related concepts are commonly defined according to Pfitzmann and Hansen [75]. Pfitzmann and Hansen define the term anonymity as a state of unidentifiability within a set dubbed the *anonymity set*. When communicating over a computer network, participants assume the roles of senders and receivers with respect to the exchanged messages. The anonymity of actors in these two roles can then be defined via their *unlinkability* to specific messages in the system. Unlinkability therein refers to an attacker's inability to determine based on observing a system if two objects of interest (e.g., actors, messages) are related in the context of the system. [1] Providing sender anonymity thus means ensuring that the sender of a specific message cannot be identified within a set of potential senders, and a given sender cannot be mapped to the set of messages they have sent.

Receiver anonymity is defined analogously. Importantly, sender anonymity pertains to the unlinkability of senders with messages observed at the *receiving* end, and receiver anonymity to the unlinkability of receivers to messages observed at the *sending* end. The impossibility of matching messages observed at the *sending* end with their senders and messages observed at the *receiving* end with their receivers is referred to as *unobservability* and generally requires actors to generate and send a stream of random decoy traffic in addition to their actual messages. Note that unobservability generally entails anonymity and thus constitutes a stronger notion. Another distinct concept is relationship anonymity, defined in terms of the unlinkability between senders and receivers. This notion is weaker and implied by either sender- or receiver anonymity.

Pfitzmann and Hansen's definition suggests quantifying anonymity based on the cardinality of the anonymity set. This approach to measuring anonymity has however been criticized as insufficient. The reason is its failure to capture that different members of the anonymity set may be more or less likely to be linked to an event (e.g., a message being sent or received) from the perspective of an adversary observing the system. To use an example given in Serjantov and Danezis [91], consider an anonymity system where users exchange messages

[1] We are not concerned with their a-priori knowledge, which they already possessed before making any observations in the system.

via one intermediate node. Assume the intermediate node operates by forwarding a batch of n messages whenever $N > n$ messages have accumulated in its buffer. Any user of the system that has ever sent a message is then included in the sender anonymity set corresponding to a specific message. However, from a statistical point of view not all of them are equally likely to be its sender. In light of the shortcomings of the anonymity set cardinality metric, Diaz et al. [31] as well as Serjantov and Danezis [91] have suggested using Shannon entropy [93] to quantify anonymity. *Shannon entropy* is an information theoretic measure of the uncertainty contained in a random variable. In the context of anonymous communications, a message m is then associated with a random variable X whose probability mass function captures how likely the individual subjects in the system are to be its sender (resp. receiver) [23].

Quantifying anonymity based on Shannon entropy is not without dispute either. Syverson [99] argues that since anonymity is a multi-faceted construct, the appropriate approach to measuring and protecting it depends on the respective context. Especially in the case of real-world deployed systems, neither the size of the anonymity set nor Shannon entropy may be a sufficient indicator of how likely users are of being de-anonymized. To illustrate, consider an anonymous communication system in which users route their traffic via a number of multiple intermediate nodes, i.e., a network of intermediate nodes constitutes the system's infrastructure. Assume further that the intermediate nodes are not equally likely to be compromised. Then a user only routing their traffic via specific nodes they know to be trustworthy is far more likely to stay anonymous than a user routing their traffic via randomly selected nodes. This circumstance is not at all captured by either the size of the anonymity set or Shannon entropy.

The respective adversaries envisioned when designing anonymous communication systems can be classified based on a multitude of dimensions [35, 84]. The two dimensions most relevant in the context of the present work are *active* - *passive* and *global* - *partial*. While a passive adversary is limited to observing and recording traffic, an active adversary is additionally capable of actively modifying, dropping, and inserting traffic [35]. Active adversaries are often assumed to control one or more network links or operate one or more nodes in the anonymity network, while passive adversaries in general only require the ability to monitor one or more network links. The global-partial dimension concerns the fraction of the network that is visible to the adversary (while the view of a partial adversary is limited, the one of a global adversary is not) [35].

The global passive adversary (*GPA*) is our focus in this work. The GPA is interesting since throughout the history of research on anonymous communications, most system designers have opted to either sacrifice performance in order to defend against a GPA, or deliberately chosen to not resist a GPA for the sake of performance [98]. The perceived trade-off between anonymity, bandwidth, and latency has been formalized in the form of the so-called *Anonymity Trilemma*, which we discuss in more detail in section 2.5. The Anonymity Trilemma has however been challenged by the publication of the *Loopix Anonymity System* [78] (see section 2.6). Further, the GPA's capabilities have long been deemed unrealistic [98]. In light of recent mass surveillance revelations, doubt has been cast on this assumption as well [78].

2.2 Chaum's Anonymous Communication System

In his seminal 1981 publication, Chaum [12] proposed what is often considered the first modern system for communicating anonymously over an unsecured underlying telecommunication system (i.e., the Internet). Since Chaum's original design introduced a number of key ideas that were adopted by later systems, we proceed to describe it in detail.

Instead of sending it directly to its destination, a user of Chaum's protocol routes their message via one or multiple intermediate nodes referred to as *mixes*. Each mix forwards messages in reordered batches, thereby destroying temporal traffic patterns based on which ingoing and outgoing messages could be associated with each other (figure 2.1). Further, it

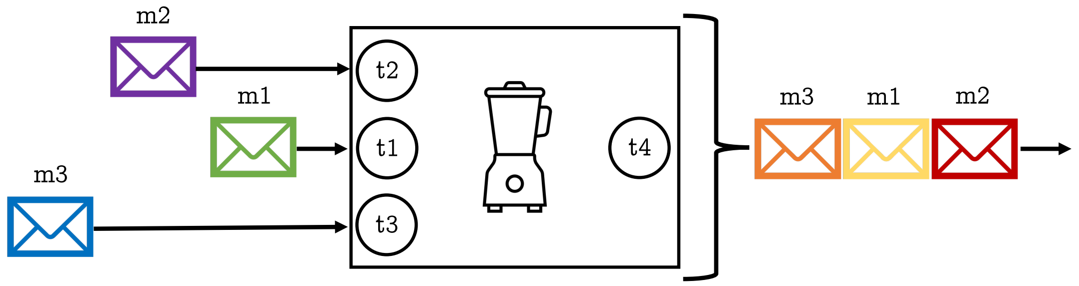


Figure 2.1: Chaum's Mix. A mix node receives three constant-size messages at three different times. It cryptographically transforms them and forwards them in a reordered batch.

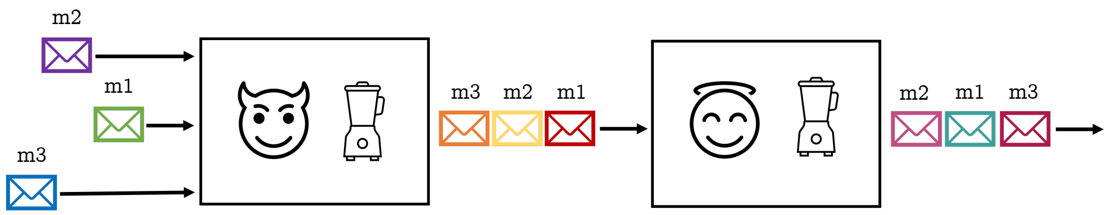


Figure 2.2: Chaum Mix Cascade. The first mix in the cascade is dishonest and does not reorder messages. The second mix is honest and does reorder them, thwarting efforts at tracing them on their way through the network.

transforms each message cryptographically before forwarding it to ensure that ingress and egress messages cannot be linked based on their appearance (bit-wise unlinkability).

Chaum's system leverages public-key cryptography to achieve bit-wise unlinkability of messages entering and leaving a mix. Specifically, before a message is submitted to the system it is encrypted once for each hop on its way to the destination, using a public key associated with the respective node. The order of encryption layers from outermost to innermost are therein mapped to the reverse order of nodes along the path, so that the outermost layer corresponds to the first intermediary node and the innermost layer to the final destination. Upon receiving a message, an intermediate node "tears off" their layer of encryption using their private key, thereby altering the message's appearance and revealing the next hop to forward it to. The final decryption operation occurs at the destination and reveals the original message.

Users can route their message via a series of mixes rather than a single one to limit the trust requirements with respect to individual mixes. Then, even if only a single mix along the route from sender to receiver acts in compliance with the protocol the unlinkability of messages at the sending end with messages at the receiving end is ensured (figure 2.2). Note that Chaum's system provides sender anonymity against a GPA: even when observing all network traffic, successfully linking a specific sender with a message intercepted at the receiving end is highly unlikely. A message observed at the sending end further does not reveal anything about its final destination, since it is of constant size, encrypted, and addressed to the first mix in its route. Receiver anonymity is therefore guaranteed. However, in the case of long messages that have to be split into multiple messages to fulfill the constant size requirement, the number of messages sent and received can be used to link ingress and egress traffic. Chaum suggests that senders supply a constant number of messages to each batch and receivers search the entire output for messages intended for them to mitigate this (thus essentially achieving unobservability, if at a considerable cost).

In summary, the key components of Chaum's system are:

1. Relaying messages via a number of intermediate nodes;
2. Leveraging layered encryption to achieve bit-wise unlinkability of messages;
3. Routing messages through *multiple* mixes to limit the required trust in individual mixes;
4. Obfuscating temporal traffic patterns by batching and re-ordering messages at intermediate nodes.

In our subsequent discussion of anonymous communication systems that were proposed since Chaum's, we distinguish two broad categories: the first category is primarily oriented towards protecting against a global passive adversary, whereas the latter category is primarily concerned with supporting low-latency traffic.

2.3 Systems Focusing on GPA-Resistance

Mix networks are the type of GPA-resistant system most relevant to the present work and thus are the main focus of this section. For completeness, we further include a brief and non-exhaustive overview of other systems designed to resist a GPA.

2.3.1 Mix Networks

Mix networks (sometimes referred to as *mixnets*) are anonymous communication systems that leverage and refine the core components of Chaum's original design. These systems are designed to ensure that messages emitted by each mix cannot be linked to messages received by it based on either appearance or temporal patterns [28].

The act of delaying and/or re-ordering messages to achieve unlinkability based on temporal traffic patterns is commonly referred to as *mixing*. According to Diaz [28], the majority of mix network designs follow one of two general approaches to mixing. The first approach consists in forwarding messages in batches. Chaum's design [12], *Babel* [41], as well as *Mixmaster* [68] and *Mixminion* [24] are prominent examples. An important parameter of batching-based mix networks is the event triggering forwarding, with the simplest strategies relying on quantitative or temporal thresholds (i.e., forwarding messages once a specific amount of messages has been received or once a specific amount of time has elapsed) [28]. Note that a threshold mix may only forward a subset of the messages currently in its buffer once the triggering event occurs. The second approach to mixing is to forward each message with an independently sampled delay. This strategy is called *continuous-time mixing* since it forwards individual messages continuously rather than in discrete rounds delineated by the occurrence of some triggering event. It was first proposed by Kesdogan et al. [58] under the name *SG-Mix* (*Stop-and-Go Mix*). To send a message, a user samples as many random delays from an exponential distribution with parameter μ as there are mixes on the path to the sender. These values are then appended to the message in between applying layers of encryption, so that one delay becomes visible at each intermediate hop. When receiving a message, a mix node then tears off their layer of encryption, retrieves the delay value and forwards the message to the next hop after the delay has elapsed. Due to the memoryless property of the exponential distribution, each message among the set of messages present at a mix node at a given point in time is equally likely to be emitted next. Linking incoming and outgoing messages based on their arrival and departure times is therefore generally not possible. Specifically, when trying to determine which message departing from a mix node corresponds to a specific message m previously received by that mix node, each message received after m and emitted before the mix runs empty is an equally likely candidate. Danezis [21] provides a formal proof of exponential delays being optimal in the context of continuous-time mixing.

Some mix networks supplement their mixing strategy by inserting fake messages (commonly referred to as *padding*, *cover*- or *dummy* traffic respectively *chaff*) into the system. These may be inserted by senders, mix nodes, or both and dropped by intermediate mixes or the receiver.

Cover traffic can be employed to increase the sets of messages' possible senders and receivers, or to hide the fact that users are sending/receiving messages at all (unobservability) [5].

An important aspect of a mix network design is the topology determining route selection, with the endpoints of the continuum of possible topologies being defined by *cascades* and *free-route* networks [23]. In a cascade topology, users are restricted to using one or multiple fixed path/s through the network to send their messages. In a free-route network by contrast, intermediate nodes form a fully-connected graph (clique) and users can send their messages along arbitrary paths. Since both topologies have been shown to possess their advantages and drawbacks, neither can be considered as generally preferable over the other. Free-Route networks give users more flexibility, e.g., they can select nodes based on criteria such as reliability and trustworthiness [30]. Further, this type of topology is more resistant to node failure and less likely to suffer from performance bottlenecks [24, 30]. Cascade topologies (and sparse topologies in general) however concentrate traffic on fewer links, which is conducive to anonymity [19]. Also, while a mix cascade is designed to provide anonymity even if just one of the mixes in the cascade is uncorrupted, free-route networks are vulnerable to a number of passive deanonymization attacks if an adversary controls many mixes [6]. Yet, mix cascades are more susceptible to active attacks [92]. A compromise between free-route networks and cascades is achieved by a stratified topology [34], which we return to in section 2.6.

Mixnets were long believed to introduce prohibitive overhead. For example, a mixing strategy relying on batching or delaying messages introduces (potentially considerable) latencies, while the introduction of cover traffic consumes bandwidth. Older designs also relied heavily on computationally intensive public-key operations. Mixnets thus did not attract as much attention as low-latency anonymous-communication systems. However, with resources like bandwidth and computation becoming ever cheaper, and efficient new mixing strategies and cryptographic techniques being proposed, interest in these systems has been renewed [29]. Modern mixnet designs address issues such as trust, scalability, and performance [90].

2.3.2 Other Types of Systems Focusing on GPA-Resistance

Aside from mix networks, *Dining Cryptographers* networks (*DC-Nets*) constitute the most influential category of anonymous communication systems offering metadata-protection against an adversary with a global network view. They were introduced by Chaum [11] in 1988 as a solution to the problem of unobservable communications. The system requires *all* users to be online concurrently and a fully connected graph of communication relationships and shared keys (i.e., each user has a direct communication path to every other user, and each user shares a cryptographic key with every other user). In order for one participant to send a message, *each* participant has to send an encrypted message to *every* other participant, where only the actual sender's message contains any content. The sender's message is then retrieved by combining all messages.

While elegant and provably secure (given that participants conform to protocol), the original DC-Net design features multiple aspects that make it difficult to implement in practice. Firstly, only one user can send at a time. Secondly, since each user has to share a communication channel and key with every other user, extending the network is cumbersome. Thirdly, sending a message requires all users to be online. And lastly, the communication overhead is in N^2. Notable published systems building on the concept of DC-Nets and addressing the original design's disadvantages are *Herbivore* [39] and *Dissent* [18].

Recent years have seen the emergence of a number of new flavors of GPA-resistant system designs [90]. Systems based on *Differential Privacy* introduce cover traffic according to a probability distribution to enforce guaranteed limits on the granularity of an adversary's inferences (e.g., *Vuvuzela* [106] and *Karaoke* [60]). In designs leveraging *Private Information Retrieval*, participants communicate by privately writing to and/or reading from a data store component (e.g., *Riposte* [17]). Finally, in systems conceptually rooted in *Secure Multiparty Computation*, participants send messages by transforming them cryptographically and supplying the results as inputs to a distributed calculation, from the output of which messages can then be derived (e.g., *MCMix* [3] and *Blinder* [2]).

2.4 Systems Focusing on Low Latency

Low-latency anonymous communication systems are typically designed to support applications requiring interactive, bi-directional communication, which imposes strict constraints on any traffic obfuscation techniques they employ. While most of this section is devoted to describing the Tor network in depth, we also give a brief overview of other systems in this category.

2.4.1 Tor

Tor [100] is the most prominent representative of the *onion routing* principle. Onion routing was conceived at the U.S. Naval Research Laboratory (NRL) in the mid-1990s with the goal of separating identification from routing, i.e., the routing required for a communication to take place should not be contingent on revealing one's identity [98]. While Tor was declared the *second-generation onion router* at the time of its publication, Syverson [98] argues that in fact, two distinct iterations of onion routing preceded Tor, making it the third-generation onion router. The specification of the first generation can be found in Goldschlag et al. [40] as well as Reed et al. [86], the second generation is described in Syverson et al. [102] as well as Reed et al. [85]. Syverson [98] provides an overview of the different generations' properties (including Tor).

Onion routing assumes a synchronous, bi-directional communication model and is designed to support applications such as web browsing and secure shell. The infrastructure of an onion routing network consists of a number of nodes connected via TCP. Traffic is transmitted in small fixed-size chunks, which are *onion-encrypted*, i.e., layered encryption in the Chaumian tradition is utilized to achieve bit-wise unlinkability. Rather than independently selecting a path for each unit of communication, onion routing is based on *circuits*. A circuit is a route connecting an initiator with a destination via a number of intermediate nodes belonging to the onion routing network. A dedicated setup phase has to take place before a circuit can be used to exchange application data. This entails the exchange of symmetric keys between the initiator and each intermediate hop, which are then used for onion-encrypting application data. A communication flow is routed along the same circuit for the entire duration of its lifetime. While each intermediate node "tears off" one layer of encryption when processing a chunk of data traveling from client to destination, one layer of encryption is *added* at each hop when processing a chunk of data traveling from destination to client.

In Tor specifically, circuits in the majority of cases comprise three intermediate nodes called *relays* or *onion routers*. Only the last node in the circuit (referred to as the *exit*) is directly connected to and communicating with the final destination (e.g., a web server). The destination is unaware of the fact that data has arrived via Tor. Any replies are therefore addressed to the exit as the destination believes it to be its communication partner. From there, they are forwarded backwards along the circuit to the Tor client.

Tor clients establish circuits by picking a number of relays (usually three) and incrementally negotiating a symmetric key with each of them via a series of Diffie-Hellman key exchanges. Once negotiation with the first hop has succeeded, extending the circuit to the second hop is mediated via the first hop, and so on. Only the client knows each relay in the circuit, whereas the relays themselves only know their predecessor and successor. Circuit building is relatively expensive as it necessitates several round-trip-times of network communication as well as public-key cryptography. Sending data along the circuit using the symmetric keys exchanged during circuit creation is then comparatively fast. The Tor client preemptively sets up circuits in the background to minimize delays.

Tor transports streams of data. These are broken into so-called *cells* of 514 bytes each. The forwarding behavior of onion routers is designed to preserve the order of cells traveling along the same circuit as well as to maximize performance. No mixing is performed.

A small group of well-known and trusted onion routers called *directory authorities* supply Tor clients with information on network state as well as available relays' *exit policies*, bandwidth, and public keys. Since Tor's onion routers are run by volunteers, they can specify which external destinations they are willing to connect to, if any (expressed in their exit policy) and how much bandwidth they are willing to donate to the Tor network. The reliance on directory authorities implies that Tor (by design) does not constitute a fully decentralized system.

The Tor network was initially deployed in 2002 [1]. As of July 2023, it has over 4 million daily users and comprises more than 6000 relays [105]. The network was specifically designed to bring anonymity to the masses, with usability constituting one of its main design goals [100].

Tor's Threat Model

Tor assumes a partial adversary whose abilities to passively observe as well as actively insert, alter, drop or delay traffic are limited to some fraction of the network [100]. Likewise, they can compromise *a subset* of the onion routers or operate *some* onion routers of their own.

Tor's ultimate goal is to thwart an attacker's efforts at linking communication partners or correlating multiple communications to or from a single user [100]. In terms of the concepts defined in 2.1, Tor aims to provide relationship anonymity as well as - to some extent - sender anonymity: the initiator of a communication flow is visible to the entry node, but not to the middle and exit nodes as well as the destination. The *location-hidden service* (respectively *onion service*) feature additionally allows destinations with high security requirements to remain anonymous towards clients addressing them. Importantly, Tor intends to defend against *traffic analysis* attacks rather than *traffic confirmation* attacks [100]. The latter are characterized by a top-down approach in which an adversary possessing some *a-priori* knowledge observes both ends of a communication flow to confirm a suspicion, e.g. that two specific parties are talking to each other. The former type of attack is more bottom-up: an adversary learns traffic patterns to profile users, identify a user's communication partners or determine which network locations they should attack.

Crucially, Tor's security argument is contingent on choosing unpredictable routes through the network. This is in stark contrast to mix networks, whose security argument hinges on the difficulty of matching a message received by a mix node with a message subsequently emitted by that mix node. As Syverson [99] argues, research on mix networks is routed in cryptology research and has thus adopted the field's tradition of reasoning about a system's security in terms of the security of its core components (*primitives*). The mix is then regarded a mix network's central primitive, whereas Tor does not view onion routers in such a way.

De-Anonymization Attacks on Tor

By design, Tor does not provide anonymity against a global adversary. Since traffic retains its shape on the way through the network (recall that relays do not perform any mixing), an adversary observing both ends of a communication flow can easily match initiator and receiver. A number of publications detail how exactly this vulnerability can be exploited by adversaries controlling a number of relays (e.g., Ling et al. [61], Johnson et al. [54]), autonomous systems (e.g., Feamster and Dingledine [37], Johnson et al. [54], Sun et al. [97]) or internet exchange points (e.g., Murdoch and Zielinski [70], Johnson et al. [54]). Yet, even a weaker adversary limited to a partial view of the network stands a chance of mounting a de-anonymization attack against Tor. In 2005 respectively 2006, just a few years after Tor's initial deployment, Murdoch and Danezis [69] as well as Overlier and Syverson [72] presented the first attacks feasible for an adversary controlling just a single Tor node.

In recent years, a type of attack referred to as *website fingerprinting* [44] has garnered particular attention. This attack can be performed by a passive adversary observing the link between a Tor client and the entry node through which the client's traffic enters the Tor network. Website fingerprinting techniques typically involve training a machine learning or deep learning model on traffic patterns produced by browsing specific websites. The model can then predict which websites were visited by a Tor user based on traffic observed at the link between the client and the respective entry node. Website fingerprinting attacks have been shown to be incredibly effective in a lab setting (e.g., Wang and Goldberg [108], Hayes and Danezis [42], Sirinam et al. [95, 96]). In a recent publication, Cherubin et al. (2022) [15] demonstrate that the effectiveness of the attack introduced in Sirinam et al. [96] is somewhat diminished when mounted on the real Tor network.

2.4.2 Other Systems Focusing on Low Latency

Crowds [87], *Tarzan* [38] and *MorphMix* [88] are prominent examples of low-latency anonymous communication systems based on peer-to-peer networks. The participants themselves form the systems' infrastructure, so that each node assumes the dual roles of user and relay. Thus, determining the initiator of a communication flow is non-trivial since a node emitting traffic may be its originator or may simply be relaying it on behalf of a peer.

Crowds operate on the application layer and do not employ any encryption. The concept was very influential, has however been demonstrated to be broken (e.g. Wright et al. [112]).

The Tarzan- and MorphMix systems operate below the application layer and share many of the characteristics of onion routing. Specifically, they are circuit-based, support bidirectional, synchronous communication, and transmit data in fixed-sized, onion-encrypted chunks. While both systems allow an adversary to break anonymity if each participant only has partial knowledge of the network [22, 103], requiring each participant to have global knowledge of the network limits the systems' practicality [23].

The *Freedom Network* was a failed attempt at offering an anonymous communication system in the spirit of onion routing as a commercial service [7]. Interestingly, paid-for anonymity has recently regained some traction (see section 6.1).

Web MIXes [4] constitute an interesting contribution that aims to anonymize web traffic against a global adversary. The design intends to thwart known de-anonymization attacks against low-latency anonymous communication systems by using a few fixed cascades and by padding the link between each user and the first mix in their respective cascade to a constant rate in both directions. A deployed version of this system (the *Java Anon Proxy JAP*) had to refrain from implementing the intended constant-rate padding for efficiency reasons [35]. The original idea, however, serves as an apt illustration of the latency-bandwidth trade-off in anonymous communication systems, which is the subject of the next section.

2.5 The Anonymity Trilemma

In the literature, anonymous communication systems are typically classified as belonging to one of two categories: while systems in the one category are built to resist a global (passive) adversary, systems in the other category support low-latency communication. This illustrates the notion of a perceived trade-off between anonymity and performance. Protection from an adversary that is able to observe traffic in the entire network requires obfuscating traffic patterns, so that senders and receivers cannot simply be de-anonymized by tracing communication flows on their way through the system. Traffic patterns can be obfuscated by introducing delays, as in a threshold mix design that thwarts efforts at associating messages based on temporal information. If the additional latencies introduced by such an approach cannot be tolerated, another option is to artificially augment the sets of potential senders respectively receivers by introducing dummy traffic into the system. This, in turn, consumes additional bandwidth.

It thus appears that it is not possible for an anonymous communication system to achieve strong anonymity, low latency, and low bandwidth consumption at the same time. This intuitively appealing *anonymity trilemma* is formally proven to hold by Das et al. [26]. Das et al. further compare existing anonymous communication systems according to their latency and bandwidth requirements. The Loopix anonymity system, which we proceed to describe in detail in the following section, occupies an interesting spot in this landscape as it is the only system (among the ones included in Das et al.'s analysis) that incurs *moderate* cost in terms of both latency and bandwidth while still providing protection against a global adversary.

2.6 The Loopix Anonymity System

Loopix [78] is an anonymity system in the tradition of mix networks, whose mixing strategy relies on exponential delays as well as cover traffic. The system thus offers a flexible latency-bandwidth-trade-off: smaller delays can be compensated by increasing the ratio of cover-to-real traffic, and vice versa.

As in Kesdogan's continuous-time mixing design [58], mix nodes forward each message with a delay independently sampled from an exponential distribution with parameter μ. Further, the traffic emitted by clients is shaped according to a Poisson process. This is achieved by delaying and inserting messages wherever appropriate to ensure that the time periods between sending two messages constitute samples from an exponential distribution with parameter λ. [2]

The Loopix design is illustrated in figure 2.3. Mix nodes in Loopix are arranged in a stratified topology, i.e., there are multiple layers of which each mix is assigned to exactly one. Each path through the network includes one mix node per layer. Thus, as opposed to a fully

[2] A Poisson process is characterized by a series of events where the time periods separating consecutive events are exponentially distributed.

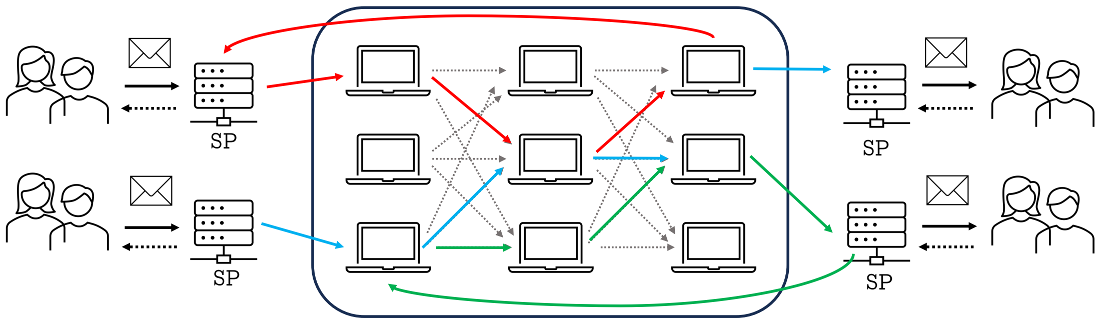


Figure 2.3: Design of the Loopix Anonymity System. Mixes are arranged in a stratified topology. Participants send and (asynchronously) receive messages via their associated service providers (SP). Online participants emit a continuous stream of real and cover traffic in the form of a Poisson process (blue, red). Mixes emit a continuous stream of cover traffic in the form of a Poisson process (green). Looping cover traffic back to its sender is a feature protecting against active attacks, which are not our focus in this work. Diagram adapted from Piotrowska et al. [78]

connected topology (as in, e.g., Tor) there is a smaller number of possible routes, which in turn implies a higher concentration of traffic on the individual links. Access to the network is mediated via so-called *providers*. Each user is associated with one provider long-term and sends as well as receives their messages through it. Asynchronous message retrieval is therefore possible.

Loopix provides sender-receiver third-party unlinkability (equivalent to relationship anonymity in terms of the definitions given in section 2.1). Communicating pairs of senders and receivers are thus aware of each other's identities, but cannot be associated by external parties. This property is based on a message indistinguishability argument. As a consequence of traffic shape corresponding to a Poisson process with parameter μ and message forwarding occurring with a delay sampled from an exponential distribution with parameter λ, the mean number of messages present at a mix node at any point in time can be computed based on μ and λ. Each of these messages is emitted next with equal probability, independent of the time of their arrival. This is due to the memoryless property of the exponential distribution (see Mitzenmacher and Upfal [66] for details). Additionally, mix nodes emit cover traffic according to a Poisson process to further increase the size of the set among which an individual message is indistinguishable. Since an adversary is thus highly unlikely to successfully match a message received by a mix node with a message subsequently emitted by it, they cannot link a message's sender with its receiver by tracing the message through the network.

Moreover, Loopix hides the fact that a given sender is communicating with any receiver at all, as well as that a given receiver is communicating with any sender at all (sender/receiver unobservability). Sender unobservability results from users continuously generating a stream of messages following a Poisson process: at exponentially distributed intervals, a user either sends a real message if it has one to send or emits a dummy message. An adversary is therefore unable to determine whether a specific message sent by a user is real or fake. Receiver unobservability in turn stems from the fact that a provider always emits a constant number of messages when a client polls it to retrieve their messages. If the number of messages in a client's mailbox lies below this constant number, the provider inserts additional dummy messages. The adversary thus cannot tell how many messages a specific client has actually received.

The adversary anticipated by Loopix is able to passively observe traffic inside the entire network (GPA), corrupt a fraction of the mix nodes and/or providers, and control a limited number of system users. While corrupt mix nodes and corrupt users may act maliciously, the threat model explicitly restricts corrupt providers to being honest but curious. Providers can thus be regarded (semi)-trusted, which has been pointed out as one of the weaknesses of the design [59, 60].

Loopix also includes features thwarting active attacks. Since active attacks are not the focus of the present work, we do not go into detail on this aspect.

Chapter 3

Integrating Loopix into Tor: Conceptual Challenges

The security guarantees provided by Loopix are rooted in the mathematical model of queuing theory. The level of security achievable for a certain configuration of the system's parameters can therefore be computed in a straightforward manner. Tor is a large and complex real-world deployed system whose behavior depends on too many variables to lend itself to concise mathematical description. This makes it difficult to gauge how integrating certain aspects of the Loopix protocol into Tor could help increase Tor's anonymity. In this section, we discuss how Tor might be tweaked to adopt the core features of Loopix. We pay particular attention to how the individual building blocks of the Loopix design fit together and how we would expect the system to behave if individual assumptions were loosened or removed.

3.1 Message- vs. Stream Orientation

A fundamental difference between Loopix and Tor consists in their communication model. As a design in the tradition of Mix networks, Loopix is built for exchanging *discrete* messages.

Mix networks crucially require the notion of a message since messages constitute the very objects of mixing. Tor however transports *continuous* streams of data rather than clearly delineated messages. When exploring the possibility of making Tor behave more like a Mix network, specifically Loopix, we need to first ask ourselves: What would we apply mixing to?

As a message-based anonymous communication system, Loopix supports applications such as e-mail or instant messaging. A message exchange requires both the sender and receiver to run the Loopix software, but explicitly does not require both of them to be online at the same time. While providing low-latency communication, message exchange via Loopix is still asynchronous in nature. Individual users assume the roles of senders and receivers dynamically with respect to specific messages, i.e., they are not confined to either role. Application-layer messages that do not fit into one Loopix message are chunked and transported using multiple Loopix messages. A set of Loopix messages that together contain one application-layer message however are not treated as related in the context of the Loopix protocol. Every Loopix message is treated atomically and independently, no matter if it is self-contained in terms of the application layer or not and irrespective of the fact if it constitutes a reply to another message or not.

Tor by contrast is designed to support synchronous, bi-directional interactions taking place in the context of a TCP session. Since HTTP is the application-layer protocol most commonly used with Tor [48, 51], a large portion of the traffic transported over the Tor network is interactive, following a request-response pattern. In the average Tor usage scenario, a Tor user running the Tor client software on their computer makes a request to a destination (e.g., a web server) via the Tor network, where the destination is oblivious to the fact that it is being accessed via Tor and does not run any Tor software. The destination then synchronously replies to the request. While carrying continuous streams of data rather than discrete messages, Tor does have an atomic unit of communication, its 514-byte cells.

In determining a suitable analogue of a message in the context of Tor, two obvious approaches come to mind. Firstly, we could define a message in such a way that it fits the semantics of applications typically used with Tor. Secondly, we might stick to the semantics of the Tor protocol and simply define a message as a cell. We now proceed to more closely examine these two possibilities.

Regarding the first approach, we consider mapping the size of an application-layer message to a number of Tor cells carrying data of that size. (More sophisticated mappings may naturally be conceivable.) The majority of the traffic transported over Tor is web traffic [48, 51], with clients sending small requests and servers replying with webpage content. When loading a webpage, the majority of the exchanged data thus travels from destination to client. This makes it hard to determine a unified notion of a message for both directions of traffic (client $\rightarrow$ server vs. server $\rightarrow$ client); we would likely need one for each. Further, the sizes of client requests and webpages vary too greatly to justify defining a message as the number of cells amounting to an average request/webpage. In fact, the amount of data sent and received when loading a specific webpage is heterogeneous enough to make it one of the top predictors in website fingerprinting attacks (see section 2.4.1). Rather than defining it statically in terms of some average value, message size may also be computed dynamically, e.g., in the form of a moving average of the amount of data sent and received per application layer request. Having different notions of how many cells are considered one message across the network would however produce new problems: How does a node receiving x cells understand them to be one message when it itself currently assumes a message to consist of y cells? To summarize, this approach is far from straightforward.

Alternatively, messages could be chosen to correspond to cells. Note that in a way, this correspondence is already inherent in the Tor design, since Tor subjects cells to layered encryption in order to achieve bit-wise unlinkability while mix networks do the same with their messages (see section 2.2). A potential downside of this mapping is that cells are quite small (514 bytes at the time of this writing). A message in Loopix is 2048 bytes in total [20, 76]. The smaller messages are, the more overhead we expect to be incurred by mixing when transferring a certain amount of data. Tor's cell size however is a parameter that can be configured.

Taken together, we consider mapping mix network messages to Tor cells to be the more straightforward option, and in the following assume this mapping to hold.

3.2 Exponential Delays

Recall that in the Loopix system, mix nodes apply exponentially distributed delays to message forwarding so that each of the messages present at a mix node at any point in time is forwarded next with equal probability [78]. Thus, Loopix employs continuous-time mixing in the tradition of Kesdogan [58] (see section 2.3.1). In this section, we highlight the relationship between exponential delays and Poisson-shaped ingress traffic in continuous-time mixing. Further, since exponential delays constitute the core component of Loopix' mixing strategy, a discussion of the general tension between Tor's performance-aware cell forwarding and the implementation of a mixing strategy is also located in this section.

3.2.1 Exponential Delays and Message Arrival Rate

In introducing his mixing strategy, Kesdogan assumes ingress traffic to follow a Poisson process. In his proof of exponentially distributed delays being optimal in continuous-time mixing, Danezis [21] likewise makes this assumption. The Loopix design shapes traffic so that the assumption is fulfilled. The reason why continuous-time mixing couples exponential delays with Poisson-shaped ingress traffic is queuing theory, which forms the mathematical basis of the mixing strategy's security argument. Each mix in Kesdogan's system, respectively Loopix, is conceptualized as a queue with exponentially distributed arrival rate and service time. The traffic rate therein corresponds to the arrival rate, while the delays are mapped to the service time (once the delay applied to a specific message has elapsed, it leaves the mix and is thus considered "serviced"). Applying this mathematical model to the mixes allows for precise computation of the expected number of messages stored at a mix at any given point in time [66], which in turn tells us how many messages a message received by a mix is indistinguishable among. Without an exponentially distributed arrival rate, the model no longer applies, and the straightforward computation of indistinguishability guarantees based on queuing theory is no longer possible. In section 3.3 we examine the possibility of shaping Tor's traffic according to a Poisson process.

3.2.2 Mixing and Tor's Performance-Aware Cell Forwarding

Tor relays currently do not perform any kind of mixing. By contrast, the iteration of onion routing that immediately preceded Tor demanded that relays emit all cells having arrived within a fixed time interval in a pseudo-random order to limit the temporal linkability of cells entering and leaving a relay [85, 101]. The cell re-ordering strategy was designed to preserve the order of cells belonging to the same connection [102]. According to Syverson [98], the reason for not including any mixing in Tor was the absence of a theoretical justification for a benefit against an active adversary.

The ultimate purpose of mixing is to increase an adversary's uncertainty in associating an ingress message with an egress message. In essence, it is ensured that many of the messages forwarded by a mix node are sufficiently likely to correspond to a specific message previously received by that mix node. The target message is thus indistinguishable within a set of a certain size. The expected size of this set can typically be computed based on the mixing strategy's parameters, i.e., security guarantees can be quantified. It is therein implicitly assumed that message forwarding at a mix node is exclusively determined by the mixing strategy. In Loopix, message forwarding is dependent on the delay specified for the respective message and hop, and nothing else. This is a necessary condition for exploiting the memoryless property of the exponential distribution: if delays are exponentially distributed and message forwarding is contingent on nothing except the respective delays, then each message present at a mix node is equally likely to be emitted next, resulting in indistinguishability of all messages stored at a mix at a given point in time.

How could Loopix' cell forwarding behavior be emulated in Tor? Naively, we might imagine starting a timer based on the respective delay whenever a cell is received at a relay, and sending the cell onward in the direction of the next hop immediately once the timer has elapsed. Notice the extreme simplicity of cell forwarding implicitly required by this behavior.

Yet, cell forwarding at Tor relays is a fairly complex matter. Within a circuit, cells are emitted in a first-in, first-out (FIFO) fashion so that the order of cells entering a circuit at the entry node corresponds to the order of cells leaving the exit node. Concretely, each relay keeps two FIFO cell queues for each circuit traversing it (one for each direction of traffic, client $\rightarrow$ destination as well as destination $\rightarrow$ client). Among all circuits belonging to any writable socket, the one that gets to write cells to the network next is determined according to an exponentially-weighted moving average (EWMA) algorithm developed by Tang and Goldberg [104]. When selecting a circuit to flush cells from, the one with the lowest EWMA of cells sent is picked. As a result, newly created and bursty circuits are prioritized and circuits with bulk traffic are kept from clogging connections. This ensures that delays are kept small in the case of interactive traffic such as web browsing, at the expense of slowing down more delay-tolerant traffic such as file transfers. Jansen et al.'s *Kernel-Informed Socket Transport* (*KIST*) algorithm [50, 52, 53] further determines how much data is written to each socket based on real-time kernel information and TCP state information. The introduction of KIST has reduced circuit congestion in Tor, thereby achieving lower network latencies and increased throughput.

In summary, cell forwarding in Tor is optimized to maximize performance and user experience. This behavior is well-aligned with Tor's overarching goal of attracting a large user base by being accessible and fast, and thus offering each user many fellow users to hide among. Tor's performance-aware cell forwarding is however at odds with the introduction of a mixing strategy, whose efficacy depends on forwarding exclusively being contingent on its parameters. Also, a mixing strategy that does not leave the order of cells within a circuit intact would necessitate additional mechanisms for restoring this order, thereby introducing further overhead.

3.3 Shape of Ingress Traffic / Cover Traffic

In section 3.2.1, we explain why continuous-time mixing strategies couple exponential delays with traffic shaped according to a Poisson process. In this section, we discuss how Tor's traffic may be molded to fit this shape. Moreover, since Loopix makes use of cover traffic to achieve the desired traffic shape, we proceed to describe Tor's cover traffic framework.

3.3.1 Homogenizing Tor's Traffic

The majority of traffic transported over the Tor network is interactive HTTP traffic [48, 51]. This type of traffic is *bursty*: periods in which many cells are emitted in quick succession are followed by periods of inactivity. The Loopix design requires the durations of intervals in between sending two messages to follow an exponential distribution with parameter λ. The interactive TCP traffic typical of Tor would have to be *homogenized* to fit into this shape, i.e., we would have to lower the traffic rate's variance.

As outlined by Shmatikov and Wang [94], homogenizing interactive TCP traffic comes at a high cost. For example, implementing a defense so that users appear to send at a constant rate requires padding respectively delaying traffic to enforce this rate. Either cover traffic is introduced so that traffic is always emitted at the highest rate naturally occurring in the system, which incurs prohibitive cost in terms of bandwidth. Alternatively, picking a lower constant traffic rate to save bandwidth means having to artificially decrease the traffic rate in bursty periods, thus incurring cost in terms of delay.

In the same manner as sending at a constant rate, enforcing exponentially distributed inter-emission intervals is a traffic shaping strategy that does not take the form of the underlying application traffic into account. Observe that sending in intervals sampled from an exponential distribution with mean x means that on average, a message is emitted every x units of time, as it would be in the case of sending at a constant rate. We therefore expect the cost incurred by homogenizing bursty traffic using Loopix' traffic shaping strategy to be considerable for the reasons detailed above. In chapter 4, we provide an empirical analysis of the overhead introduced by applying Loopix' traffic shaping strategy to Tor traffic.

3.3.2 Tor's Padding Framework

Initially, Tor did not include any *padding* (cover traffic) [100]. The introduction of padding was deferred to a future time when a convenient strategy providing provable security against a realistic adversary were available. In 2019, the Tor project published a new release containing a padding framework [65]. Tor now supports cover traffic on the level of individual connections between two nodes (a Tor client and a relay or two relays) as well as on the level of entire circuits [74]. The two levels of padding are orthogonal to each other and each serve a different purpose.

The goal of padding individual links between two nodes consists in reducing the resolution of connection-level metadata. Internet routers periodically report this metadata to a "collector", from where ISPs and surveillance infrastructure gather it for further analysis. Since metadata for a specific connection is reported more frequently and thus with higher granularity if the connection is inactive for some number of seconds (to reduce memory requirements at the router), padding is inserted strategically into momentarily idle connections to avoid them appearing inactive. By default, Tor injects connection-level padding into the links between clients and entry nodes [74].

Circuit padding in Tor conceptually builds onto the *Adaptive Padding* approach first suggested by Shmatikov and Wang [94]. Adaptive Padding is a defense against *timing analysis* attacks, developed for low-latency mix networks. In a timing analysis attack, an adversary correlates inter-packet intervals on two links to determine if they belong to the same route. The defense is implemented by having intermediate nodes inject cover traffic into statistically unlikely gaps in the packet flow. Suitable instants for cover traffic injection are therein determined based on the statistical distribution of inter-packet intervals for a "normal" flow. The respective notion of a "normal" flow may be based on external data or learned by the intermediate nodes over time. This implies that the defense's performance may be subpar in the case of flows whose inter-packet interval distributions deviate substantially from the assumed one. Compared to deterministic defenses such as sending at a constant rate, adaptive padding only provides probabilistic protection; its low cost, in particular the fact that it does not delay application data at all however makes it very attractive. Shmatikov and Wang's defense is designed to pad all links in a communication path, except the link between the initiator and the first node (cover traffic is injected by all intermediate nodes). Intermediate nodes are unable to recognize dummy packets as such, while the destination is required to detect and drop them. The defense is intended to provide protection against an adversary attempting to associate *entry* links with their respective *exit* links.

Based on the idea of Adaptive Padding, Juarez et al. [56] develop *WTF-Pad* (short for *Website Traffic Fingerprinting Protection with Adaptive Padding Defense*). This defense is designed to protect a Tor client's entry link and implemented as a *Pluggable Transport*. Pluggable transports are plugins for Tor designed to obfuscate the traffic between a Tor client and the Tor node through which it connects to the Tor network [67]. The associated attacker model is displayed in 3.1. The goal of these plugins is to make it hard to distinguish Tor traffic as such, for example by molding it into a form typical of an innocuous protocol such as Skype. An

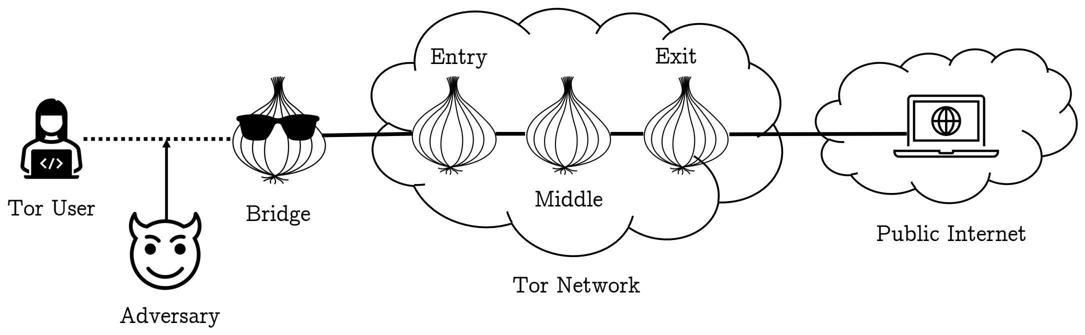


Figure 3.1: Pluggable Transport Attacker Model. A Tor client connects to the public Internet via the Tor network. The first node on the path is a *bridge*, a special Tor node not listed in the official Tor directory and thus (hopefully) unknown to the adversary. Client and bridge both run a Pluggable Transport in order to protect their shared link, on which the adversary is observing traffic. Diagram adapted from Juarez et al. [56]

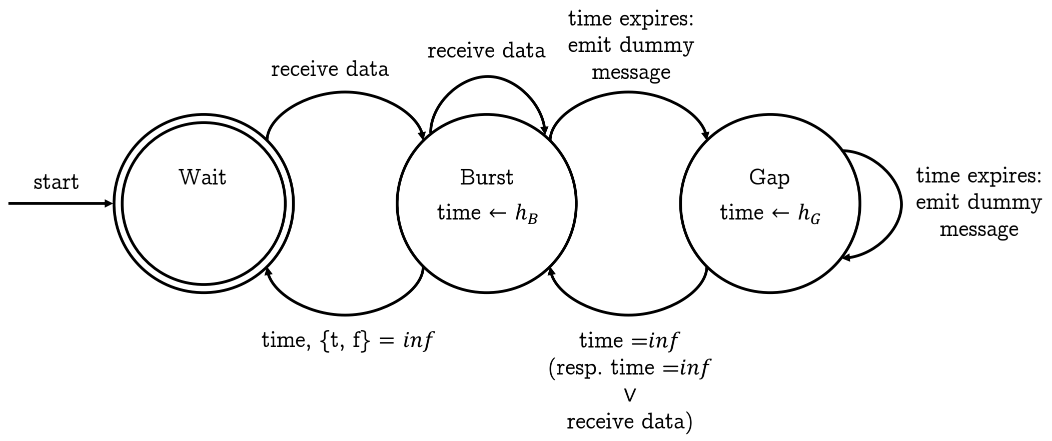

time expires: emit dummy message
receive data
receive data
start
Wait
Burst
time ← h_B
Gap
time ← h_G
time expires: emit dummy message
time, {t, f} = *inf*
time =*inf* (resp. time =*inf* ∨ receive data)

Figure 3.2: WTF-Pad State Machine. h_B = histogram burst, h_G = histogram gap. By observing application traffic, the machine decides whether patterns are typical of a burst or a gap and transitions between states accordingly. In gap mode, it inserts cover traffic to produce a fake burst. h_B and h_G contain empirical measurements of inter-emission intervals typical of bursts respectively gaps.

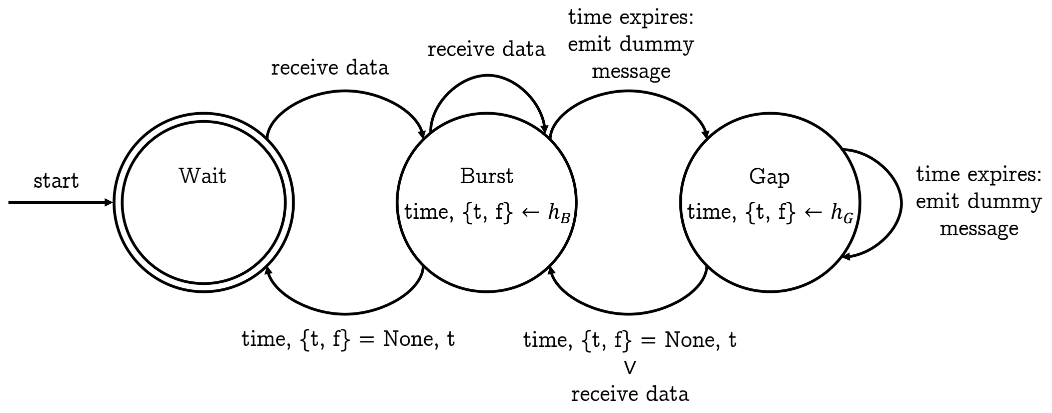


Figure 3.3: APE State Machine. h_B = histogram burst, h_G = histogram gap. The machine operates in the same general manner as the WTF-Pad machine. h_B, h_G are represented by randomized log-normal distributions rather than based on empirical measurements. The ∞-bins of WTF-Pad's empirically constructed histograms are replaced by probabilistic transition decisions.

important application is to enable Tor users to access the network from a region in which Tor is blocked (e.g., a country ruled by a totalitarian regime). A Pluggable Transport requires collaboration of as well as trust in the protected link's other endpoint. On the protocol stack, a Pluggable Transport is located in between the Tor process on the user's machine and the transport layer. WTF-Pad builds on the observation that website fingerprinting attacks often recognize webpages based on their specific patterns of *bursts* (defined as sequences of packets that have been sent in a relatively short time) and *gaps* (defined as sequences of packets that have been sent over a longer timespan). The key contribution of WTF-Pad is to implement the defense as an event-driven finite state machine (figure 3.2). By observing real application traffic, the machine transitions between states representing burst mode and gap mode. Bursts and gaps are distinguished by data chunks' inter-arrival times. The distributions of inter-arrival times characteristic of bursts respectively gaps are encoded as binned histograms constructed based on empirical measurements.

At the beginning of its operation, the state machine enters burst mode as soon as a piece of data arrives. It then samples an inter-arrival interval from the burst histogram. Should a piece of data arrive before the interval elapses, the machine stays in burst mode and draws a fresh sample. Should the interval elapse before a piece of data has arrived, the machine transitions into gap mode. While in gap mode, it samples inter-arrival intervals from the gap histogram and emits a dummy packet at the end of each sampled interval. A sample from the burst respectively gap histogram can assume the value ∞. Should this happen, the machine transitions from gap mode to burst mode respectively from burst mode to wait mode. This mechanism ensures that the machine exits probabilistically after a transfer has ended. If desired, the machine can additionally be configured to transition from gap mode to burst mode upon receiving a piece of data. Each participant is assigned *two* state machines that each observe application traffic and insert dummy traffic. One of these machines observes *upstream* traffic, i.e., traffic arriving locally from the application on the user's computer. The other machine observes *downstream* traffic, i.e., traffic arriving from the communication partner. This allows for the simulation of the request-response patterns typical of HTTP.

The complexity of histogram construction constitutes the main limitation of WTF-Pad: since inter-arrival time distributions depend on factors such as the user's bandwidth and change over time even for the same constellation of influencing factors, up-to-date measurements have to be carried out frequently to preserve the defense's performance. To address this difficulty, Pulls proposes the *Adaptive Padding Early (APE)* defense [79]. APE replaces the empirical burst/gap histograms with randomized log-normal distributions. Whenever a sample is requested from one of these distributions, the machine additionally randomly decides whether or not (transition decision $\in \{T, F\}$) to transition between states (represented by the $(None, T)$ result in the diagram). This accounts for WTF-Pad's ∞-samples. APE's state machine design is displayed in figure 3.3. It is designed explicitly as a cheap and naive alternative to WTF-Pad that should be abandoned as soon as a practical approach to histogram generation for WTF-Pad has been developed. The APE defense is available as a Pluggable Transport.

Tor's circuit-level padding framework has adopted the conceptualization of traffic analysis defenses as event-driven state machines from WTF-Pad. A detailed developer documentations is available [73]. The framework is highly flexible, allowing for the specification of state machines reacting to arbitrary events in the system. The inter-transmission delays characterizing cover traffic generation in a specific state can be defined using custom histograms or parameterized probability distributions (i.e., the behavior of both WTF-Pad and APE can be encoded). Padding can take place between the client and an arbitrary hop along the circuit. By default, circuit padding state machines for obfuscating client-side onion service setup as well as for hiding client-side rendezvous circuits are currently active in Tor [74]. Performant circuit padding state machines for defending against website fingerprinting attacks developed by Pulls [82] using genetic programming are also available.

The Tor developers oppose the introduction of any delays due to concerns regarding technical hurdles as well as user experience [73]. In light of this, it stands to reason that their efforts at strengthening Tor's anonymity guarantees are focused on cover traffic. Adaptive Padding in theory does not delay application traffic[1] and only introduces comparatively low bandwidth overhead due to being probabilistic. While the primary goal of Adaptive Padding in Tor currently consists in defending against website fingerprinting attacks [73], building end-to-end defenses using the circuit-padding framework is conceivable.

3.4 Path Selection

The Loopix design as well as its ancestor, Kesdogan's SG-Mix, require each message to be sent along an independently selected path through the network. Of course, the primary intention of this requirement is to limit the linkability of messages based on their respective paths. In this section, we analyze how independent random path selection additionally serves

[1]Witwer et al. [110] demonstrate that network-wide deployment of Adaptive Padding in Tor does in fact introduce higher latencies as it increases the lengths of cell queues.

to ensure that the continuous-time mixing strategy's privacy guarantees can be calculated precisely. We give a detailed overview of how Tor selects paths through the network and explore if and how we could fulfill the requirement that each "message" be sent along an independently selected path in the context of Tor. Lastly, we argue that uniformly random path selection in Tor is desirable only in support of a mixing strategy, but not on its own.

3.4.1 Selecting a Fresh Path for Every Message

What would happen if we removed Loopix' assumption of uniformly random path selection, all else being equal? As before, users then continuously emit a mix of real and cover traffic in the shape of a Poisson process, and mix nodes forward each message with a delay sampled from an exponential distribution. The parameters μ and λ of the exponential distributions governing traffic emission and message delays are public and the same for all system users. Without the requirement that a path through the network is picked uniformly at random for each message, we lose the guarantee that the expected rate of ingress traffic is equal across all mix nodes [58]. The mixing strategy however relies on the rate of traffic received by each mix node in the system following a Poisson process with the same parameter. Only if this is the case does the message indistinguishability argument described in section 2.6 hold. Uniformly random per-message path selection thus not only hinders correlating messages based on their respective paths, but further ensures that anonymity guarantees can be computed precisely. The extent of deviation from the assumption of equally distributed arrival rates when independent per-message path selection is removed from the design evidently depends on the respective replacement path selection policy.

3.4.2 Circuit Management in Tor

A number of circuit types with different purposes exist in the Tor protocol. Our focus in this analysis are *general-purpose* circuits, the most common type of circuit. Its purpose is the exchange of data between a Tor client and a destination located outside the Tor network. A general-purpose circuit commonly consists of three intermediate relays: entry, middle, and exit. In the following, we use the term *circuit* to refer to this type of circuit. Further, we use the terms *circuit* and *path* synonymously in the context of Tor.

While Loopix selects paths through the network uniformly at random, path selection in Tor takes many aspects into account [33]. Only relays whose uptime and bandwidth surpasses certain thresholds are considered for the entry position of a circuit. Eligible exit nodes have to allow connections to the types of external destinations a client intends to connect to. Any available relay may be picked for the middle position. When selecting a node for a specific position in the circuit (entry, middle, exit), possible candidates are weighted according to their bandwidth (with some load-balancing being applied). Further, a number of policies ensuring sufficient network diversity across the three nodes in a path are enforced (e.g., no two nodes may be located in the same subnet).

Tor's circuits are typically composed of three hops since this circuit length is considered the smallest providing sufficient protection [98]. In fact, onion routing circuits did not always consist of three intermediate nodes: the first generation used circuits of length five, while circuit length was variable in the second generation. Longer circuits however result in higher latencies, which is why the Tor design opted for the minimal safe length. Yet, the Tor authors list the choice of path length as an open question in onion routing [100]. Choosing path length randomly from a geometric distribution is suggested to thwart corrupt onion router's efforts at inferring their location on a circuit based on timing.

Tor's goal is to protect against an adversary assumed to control some fraction of the network. If the entry and exit nodes of each circuit were chosen uniformly at random, then an adversary controlling $\frac{k}{N}$ of the network would be able to de-anonymize $(\frac{k}{N})^2$ of all circuits [62]. This is because since Tor relays do not perform any kind of mixing, a stream of traffic retains its shape on its way through the network. An adversary can thus trivially match a sender with a receiver if they can observe both ends of the communication (i.e., the links between client and entry node as well as exit and destination). Therefore, each Tor client picks a set of three eligible relays as their *entry guards* and chooses the first hop of every circuit it constructs from among the nodes in this set. Each relay in a client's entry guard set is replaced after a random period of time between 30 and 60 days has elapsed. The rationale is that if none of the client's entry guards is compromised by an adversary, the client is safe from de-anonymization attacks at least until the first relay in the set is replaced [36].

When an application on a user's machine first connects to a destination over Tor, the Tor process picks a suitable circuit for this connection. Circuits are built preemptively in the background to ensure that in the majority of cases, a ready-to-use circuit is available instantly once required.[2] If multiple circuits fitting the connection's characteristics are present, the candidate circuits are compared to determine the best match. Time of creation and time of initial usage are the two criteria applied for comparison: if none of two compared circuits has been used yet, the more recently created one is preferred; if one of two circuits has already been used whereas the other has not, the unused circuit is preferred; and if both of two compared circuits have already been used, the one whose time of initial usage is more recent is preferred. One connection sticks to the circuit it has been assigned for the entire duration of its lifetime. Circuits are expired once they have been unused for too long, or if they do not have any streams attached and too much time has elapsed since they have been used first. Due to the selection policy preferring fresher circuits, it is ensured that old circuits are terminated eventually.

Tor's circuit selection policy implies that all connections that are opened within a certain period of time are assigned to the same circuit: a circuit continues to be ranked best as in most recently used for the first time respectively most recently created until a freshly constructed circuit becomes available. Each circuit (as viewed from the client's perspective) is thus usually shared by multiple TCP streams (connections). Continuously building new

[2]Occasionally a circuit has to be launched ad-hoc if none of the available circuits match a connection's requirements.

circuits and expiring old ones however results in a cap on the number of requests that can be linked based on using the same circuit.

3.4.3 Increasing Tor's Circuit Switching Frequency: How?

What would happen if we enforced circuit switching for each cell without changing the circuit selection policy? We likely would not achieve much: it is to be expected that in many cases, in between picking a circuit for a connection and sending one cell along it no fresh circuit would have become available. Hence, the "old" circuit would still be considered best, and switching circuits would merely result in re-assigning the original circuit to the connection. If we changed the circuit selection policy, e.g., so that each of the active circuits is equally likely to be picked, it is to be expected that the circuits used by a client would never be expired since they would not cease to have streams attached to them. New circuits may still be built if the Tor process does not deem the available circuits sufficient for handling all anticipated ports. Anonymity degradation may result from each circuit staying active for a potentially lengthy period of time (until the Tor process is restarted), thereby increasing the number of linkable streams compared to Tor's regular operation.

Based on these considerations, it seems that merely increasing the circuit switching frequency would not be very helpful, even if the circuit selection policy were adapted as well. In order to decrease the number of streams that can be linked based on using the same circuit, Tor's circuit selection policy would have to be kept as it is while increasing the frequency of circuit construction and circuit switching. Building a new circuit is however relatively expensive as it involves opening TCP connections between nodes as needed and negotiating a symmetric key with each hop along the circuit, thus requiring several round-trip times of network communication as well as public key cryptography. Further, re-ordering may result from packets belonging to the same communication flow being routed along different circuits. Restoring the correct order within Tor or on the application layer would incur additional costs.

The Tor designers [100] suggest two cheaper alternatives to constructing fresh circuits. Firstly, Tor's *leaky-pipe* circuit topology allows clients to send data to intermediate hops along a circuit. This is achieved by applying onion encryption only up to the layer corresponding to the destination hop, who thus understands itself to be the target. With leaky-pipe, a three-hop circuit in theory provides three sub-paths through the network. However, this is contingent on all intermediate nodes being willing to forward traffic to the respective destination outside the network. The second suggested low-cost alternative to fresh circuit construction is to truncate and re-extend existing circuits. The approaches based on leaky-pipe respectively truncating and re-extending naturally yield paths that are correlated with the original paths they were built from.

How can Loopix efficiently use an independently selected path for each message? Firstly, Loopix uses UDP as its transport layer protocol. Since UDP is connectionless, no sessions need to be torn down or established when switching paths. Secondly, Loopix does not require the negotiation of symmetric cryptographic keys. This is because Loopix leverages the *Sphinx* [25] cryptographic packet format. With Sphinx, the symmetric key shared between two nodes can be derived from their respective RSA keys and information contained in the packet header. Thus, no interaction is required prior to exchanging application data. It has been suggested to employ the Sphinx format in onion routing circuit construction as well [57].

3.4.4 Increasing Tor's Circuit Switching Frequency: Why?

Based on these considerations, we expect more frequent circuit switching to be rather costly. Not only that, observe that even if we picked a fresh circuit for every single cell, this would not amount to selecting paths uniformly at random from all possible paths in the network. Firstly, as described above suitable relays are selected based on a number of properties when constructing circuits, rather than randomly. Secondly, due to Tor's entry guard feature, the first hop in each circuit is picked from among the same three relays until one of the entry guards is replaced. Finally, traffic transported over Tor is typically bi-directional, with both directions of a stream traversing the same circuit. Since external destinations addressed via Tor are not assumed to run any software related to Tor, they cannot build separate circuits along which to route their replies.

Beyond the question *if* and *how* uniformly random path selection can be achieved in Tor, we need to ask ourselves *why* we would want to achieve it. In Loopix, intermediate nodes obfuscate traffic patterns by mixing messages to defend against an adversary assumed to have global view of the network. Selecting each message's path uniformly at random ensures the proper functioning of the mixing strategy. The adversary anticipated by Tor's designers is partial. Since no mixing whatsoever is performed by intermediate nodes, Tor's security hinges on the adversary not being able to observe both ends of a communication flow. As illustrated by the argument behind the entry guard feature, more variation in the entries and exits used by a client then means increasing the risk of de-anonymization. Deviating from uniformly random path selection is thus a security feature in anonymous communication systems that assume a partial adversary [111].

Taken together, Tor's path selection not being uniformly random is a feature, not a bug. Firstly, allowing relays to specify if they are willing to make connections to external destinations, and if so which ones, is vital since Tor relays are run by volunteers. As the exit node is the one making the actual connection to a destination, it needs to be able to disallow connections to destinations deemed objectionable or risky. Secondly, taking relays' bandwidth into account when selecting nodes for a circuit supports Tor's performance and respects relays' restrictions regarding how much bandwidth they are willing or able to donate to the network. Thirdly, entry guards were deliberately introduced to lower the risk of de-anonymization under the circumstances of no mixing and a partial adversary. Only if a mixing strategy designed to defend against a global adversary were put into place may uniformly random path selection support anonymity guarantees rather than increase the risk of exposure.

Chapter 4

Empirical Analysis

In chapter 3, we examine the main challenges concerning the integration of Loopix with Tor from a conceptual point of view. Among these challenges, we select the shaping of ingress traffic according to a Poisson process for further empirical assessment.[1] We choose this challenge for closer inspection for two reasons. Firstly, all users continuously emitting traffic in the shape of a Poisson process lies at the heart of the Loopix design: Poisson-shaped ingress traffic and exponentially distributed forwarding delays in conjunction allow for the quantification of anonymity guarantees using the mathematical model of queuing theory. Secondly, a large portion of the traffic transported over Tor is interactive HTTP traffic [48, 51], which produces a pattern of alternating bursts and gaps. Loopix' traffic shaping strategy molds egress traffic without taking the form of the application traffic to be emitted into account. We expect the application of such a homogenization technique onto Tor's bursty traffic to incur a large overhead and thus constitute the biggest hindrance to integrating Loopix into Tor.

In our analysis, we measure the overhead incurred by molding the type of traffic characteristic of the Tor network to conform to this shape. Moreover, we analyze the overhead emerging

[1]Recall that a Poisson process is a series of events occurring at exponentially distributed intervals.

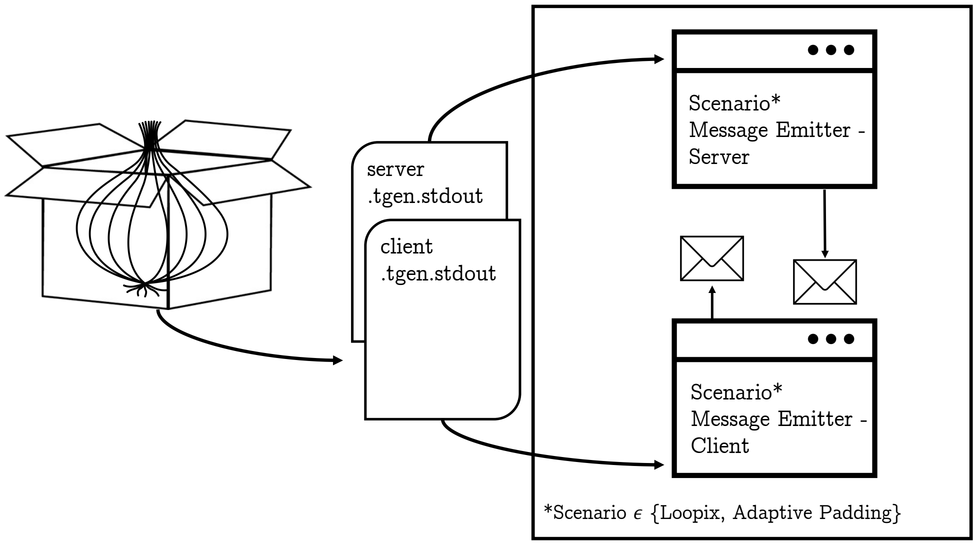


Figure 4.1: Main Experiment Design (High-Level Overview). `tgen`-over-Tor traces are collected using the `Shadow` Discrete Event Network Simulator. These traces are used to generate traffic in an abstract simulation setting in which we investigate the overhead incurred by the two scenarios' traffic shaping strategies.

from protecting the same traffic with a defense from the Adaptive Padding family. Our reasons for including an Adaptive Padding defense in our analysis are twofold. Firstly, since Adaptive Padding represents the cover traffic generation strategy that inspired Tor's cover traffic framework, we deem it a suitable reference point. Secondly, Adaptive Padding defenses for Tor are commonly analyzed in the context of defending against website fingerprinting attacks. However, Adaptive Padding is not limited to protecting HTTP traffic [94]. We thus consider it of interest to explore its overhead when applied to general Tor traffic rather than just web browsing traffic specifically.

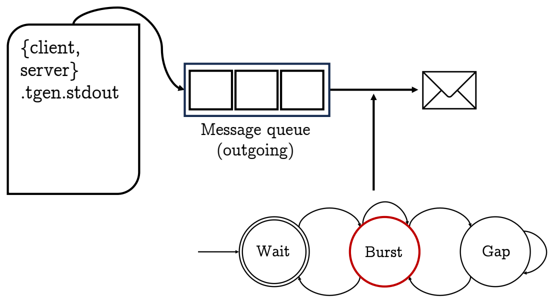
4.1 Main Experiment Design

The design of our main experiment is illustrated in figure 4.1. We design our experiment set-up with the intention to isolate the traffic shaping strategies' effects on the emission behavior of communicating client-server pairs. A traffic shaping strategy's effectiveness requires that emission behavior is governed by it exclusively. Since achieving this in the context of a complex system such as Tor is highly non-trivial, we create an extremely abstract set-up in which participants are reduced to entities emitting a specific background load according to one of the strategies under examination. The patterns of background load therein correspond to the shape of traffic typically carried over the Tor network, and data is emitted in the form of fixed-sized 514-byte cells. We explicitly abstract from the underlying communication network (and operating system). Further, we leave open the actual location of the server side of the defenses (since the Tor network does not expect destinations to participate in the Tor protocol, the remote end of a traffic shaping defense may for example be located at an exit relay).

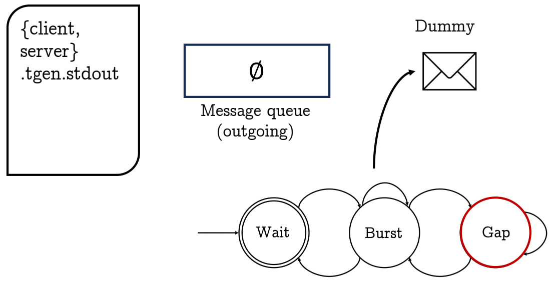
To obtain baseline traffic representative of the Tor network, we rely on existing models that were learned (in a privacy-preserving manner) at a number of Tor relays [51]. We leverage the `tgen` (v1.1.1) utility, which generates application traffic according to these models in the context of simulations carried out using the `Shadow` discrete event network simulator [47, 49]. We add additional logging to `tgen`'s source code to be able to reconstruct its behavior from the respective `.stdout` files. Using `Shadow` (v3.0.0), we collect 4000 traces of `tgen`-over-Tor communications between client-server pairs. Communicating parties are therein configured to interact with each other exclusively. Each interaction represents a single *flow* consisting of a varying number of streams. In `tgen`'s semantics, a flow is a generator of streams [51]. `tgen`'s flow generation model is based on Tor users' circuit building behavior. While Tor assigns new streams to available circuits, `tgen`'s flows *generate* streams; both a circuit and a flow thus produce a logical grouping of a set of streams.

Using the `SimPy` (v4.0.2) discrete event simulation library for `Python 3`, we simulate each of the 4000 `tgen` traces in two different scenarios:

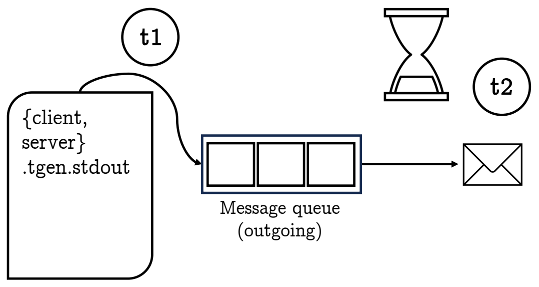
1. **Adaptive Padding**: in this scenario, each piece of data (chunked into 514-byte-cells) is emitted at the precise time it is scheduled for according to the underlying `tgen` trace



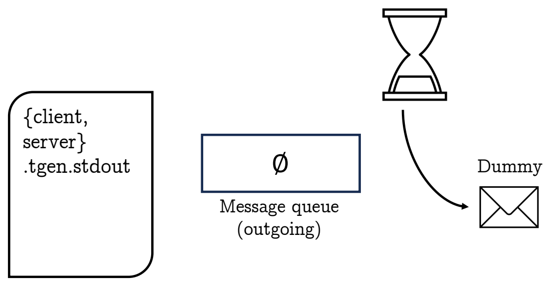
(a) Emission of Data in the Adaptive Padding Scenario. The state machine governing cover traffic emission observes that application data is emitted, and cover traffic insertion is therefore not necessary.



(b) Emission of a Dummy Cell in the Adaptive Padding Scenario. The state machine governing cover traffic emission observes that there is a gap in application traffic, and begins to insert cover traffic to hide this gap.



(c) Emission of Data in the Loopix Scenario. At t_1, a cell containing application data is put into the queue. At t_2, the next inter-emission interval expires, and the cell is emitted.



(d) Emission of a Dummy Cell in the Loopix Scenario. At the time the inter-emission interval expires, the message queue is empty and a dummy cell is emitted.

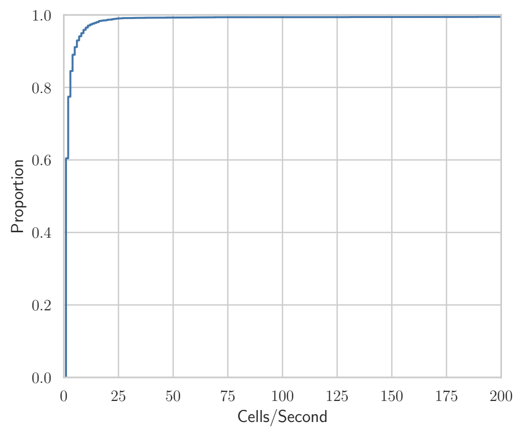
Figure 4.2: Overview of Data- and Dummy Cell Emission Behavior in the Adaptive Padding- and Loopix Scenarios.

(figure 4.2a). Further, two APE (Adaptive Padding Early) state machines (see section 3.3.2) per participant govern the emission of dummy cells for obfuscating gaps in application traffic (with one machine observing traffic arriving locally from the application, and the other observing traffic arriving from the communication partner). Recall that APE state machines operate by observing egress- respectively ingress traffic and deciding whether inter-emission- respectively inter-reception intervals are characteristic of a burst or a gap. While in gap mode, an APE state machine induces an artificial burst by inserting cover traffic (figure 4.2b). Our implementation is based on the APE Pluggable Transport, which is included in the *basket2* series of obfuscation tools [80].

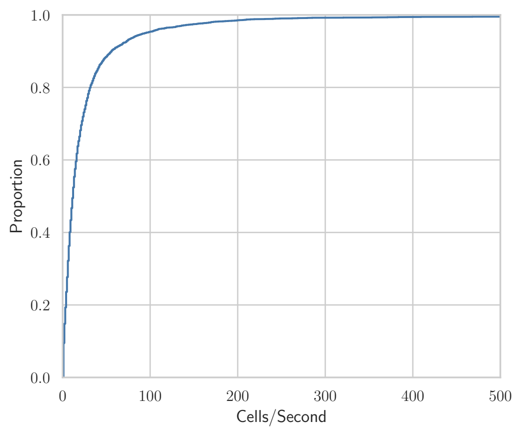
2. **Loopix**: in this scenario, cells are emitted according to a Poisson process, i.e., in intervals sampled from an exponential distribution with a specific scale parameter. Once a sampled interval expires, a cell containing application data is emitted if available (figure 4.2c); otherwise, a dummy cell is emitted (figure 4.2d). Our implementation is based on the Loopix prototype [76].

Among the available Adaptive Padding defenses, we choose to examine the APE design rather than the WTF-Pad design to circumvent the empirical construction of histograms representing inter-arrival times characteristic of bursts respectively gaps in traffic. Further, since APE is available as a Pluggable Transport, we deem it to constitute a better fit for our highly abstract experiment setting compared to more recent designs [82] developed directly within the semantics of Tor's circuit padding framework.

The Loopix scenario requires parametrization with the mean of the exponential distribution from which inter-emission delays are sampled. To determine a suitable range of values for examination, we firstly compute the median cell rates of clients and servers in a baseline scenario, in which each piece of data (chunked into 514-byte-cells) is emitted at the precise time it is scheduled for according to the underlying `tgen` trace (figure 4.2a). For clients, the median cell rate corresponds to 1 cell (4112 bits) per second, whereas for servers it amounts to 12 cells (49344 bits) per second (figure 4.3). Based on this, we select 25 evenly spaced bit rates



(a) Cells/Second Emitted by the Client in the Baseline Setting. med. = 1.0, $\sigma = 12.15$.



(b) Cells/Second Emitted by the Server in the Baseline Setting. med. = 12.0, $\sigma = 39.52$.

Figure 4.3: Cell Rates in the Baseline Setting.

from an interval bounded by the extremes 1371 bits (one cell every 3 seconds) and 102800 bits (25 cells per second). The 25 bit rate values are then converted back to (mean) inter-emission intervals based on the formula *inter-emission interval* $= 1/(\text{bit-rate}/8 \cdot 514)$ *seconds*.

We quantify overhead in terms of two dimensions: latency (resulting from intentionally delaying traffic) as well as bytes (resulting from the insertion of cover traffic). Both are determined per flow via comparison with a baseline setting in which no traffic shaping is performed. In the Adaptive Padding scenario, a flow is considered complete once both communication partners have emitted and received all data they are scheduled to emit and receive according to the underlying `tgen` trace. In the Loopix scenario, a flow is considered complete once both communication partners have emitted all data they are scheduled to emit according to the underlying `tgen` trace. In the case of the Loopix scenario, we do not consider data reception events since unlike Adaptive Padding, Loopix' traffic shaping strategy does not incorporate them. To complement the bi-directional perspective onto a communication flow, we also determine overhead from a uni-directional perspective, i.e., per participant (with role $\in \{CLIENT, SERVER\}$), in the Loopix scenario. A flow is then considered complete from a participant's point of view once they have emitted all data they are supposed to emit based on the respective `tgen` trace. This allows us to examine the trade-off between transfer delays and cover traffic with respect to role-specific patterns in traffic emission. In the Adaptive Padding scenario, we do not include a uni-directional perspective since due to the presence of state machines reacting to data received from the communication partner, the defense is inherently bi-directional.

Overhead in terms of bytes is computed per combination of scenario and participant by dividing the number of bytes sent by the respective participant in the respective scenario by the number of bytes sent by the respective participant in the baseline setting.

Similarly, overhead in terms of latency is computed per scenario by dividing time taken to complete a flow in the respective scenario by the duration of the same flow in the baseline setting. In the uni-directional overhead analysis of the Loopix scenario, latency overhead is determined per participant since client and server have a potentially different understanding of a flow's completion time.

4.2 Main Experiment Results

In this section, we present the overhead in terms of latency and bytes introduced by Adaptive Padding (represented by the APE defense) as well as Loopix' traffic shaping strategy found in our main experiment.

4.2.1 Overhead Incurred by the APE Defense (Adaptive Padding Scenario)

In presenting our results regarding the APE defense, we contextualize these with findings concerning the overhead of Adaptive Padding defenses reported in the literature:

1. **WTF-Pad** (Juarez et al. 2015 [56]). The defense is evaluated on the basis of 4000 web crawling traces, which implies that overhead is determined from the client's perspective only. Overhead is determined in the form of latency and bandwidth. Latency is defined as the time required to send and receive all data the client is expected to according to the underlying trace. Bandwidth is defined as the number of bytes transferred in a trace divided by the duration of the trace. Overheads are determined by dividing the latency respectively bandwidth for a trace protected by the defense by the respective metric computed for the same trace with no protection.

2. **APE** (Pulls 2016 [79]). Pulls provides an informal analysis of the APE defense's performance and overhead. The defense is evaluated based on a sample of 20000 web crawling traces, i.e., from the perspective of the client. While no overhead metrics per se are reported, Pulls includes good-put measurements.

3. ***Spring*** and ***Interspace*** (Pulls 2020 [82]). Spring and Interspace are two state machine designs developed within Tor's circuit-padding framework. 20000 web crawling traces form the basis of the evaluation, i.e., overhead is determined from the perspective of the client only. Overhead is reported in the form of bandwidth. Bandwidth and bandwidth overhead are defined as in Juarez et al. Latency overhead measurements are available in another publication which comparing several website fingerprinting defenses [64]. Latency as well as latency overhead are therein defined as in Juarez et al.

Juarez et al. [56] report a median latency overhead of 0% ($\sigma = 0.00$) for the WTF-Pad defense. For APE, latency overhead is not available [79]. Spring and Interspace are reported to incur 0% latency overhead [64]. In our analysis, we find a median latency overhead of 0% ($\sigma = 0.00$), conforming to the expectation that as an Adaptive Padding defense, APE incurs no latency overhead.

Overhead in terms of bytes per participant is displayed in figure 4.4. The WTF-Pad defense's median bandwidth overhead amounts to 77% (from the client's perspective) [55]. Since WTF-

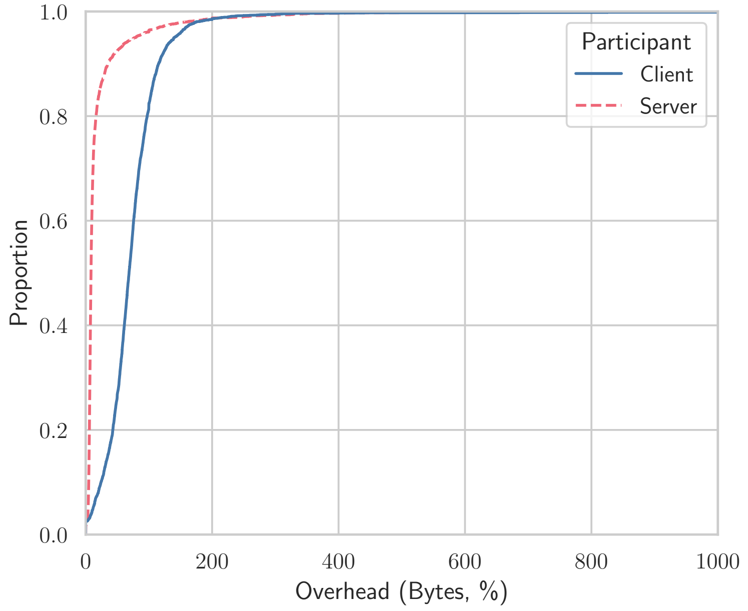


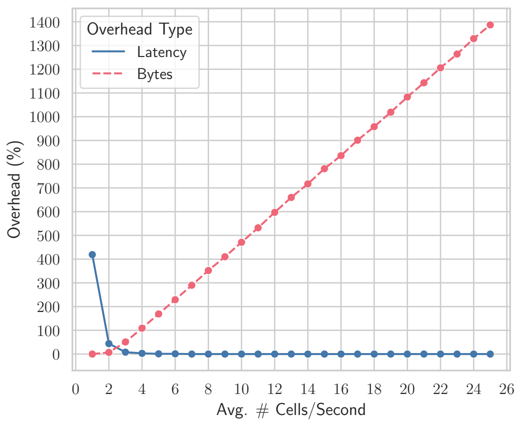
Figure 4.4: Overhead per Participant in Terms of Bytes in the Adaptive Padding Scenario. Client: med. = 68.73 (σ = 48.81); Server: med. = 8.43 (σ = 113.74)

Pad incurs no latency overhead, we can directly compare our overhead in terms of bytes with their overhead in terms of bandwidth. Pulls [79] reports a median good-put of 0.37 for traffic protected by APE, compared to a median good-put of 0.99 with no protection. We convert this to an overhead in terms of bytes corresponding to $(0.99-0.37)/0.37 = 167\%$. The Spring and Interspace machines incur a bandwidth overhead of 210% and 230%, respectively [82]. Again, given that the two machines are reported to incur no latency overhead [64], direct comparability of their bandwidth overhead with our bytes overhead is given. Overall, the bytes overhead (from the client's perspective) we find is thus markedly lower than the values reported in the literature. We attribute this to our traffic model conforming to general Tor traffic rather than web traffic specifically. Not all traffic transported over Tor is interactive: for example, the BitTorrent protocol is the second-most popular application layer protocol used with Tor (after HTTP) [48, 51]. Therefore, we presume the comparatively low bytes overhead found in our experiment to be due to the underlying traces being less bursty than traces obtained using web crawling, consequently requiring an Adaptive Padding defense to insert a lower amount of cover traffic. The bytes overhead from the server's perspective is much lower than from the client's, reflecting the asymmetry of the traffic characteristic of Tor.

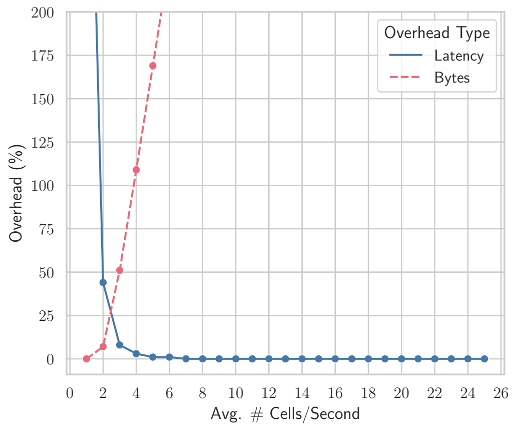
4.2.2 Overhead Incurred by Shaping Traffic According to a Poisson Process (Loopix Scenario)

Recall that in the Loopix scenario, we regard overheads from two different angles. Firstly, we determine overheads *uni-directionally*, meaning that a participant considers a flow complete once they have emitted all data they are expected to according to the underlying `tgen` trace. This allows us to examine the overheads as specific to the characteristic traffic patterns of the two participant roles (i.e., client vs. server). Secondly, we examine overheads *bi-directionally*, meaning that a participant considers a flow complete once *both* communication partners have emitted all data they are expected to according to the underlying `tgen` trace. This perspective enables a more sound comparison with the Adaptive Padding scenario, which is inherently bi-directional (see section 4.1).

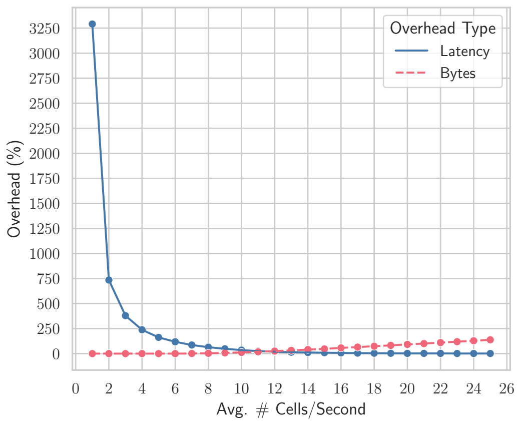
Median overheads per average cell rate in terms of latency and bytes (in %) are displayed in figure 4.5. When viewing flows uni-directionally, the average cell rate at which the two overhead dimensions approach the same value lie in the vicinity of the median cell rate determined for the respective participant in the baseline setting (1 cell per second in the case of the client, 12 cells per second in the case of the server). Note that this corresponds to what we would expect to see when enforcing a *constant* emission rate μ: Whenever the actual cell rate exceeds μ, it has to be slowed down; whenever it falls below μ, cover traffic has to be inserted. Setting μ to the median value naturally occurring in the system thus results in a compromise regarding the two overhead dimensions. Based on our results, a similar



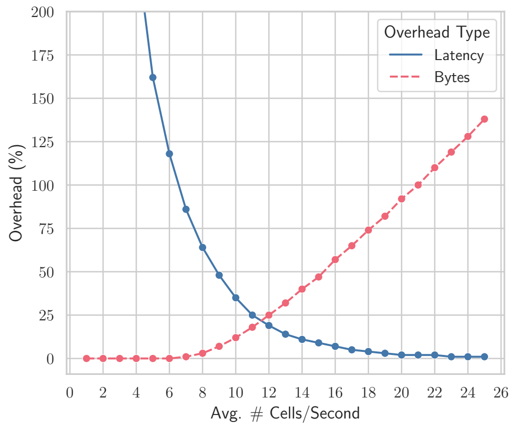
(a) Overhead from the Client's Perspective, Viewed Uni-Directionally.



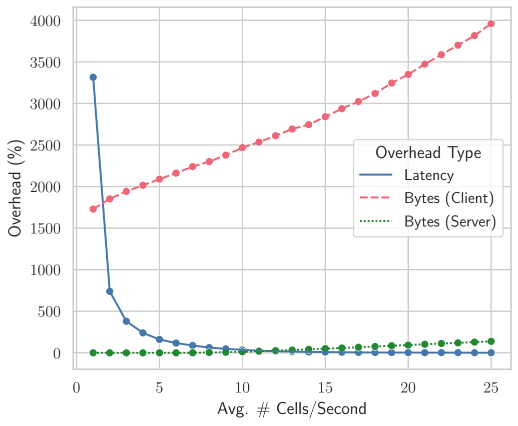
(b) Overhead from the Client's Perspective, Viewed Uni-Directionally (Close-Up).



(c) Overhead from the Server's Perspective, Viewed Uni-Directionally.



(d) Overhead from the Server's Perspective, Viewed Uni-Directionally (Close-Up).



(e) Overhead from the Client's- and Server's Perspective, Viewed Bi-Directionally.

Figure 4.5: Median Overhead (%) in the Loopix Scenario.

argument seems to apply when enforcing exponentially distributed inter-emission intervals sampled from an exponential distribution with mean μ.

Our results show that for participants in the roles of clients respectively servers, different cell rates appear to be optimal in terms of minimizing overhead. All participants emitting traffic at the same average rate irrespective of their role, as required by the Loopix protocol, does not seem feasible at a reasonable cost. If participants are parameterized differently depending on their role, anonymity guarantees cannot be computed precisely anymore. Yet, it is still possible to compute upper and lower bounds on these guarantees.

From the uni-directional perspective, it appears as though the overhead in terms of bytes required to shape egress traffic according to a Poisson process (while avoiding latency overhead) is comparable to the bytes overhead introduced by Adaptive Padding defenses. Recall however that our uni-directional perspective onto the Loopix scenario is not directly comparable with the inherently bi-directional Adaptive Padding scenario.

When viewing flows bi-directionally, a different picture emerges. The median duration of a flow from the perspective of both participants then amounts to 572.74 seconds. Yet, when viewing flows uni-directionally, the median duration of a flow from the point of view of the client by contrast is equal to just 140.64 seconds, while the server generally has the same understanding of a specific flow's duration irrespective of assuming a uni-directional or bi-directional perspective. To understand why, consider the fact that HTTP is the application layer protocol most commonly used with Tor [48, 51]. A large portion of the traffic typically found in the Tor network is thus characterized by a Tor client sending a number of comparatively small requests to a server, who replies by sending comparatively large amounts of data. Considering a flow complete from the client's perspective once it has sent all its requests disregards the fact that in between sending all requests and the flow actually ending there will often ensue a lengthy period of time during which the client merely receives data and therefore has to continuously emit cover traffic to conform to Loopix' traffic shaping strategy. From the bi-directional point of view, we thus find a prohibitive overhead in terms of bytes for participants in the role of the client, especially in comparison with the defenses from the Adaptive Padding family. Recall however that available Adaptive Padding defenses generally only aim to provide unlinkability, while Loopix' traffic shaping is designed to provide unlinkability as well as unobservability.

4.3 Supplementary Analysis of Idle Time

Loopix' traffic shaping strategy demands that participants emit a continuous stream of egress traffic even during lengthy periods spent passively receiving data from communication partners. Our main experiment's results show that this produces a high overhead in terms of bytes. We begin this section by detailing a supplementary experiment gauging the amount of time Tor clients typically spend idle *in between* flows respectively streams (sub-section 4.3.1). Further, we use this supplementary investigation to validate a method for generating Tor traffic in abstract simulation settings such as ours (sub-section 4.3.2).

4.3.1 Gauging Idle Time between Streams and Flows Using `tgen`

In our main experiment, we analyze the overhead of traffic shaping strategies *during an active flow*. The WTF-Pad- and APE defenses exit probabilistically after a transfer has ended (see section 3.3.2). Loopix' traffic shaping strategy by contrast is designed to not only obfuscate patterns in *active* transfers, but to hide the mere fact that a participant is currently engaged in any communication (unobservability). This implies that in idle periods, the overhead incurred by its traffic shaping strategy (in terms of bytes) is much higher by definition. To fulfill the requirement that messages be emitted according to a Poisson process with a specific parameter, cover traffic has to be inserted continuously to hide the fact that a participant is not currently sending any data.

To gauge the fraction of time a Tor client[2] typically spends idle over a specific period, we conduct a supplementary experiment using `tgen`. A client is therein considered idle if no stream actively generating data is currently associated with this client. Note that the main experiment's data is not suitable for gauging idle time (defined in the aforementioned way) since each of the main experiment's traces covers exactly one flow of communication and thus does not capture potential idle periods between flows.

In this supplementary experiment, we collect 940 [3] `tgen`-over-Tor traces using the `Shadow` discrete event network simulator. Each trace represents one Tor client generating flows for a period of two hours of simulated time. We measure a client's idle time as the fraction of time during these two hours in which no stream belonging to this client is active. A stream is considered active for the period of time in between the first and last pieces of payload data being sent on that stream (in either direction). Observe that this definition of idle time intentionally does not cover periods spent passively receiving *while one or multiple streams are active* since these are already captured by the main experiment.

Our findings show that clients predominantly spend a negligible fraction of time in an idle state (figure 4.6). This suggests that the typical behavior of online Tor clients does not include lengthy periods in which no streams are active and continuous cover traffic insertion would be required according to Loopix' traffic shaping strategy. However, the models determining `tgen`'s traffic generation were learned at Tor relays [51]. The traffic initiation behavior of individual `tgen`-clients is therefore merely extrapolated from what was observed at these relays and does not necessarily accurately reflect the actual behavior of *individual* users.

[2]Note that it is evidently not possible to extrapolate the idle time of *destinations* addressed over Tor based on `tgen`'s traffic models.

[3]Of an original number of 1000 traces, 60 had to be discarded due to connectivity errors inside the `Shadow` simulation.

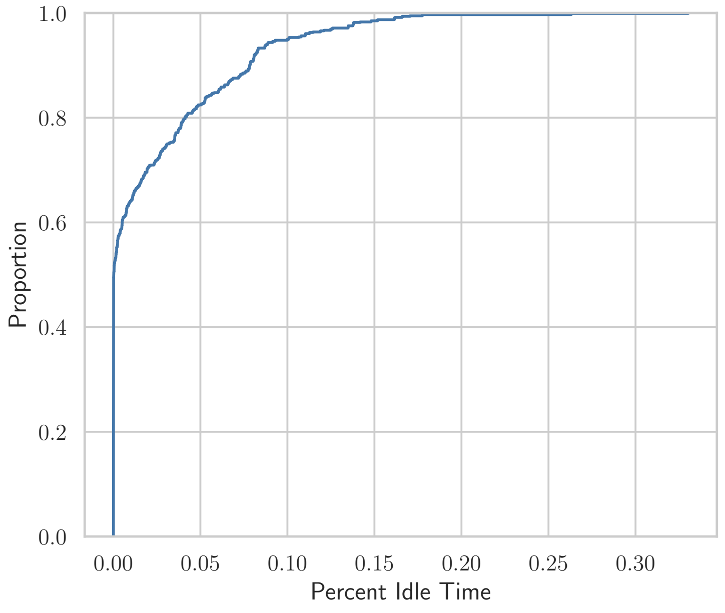
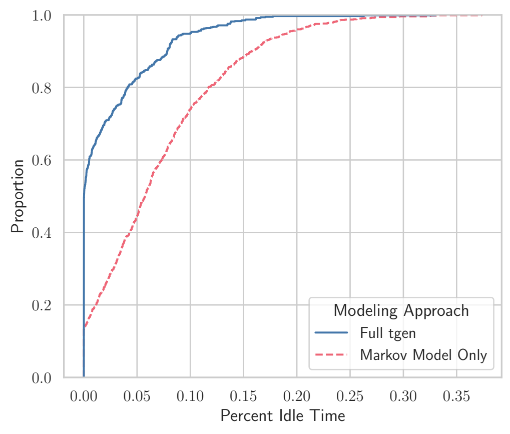


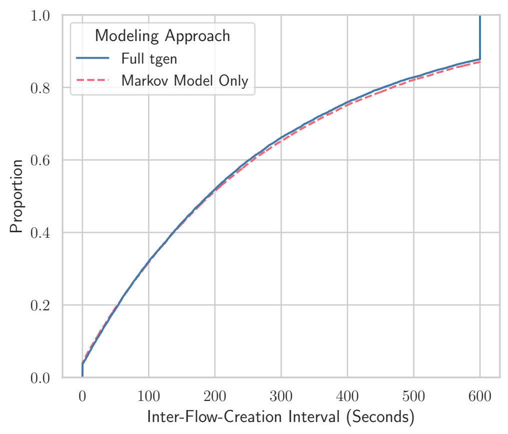
Figure 4.6: `tgen`-over-Tor: Distribution of the Fraction of Time Tor Clients Spend Idle Over a 2-Hour Period (med. < 0.001, $\sigma = 0.04$).

4.3.2 Validation of an Alternative Approach to Tor Traffic Generation in Abstract Simulation Settings

Additionally, we use this investigation to validate a `SimPy`-based implementation of `tgen`'s traffic generation behavior. `tgen` operates by generating flows according to a Markov model, each of these flows generating streams according to a second Markov model, and each of these streams generating packets according to a third Markov model. Beyond this core logic, `tgen` also handles low-level networking. Importantly, each stream is associated with a socket, and reading data from as well as writing data to a stream is governed by the associated socket becoming readable respectively writable. Since data is only written to a stream upon the corresponding socket becoming writable, the Markov model governing packet emission on that stream is only evaluated when this occurs. Our `SimPy`-based implementation strips `tgen` down to its core functionality, i.e., generating flows, streams, and packets based on the Markov



(a) Distribution of Fraction of Time Spent Idle. `tgen`: med. < 0.001, $\sigma = 0.04$; `SimPy`: med. $= 0.07$, $\sigma = 0.06$.



(b) Distribution of Inter-Flow Creation Intervals (Seconds). `tgen`: median $= 189.26$, $\sigma = 199.01$; `SimPy`: med. $= 191.53$, $\sigma = 201.653$.

Figure 4.7: Comparison of `tgen` with a `SimPy`-based Traffic Generation Implementation.

models learned at Tor exit relays. Any dependencies on the underlying operating system and network are disregarded. Our intention herein is to determine if `tgen`'s Markov models can be used *outside of* `tgen` to generate realistic Tor traffic in more abstract simulation settings such as ours. 1000 traces of clients generating flows for a period of two hours of simulated time each are collected using this implementation. We compare the behavior of `tgen` with the behavior of our `SimPy`-based implementation based on idle time as well as inter-flow-generation intervals. The results are displayed in figure 4.7. Idle time consistently falls below 1% in the case of both `tgen` and the `SimPy`-based implementation. Clients in the `tgen` simulation spend slightly less time in an idle state, which we attribute to Markov model evaluation being dependent on socket write-ability. Inter-flow creation intervals are nearly perfectly aligned. Based on these findings, it appears as though `tgen`'s Markov models can indeed be utilized to generate Tor traffic outside of `Shadow`, without requiring `tgen` itself.

Chapter 5

Discussion

In our theoretical analysis of challenges regarding the possibility of integrating Loopix with Tor, we explore heterogeneities in between the two systems' designs and how they may be resolved. In our empirical analysis, we determine the overhead incurred by shaping the kind of traffic typical of Tor according to a Poisson process - as required by the Loopix protocol - and find it to be prohibitive from the perspective of the client (when considered over the total duration of a communication flow), in particular when contrasted with APE, a defense from the Adaptive Padding family. In this section, we discuss limitations of our work and derive avenues for future work.

5.1 Scope and Focus

Our work focuses on the feasibility of enhancing Tor's privacy guarantees by approximating Loopix' behavior within the constraints of Tor's design. Yet, the integration of Loopix with Tor can just as well be viewed from an entirely different angle. Rather than altering its own protocol, the Tor network could incorporate Loopix as an optional sub-protocol. This would allow users to conveniently switch in between the two protocols depending on the desired level of privacy and the requirements of the respective application-layer protocol. Further, integration into Tor as a well-established system with a large user base might help accelerate the process of Loopix achieving a critical mass of users. It is therein not only of interest to explore how Loopix' architecture and protocol might be embedded into Tor's, but also how the privacy guarantees of the two protocols' users would interrelate.

5.2 Traffic Modeling

In each of our simulations, a client initiates a single communication flow. As stated earlier (see section 4.3), `tgen` conceptually maps flows to circuits. Tor continuously builds new circuits and assigns each new stream to the most recently created circuit, while one stream will use the same circuit for the entire duration of its lifetime. Empirical data shows that while the majority of Tor clients uses just one circuit at a time, it is not uncommon for a Tor client to use multiple circuits concurrently [51]. Multiple flows (circuits) generating data in parallel affects traffic shaping strategies differently, depending on which level they operate on (i.e., per circuit, or globally on all egress traffic).

In our APE scenario, we assume the defense to be implemented as a Pluggable Transport. A Pluggable Transport is generally used by connecting to the Tor network via a bridge (recall that a bridge is a special type of Tor node not listed publicly) that supports the Pluggable Transport. The bridge through which a client is connected to Tor then is the first hop traversed by all traffic generated by this client, and the last hop traversed by all traffic received by this client. Since Pluggable Transports are designed to protect just one link in a communication flow, they are active neither globally nor per circuit, but *per connection*, i.e., a Pluggable Transport defense operates on all traffic exchanged between a Tor client and the bridge through which it connects. In this setting, multiple flows (circuits) generating data concurrently would thus all share the same instance of the defense. Depending on how the flows' traffic patterns interact with each other, the APE defense implemented as a

Pluggable Transport might then introduce less cover traffic than in a single-flow setting: one flow's gap may co-occur with another flow's burst, requiring no introduction of cover traffic where it would have been required in a single-flow setting. With Adaptive Padding defenses implemented within Tor's circuit-padding framework, one instance of a defense is active per circuit on both endpoints. The defense then operates per flow, as in our simulations.

Loopix' traffic shaping strategy is unaware of the notion of circuits. Recall that Loopix' communication model is connectionless and message-based, i.e., each transferred piece of data is regarded as an atomic unit traversing an independently selected path through the network. It is therefore unclear at which level Loopix' traffic shaping strategy would operate if it were transferred into a connection-oriented setting. If it operated globally, we expect multiple flows coexisting simultaneously to result in lower cost in terms of bandwidth (due to fewer gaps in overall traffic) but higher cost in terms of latency (due to multiple flows concurrently producing bursts), especially on the server side. If it operated per circuit, we expect the strategy's behavior and overhead to correspond to what we observe in our simulations.

5.3 Adaptive Padding Scenario

In our evaluation, we include the APE state machine design as the representative of the Adaptive Padding family of defenses. This defense is not accompanied by a traditional scientific publication and was developed as a good-enough solution with makeshift character. We chose APE over more recent and sophisticated Adaptive Padding state machine designs such as Spring and Interspace that are implemented directly within Tor's circuit-padding framework since our simulation abstracted from details of the Tor protocol. However, any adaptations required when transferring these defenses from their original context into the highly abstract setting of our simulation may have very well been justifiable. Further, note that the WTF-Pad defense (on which APE is based) is not able to resist modern website fingerprinting attacks based on deep learning [56]. While our examination is focused on overhead only and thus does not require the most *effective* defense from the Adaptive Padding family, including a more contemporary and scientifically sound representative would have produced more interesting findings in this scenario.

APE uses randomized log-normal distributions to model the inter-emission (respectively inter-reception) intervals typical of bursts and gaps. These distributions are configured to achieve a good enough good-put for web traffic specifically, since APE is designed as a website fingerprinting defense. Our finding of a lower bandwidth overhead from the client's perspective compared to what is reported in the relevant literature might therefore be attributable to the fact that APE's distributions are ill-configured for the general Tor traffic used in our simulations.

5.4 Loopix Scenario

Loopix demands egress traffic to follow a Poisson process. This is achieved by emitting messages in intervals sampled from an exponential distribution with a specific scale parameter, which corresponds to the distribution's mean. In our analysis, we examine the overhead incurred when applying Loopix' traffic shaping strategy to Tor traffic for a range of cell rates. Each of these cell rates is therein converted to a scale parameter used for parameterization. We determine a range of suitable values based on the median cell rates of clients and servers in the absence of traffic shaping. Since the underlying traces of traffic based on which we measure these median cells rates are identical to the ones used in determining the overhead of Loopix' traffic shaping, the examined cell rates may overfit our data.

In chapter 3, we motivate the conceptual mapping of Loopix' messages onto Tor's cells. Specifically, we deem this mapping particularly clean and straightforward since it does not require the introduction of a new concept the Tor design does not already contain. Our implementation of the Loopix scenario consequently applies Loopix' traffic shaping strategy to cells. Yet, messages in Loopix are 2048 bytes in size [20, 76], almost four times as large as Tor's 514 byte cells. Examining how the overhead of Loopix' traffic shaping strategy (as applied to Tor traffic) varies as a function of message size constitutes an interesting subject of future research.

Further, our analysis does not consider any potential optimizations that may reduce the overhead incurred by Loopix' traffic shaping strategy. For example, the rate of traffic may be dynamically adjusted depending on patterns in application traffic, in order to require less cover traffic injection in idle periods and introduce smaller delays during bursts in traffic. Naturally, in the presence of such an optimization anonymity guarantees could only be computed in terms of upper and lower bounds.

Chapter 6

Related Work

Our focus in this work is to explore if and how Tor may evolve in the face of ever-more-powerful adversaries by adopting the key features of Loopix, a low-latency mix network. In this chapter, we discuss three adjacent areas of research. Firstly, we introduce efforts at optimizing Loopix' performance and deploying low-latency mix networks in the real world. Secondly, we discuss traffic analysis defenses for Tor that are qualitatively different from Adaptive Padding. Thirdly, we give an overview of endeavors at providing anonymous communications without requiring an overlay network.

6.1 Performance Optimizations for Loopix and Deployed Low-Latency Mix Networks

Hugenroth et al. [46] present *Rollercoaster*, an efficient multicast scheme for mix networks. The authors demonstrate and evaluate Rollercoaster in the context of using it with Loopix as the underlying mix network. When sending a message to members of a group over Loopix without optimizations, a lag in between the first and last member of the group receiving the message emerges due to the requirement that traffic emission must follow a Poisson process with a specific parameter. Rollercoaster builds a distribution graph for multicasts in which group members that have already received a broadcast message participate in its distribution to accelerate the delivery process.

The *Nym* project [29] aims to deploy the Loopix mix network design in a real-world setting. Access to the network is mandated by a dedicated crypto-token, which is also used to reward mix nodes for providing a high quality of service. The Nym design includes a number of modifications and extensions to the original Loopix protocol and architecture. Yet, many of Loopix' core features - such as the usage of the Sphinx packet format, stratified network topology as well as continuous-time mixing strategy - are adopted. Nym's threat model assumes an adversary that is able to observe traffic in the entire network, corrupt a subset of participants in either of the roles defined by the protocol, and mount active attacks using a limited budget of Nym's cryptocurrency. Nym is intended for use with any message-based application tolerating packet loss and packet reordering, with a specific focus on blockchain applications. The Nym project is a for-profit organization.

The *xx network* [109] is another commercial venture providing metadata-private message-based communication as a service. Its mixing strategy in principle relies on batching and re-ordering in the manner of Chaum's original mix network design, but includes a number of performance optimizations. Most notably, expensive public-key operations are carried-out in a dedicated *pre-computation* phase before the actual processing of messages takes place in real time. As the Nym design, the xx network design uses its own crypto-token for mandating access and rewarding mixes.

Finally, Hopr [8] is an incentivized mix network designed to provide network-layer privacy for decentralized *web3* applications. Its defining characteristic is its peer-to-peer architecture, meaning that all participants assume not only the roles of senders and receivers, but also of mixes.

Piotrowska [77] provides a comparative analysis of Nym, the xx network, and Hopr in terms of anonymity, scalability, and latency. The author finds that the xx network's batch-and-reorder mixing technique results in higher latencies compared to Nym and Hopr, which use a continuous-time mixing strategy. Further, while the anonymity guarantees of Nym and Hopr increase as the user base grows, the anonymity provided by the xx network stays constant as its only determinant is the size of the batch. Hopr, which combines continuous-time mixing with a peer-to-peer topology, is able to scale well but suffers from comparatively low anonymity guarantees due to low traffic concentration on individual links. Another disadvantage of Hopr's peer-to-peer topology is that large amounts of cover traffic have to be generated in order for users to be sufficiently anonymous. In summary, based on these findings the Nym design appears to be superior compared to the xx network and Hopr.

6.2 Other Defenses against De-Anonymization Attacks in Tor

The Tor network has officially integrated support for event-based, state machine-driven traffic analysis defenses such as those from the Adaptive Padding family [65, 73]. In this section, we provide a non-exhaustive overview of approaches to protecting Tor users from traffic analysis attacks that are qualitatively different from Adaptive Padding. As is the case regarding the Adaptive Padding defenses we discuss in chapter 3, these approaches and proposed defenses were developed in the context of defending against website fingerprinting attacks (see section 2.4.1). Recall that this type of attack is currently receiving particular attention since it can be performed by a low-resourced partial adversary as the one anticipated by Tor's threat model. While website fingerprinting defenses typically only protect the link between client and entry guard, they may also be effective against a more powerful adversary when extended to protecting entire communication paths. A systematic overview of website fingerprinting defenses can be found in Mathews et al. [64].

BuFlo [9], *Tamaraw* [10], and *DynaFlow* [63] represent a family of defenses of particular interest in the context of the present work since in the same manner as Loopix' traffic shaping strategy, their behavior is deterministic in the sense that the pattern they enforce on traffic is independent of the underlying traffic's shape. While Loopix deterministically molds egress traffic into a Poisson process, BuFlo, Tamaraw, and DynaFlow enforce traffic emission at a constant rate. Intelligent design choices include dynamically adapting the rate at which traffic is emitted [63], treating incoming and outgoing traffic differently [9, 10], and choosing a packet-length[1] that maximizes performance [9, 10]. Following the security principle that *structure can be exploited*, and in the light of ever more powerful machine learning- and deep learning-based website fingerprinting attacks, this family of defenses thus chooses to remove all structure. While this results in high success rates against website fingerprinting attacks, the incurred cost in terms of latency and bandwidth is comparatively high.

HyWF [43] and *TrafficSliver* [27] are two defenses designed to obfuscate traffic patterns *without* requiring the deliberate introduction of delays or cover traffic. Instead, traffic analysis attacks are thwarted by splitting traffic into multiple concurrent but separate streams following different network routes. While this approach can provide efficient protection against a partial adversary, a more powerful attacker with a global view of the network may be able to recombine the different streams and thus defeat a defense relying on traffic-splitting.

Lastly, a number of defenses are specifically geared towards confusing the machine learning and deep learning models commonly underlying website fingerprinting attacks. *Walkie-Talkie* [107], *Mockingbird* [83], and *Blind Adversarial Network Perturbations (BANP)* [71] are prominent examples.

[1]Where the term *packet* pertains to the unit of communication from the perspective of the defense.

6.3 Non-Overlay Anonymous Communication Systems

The majority of anonymous communication systems are implemented as overlay networks, meaning that they are located in between the application- and transport layer. Network-layer anonymity systems by contrast assume that the network infrastructure itself contains support for anonymous communication. Concretely, network infrastructure components such as routers are expected to actively take part in the establishment of anonymous communication channels as well as the forwarding of anonymous traffic. The inefficiencies introduced by overlay networks' additional layer of indirection can thus be avoided. Ultimately, the hope here is to explore the limits of the anonymity trilemma [14].

Examples of network-layer anonymity systems include *LAP* (Lightweight Anonymity and Privacy) [45], *Dovetail* [89], *HORNET* [13], and *TARANET* [14]. Due to the substantial challenges implied by adapting the existing internet infrastructure to natively support anonymity, network-layer anonymity systems are typically intended for use with novel internet architectures such as *SCION* [16].

The *MaybeNot* framework [81] represents another approach to reducing the layers of indirection in anonymous communications. It allows for integrating traffic analysis defenses into existing protocols such as TLS and QUIC. The framework's state machine-based design is inspired by Tor's circuit-padding framework.

Chapter 7

Conclusion

The Tor network is the largest deployed anonymous communication system to have ever existed. Twenty years after its initial launch, Tor is struggling to resist ever-more-powerful adversaries. The Loopix Anonymity System is a low-latency mix network that has been formally proven to offer a particularly advantageous trade-off between the anonymity it can provide and the overhead in terms of latency and bandwidth it requires. In this work, we explore the possibility of approximating the behavior of Loopix within the constraints of Tor's design.

We provide a comprehensive theoretical discussion of the main challenges regarding the integration of Loopix with Tor based on the relevant literature. Suspecting Loopix' traffic shaping strategy to constitute the biggest obstacle, we conduct an empirical investigation in which we explore its cost in terms of latency and bytes when applied to traffic typical of the Tor network. We contrast this overhead with the cost incurred by applying a defense from the Adaptive Padding family to the same traffic. Adaptive Padding is the traffic analysis protection strategy that inspired Tor's official traffic pattern obfuscation framework. While Adaptive Padding defenses strategically inject cover traffic depending on observed patterns in application traffic, Loopix' traffic shaping strategy deterministically shapes egress traffic according to a Poisson process without adjusting its behavior depending on application traffic. Our findings replicate the comparatively low cost of Adaptive Padding reported in the literature. Further, we find that Loopix' traffic shaping strategy results in a prohibitive overhead from the point of view of participants in the client role when viewed across an entire communication flow.[1] We attribute this to the asymmetry of the traffic typically transported over Tor, in which clients spend substantially more time passively receiving than actively sending, resulting in lengthy periods requiring continuous cover traffic emission. A consequence of this asymmetry is that the Poisson processes governing traffic emission on the client and server sides require separate parametrization for overhead to be minimized. An interesting avenue for future work consists in exploring how Loopix' traffic shaping strategy might be refined to introduce less overhead, drawing inspiration from other deterministic traffic analysis defenses developed for Tor. Further, it is worth investigating the possibility of the Tor network including a low-latency mix network design such as Loopix as an optional sub-protocol.

[1]A communication flow therein corresponds to a number of streams logically grouped together.

Bibliography

[1] *About: History*. The Tor Project. URL: https://www.torproject.org/about/history/.

[2] Ittai Abraham, Benny Pinkas, and Avishay Yanai. “Blinder–Scalable, Robust Anonymous Committed Broadcast”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 1233–1252.

[3] Nikolaos Alexopoulos et al. “{MCMix}: Anonymous Messaging via Secure Multiparty Computation”. In: *26th USENIX Security Symposium (USENIX Security 17)*. 2017, pp. 1217–1234.

[4] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. “Web MIXes: A system for anonymous and unobservable Internet access”. In: *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings*. Springer. 2001, pp. 115–129.

[5] Oliver Berthold and Heinrich Langos. “Dummy traffic against long term intersection attacks”. In: *Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers 2*. Springer. 2003, pp. 110–128.

[6] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. “The disadvantages of free MIX routes and how to overcome them”. In: *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings*. Springer. 2001, pp. 30–45.

[7] Philippe Boucher, Adam Shostack, and Ian Goldberg. “Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems”. In: *Inc., December* (2000).

[8] Sebastian Burgel and Robert Kiel. *HOPR - a Decentralized and Metadata-Private Messaging Protocol with Incentives*. hoprnet. Nov. 2019.

[9] Xiang Cai, Rishab Nithyanand, and Rob Johnson. “Cs-buflo: A congestion sensitive website fingerprinting defense”. In: *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. 2014, pp. 121–130.

[10] Xiang Cai et al. “A systematic approach to developing and evaluating website fingerprinting defenses”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014, pp. 227–238.

[11] David Chaum. “The dining cryptographers problem: Unconditional sender and recipient untraceability”. In: *Journal of cryptology* 1 (1988), pp. 65–75.

[12] David L Chaum. “Untraceable electronic mail, return addresses, and digital pseudonyms”. In: *Communications of the ACM* 24.2 (1981), pp. 84–90.

[13] Chen Chen et al. “HORNET: High-speed onion routing at the network layer”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 1441–1454.

[14] Chen Chen et al. “TARANET: Traffic-analysis resistant anonymity at the network layer”. In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2018, pp. 137–152.

[15] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. “Online website fingerprinting: Evaluating website fingerprinting attacks on Tor in the real world”. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 753–770.

[16] Laurent Chuat et al. *The Complete Guide to SCION*. Springer, 2022.

[17] Henry Corrigan-Gibbs, Dan Boneh, and David Mazieres. “Riposte: An anonymous messaging system handling millions of users”. In: *2015 IEEE Symposium on Security and Privacy*. IEEE. 2015, pp. 321–338.

[18] Henry Corrigan-Gibbs and Bryan Ford. “Dissent: accountable anonymous group messaging”. In: *Proceedings of the 17th ACM conference on Computer and communications security*. 2010, pp. 340–350.

[19] George Danezis. “Mix-networks with restricted routes”. In: *Privacy Enhancing Technologies: Third International Workshop, PET 2003, Dresden, Germany, March 26-28, 2003. Revised Papers 3*. Springer. 2003, pp. 1–17.

[20] George Danezis. *Python implementation of the Sphinx mix format*. UCL-InfoSec. URL: https://github.com/UCL-InfoSec/sphinx.

[21] George Danezis. “The traffic analysis of continuous-time mixes”. In: *International Workshop on Privacy Enhancing Technologies*. Springer. 2004, pp. 35–50.

[22] George Danezis and Richard Clayton. “Route fingerprinting in anonymous communications”. In: *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P’06)*. IEEE. 2006, pp. 69–72.

[23] George Danezis and Claudia Diaz. “A survey of anonymous communication channels”. In: (2008).

[24] George Danezis, Roger Dingledine, and Nick Mathewson. “Mixminion: Design of a type III anonymous remailer protocol”. In: *2003 Symposium on Security and Privacy, 2003*. IEEE. 2003, pp. 2–15.

[25] George Danezis and Ian Goldberg. “Sphinx: A compact and provably secure mix format”. In: *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 269–282.

[26] Debajyoti Das et al. “Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency-choose two”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 108–126.

[27] Wladimir De la Cadena et al. “Trafficsliver: Fighting website fingerprinting attacks with traffic splitting”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 1971–1985.

[28] Claudia Diaz. “Mix Networks”. In: *Encyclopedia of Cryptography, Security and Privacy*. Springer, 2022, pp. 1–5.

[29] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. *The Nym Network*. Tech. rep. Nym Technologies SA, Feb. 2021.

[30] Claudia Diaz et al. “Panel discussion—mix cascades versus peer-to-peer: is one concept superior?” In: *Privacy Enhancing Technologies: 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004. Revised Selected Papers 4*. Springer. 2005, pp. 242–242.

[31] Claudia Diaz et al. “Towards measuring anonymity”. In: *Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers*. Springer. 2003, pp. 54–68.

[32] Roger Dingledine and Nick Mathewson. “Anonymity loves company: Usability and the network effect.” In: *WEIS*. Citeseer. 2006.

[33] Roger Dingledine and Nick Mathewson. *Tor Path Specification*. The Tor Project. Dec. 2021.

[34] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. “Synchronous batching: From cascades to free routes”. In: *International Workshop on Privacy Enhancing Technologies*. Springer. 2004, pp. 186–206.

[35] Matthew Edman and Buent Yener. “On anonymity in an electronic society: A survey of anonymous communication systems”. In: *ACM Computing Surveys (CSUR)* 42.1 (2009), pp. 1–35.

[36] Tariq Elahi et al. “Changing of the guards: A framework for understanding and improving entry guard selection in Tor”. In: *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*. 2012, pp. 43–54.

[37] Nick Feamster and Roger Dingledine. “Location diversity in anonymity networks”. In: *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. 2004, pp. 66–76.

[38] Michael J Freedman and Robert Morris. “Tarzan: A peer-to-peer anonymizing network layer”. In: *Proceedings of the 9th ACM conference on Computer and communications security*. 2002, pp. 193–206.

[39] Sharad Goel et al. *Herbivore: A scalable and efficient protocol for anonymous communication*. Tech. rep. Cornell University, 2003.

[40] David M Goldschlag, Michael G Reed, and Paul F Syverson. “Hiding routing information”. In: *Information Hiding: First International Workshop Cambridge, UK, May 30–June 1, 1996 Proceedings*. Springer. 2005, pp. 137–150.

[41] Ceki Gulcu and Gene Tsudik. “Mixing E-mail with Babel”. In: *Proceedings of Internet Society Symposium on Network and Distributed Systems Security*. IEEE. 1996, pp. 2–16.

[42] Jamie Hayes and George Danezis. “k-fingerprinting: A robust scalable website fingerprinting technique”. In: *25th USENIX Security Symposium (USENIX Security 16)*. 2016, pp. 1187–1203.

[43] Sébastien Henri et al. “Protecting against website fingerprinting with multihoming”. In: *Proceedings on Privacy Enhancing Technologies* 2020.2 (2020), pp. 89–110.

[44] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. “Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naive bayes classifier”. In: *Proceedings of the 2009 ACM workshop on Cloud computing security*. 2009, pp. 31–42.

[45] Hsu-Chun Hsiao et al. “LAP: Lightweight anonymity and privacy”. In: *2012 IEEE Symposium on Security and Privacy*. IEEE. 2012, pp. 506–520.

[46] Daniel Hugenroth, Martin Kleppmann, and Alastair R Beresford. "Rollercoaster: An Efficient {Group-Multicast} Scheme for Mix Networks". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 3433–3450.

[47] Rob Jansen and Nicholas J Hopper. "Shadow: Running Tor in a box for accurate and efficient experimentation". In: (2011).

[48] Rob Jansen and Aaron Johnson. "Safely measuring tor". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 1553–1567.

[49] Rob Jansen, Jim Newsome, and Ryan Wails. "Co-opting Linux Processes for High-Performance Network Simulation". In: *Proceedings of the 2022 USENIX Annual Technical Conference (USENIX ATC 22)*. 2022, pp. 327–350.

[50] Rob Jansen and Matthew Traudt. "Tor's been KIST: A case study of transitioning Tor research to practice". In: *arXiv preprint arXiv:1709.01044* (2017).

[51] Rob Jansen, Matthew Traudt, and Nicholas Hopper. "Privacy-Preserving Dynamic Learning of Tor Network Traffic". In: *25th ACM Conference on Computer and Communications Security (CCS)*. See also https://tmodel-ccs2018.github.io. 2018.

[52] Rob Jansen et al. "KISt: Kernel-informed socket transport for ToR". In: *ACM Transactions on Privacy and Security (TOPS)* 22.1 (2018), pp. 1–37.

[53] Rob Jansen et al. "Never Been {KIST}: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport". In: *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 2014, pp. 127–142.

[54] Aaron Johnson et al. "Users get routed: Traffic correlation on Tor by realistic adversaries". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 337–348.

[55] Marc Juarez et al. *WTF-PAD*. URL: https://github.com/wtfpad/wtfpad.

[56] Marc Juarez et al. "WTF-PAD: toward an efficient website fingerprinting defense for tor". In: *CoRR, abs/1512.00524* (2015).

[57] Aniket Kate and Ian Goldberg. "Using sphinx to improve onion routing circuit construction". In: *Financial Cryptography and Data Security: 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers 14*. Springer. 2010, pp. 359–366.

[58] Dogan Kesdogan, Jan Egner, and Roland Buschkes. "Stop-and-go-mixes providing probabilistic anonymity in an open system". In: *Information Hiding: Second International Workshop, IH'98 Portland, Oregon, USA, April 14–17, 1998 Proceedings*. Springer. 1998, pp. 83–98.

[59] Albert Kwon, David Lu, and Srinivas Devadas. "XRD: Scalable messaging system with cryptographic privacy". In: *arXiv preprint arXiv:1901.04368* (2019).

[60] David Lazar, Yossi Gilad, and Nickolai Zeldovich. "Karaoke: Distributed private messaging immune to passive traffic analysis". In: *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*. 2018, pp. 711–725.

[61] Zhen Ling et al. "A new cell counter based attack against tor". In: *Proceedings of the 16th ACM conference on Computer and communications security*. 2009, pp. 578–589.

[62] Isis Lovecruft et al. *Tor Guard Specification*. The Tor Project. Oct. 2021.

[63] David Lu et al. "Dynaflow: An efficient website fingerprinting defense based on dynamically-adjusting flows". In: *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. 2018, pp. 109–113.

[64] Nate Mathews et al. "SoK: A critical evaluation of efficient website fingerprinting defenses". In: *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2023, pp. 969–986.

[65] Nick Mathewson. *New Release: Tor 0.4.0.5*. 2019. URL: https://blog.torproject.org/new-release-tor-0405.

[66] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[67] Hooman Mohajeri Moghaddam et al. “Skypemorph: Protocol obfuscation for tor bridges”. In: *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012, pp. 97–108.

[68] Ulf Moller et al. “Mixmaster protocol—version 2”. In: (2003).

[69] Steven J Murdoch and George Danezis. “Low-cost traffic analysis of Tor”. In: *2005 IEEE Symposium on Security and Privacy (S&P’05)*. IEEE. 2005, pp. 183–195.

[70] Steven J Murdoch and Piotr Zieliński. “Sampled traffic analysis by internet-exchange-level adversaries”. In: *International workshop on privacy enhancing technologies*. Springer. 2007, pp. 167–183.

[71] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. “Defeating {DNN-Based} Traffic Analysis Systems in {Real-Time} With Blind Adversarial Perturbations”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 2705–2722.

[72] Lasse Overlier and Paul Syverson. “Locating hidden servers”. In: *2006 IEEE Symposium on Security and Privacy (S&P’06)*. IEEE. 2006, 15–pp.

[73] Mike Perry and George Kadianakis. *Circuit Padding Developer Documentation*. The Tor Project. URL: https://github.com/torproject/tor/blob/main/doc/HACKING/CircuitPaddingDevelopment.md.

[74] Mike Perry and George Kadianakis. *Tor Padding Specification*. The Tor Project. July 2022.

[75] Andreas Pfitzmann and Marit Hansen. “Anonymity, unlinkability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology”. In: (2005).

[76] Ania Piotrowska and George Danezis. *The public repository of the Loopix mix system.* UCL-InfoSec. URL: https://github.com/UCL-InfoSec/loopix.

[77] Ania M Piotrowska. "Studying the anonymity trilemma with a discrete-event mix network simulator". In: *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society.* 2021, pp. 39–44.

[78] Ania M Piotrowska et al. "The Loopix anonymity system". In: *26th {USENIX} Security Symposium ({USENIX} Security 17).* 2017, pp. 1199–1216.

[79] Tobias Pulls. *Adaptive Padding Early (APE).* The HOT Research Project. Dec. 2016. URL: https://www.cs.kau.se/pulls/hot/thebasketcase-ape/.

[80] Tobias Pulls. *APE Implementation as Pluggable Transport.* Dec. 2016. URL: https://github.com/pylls/basket2/blob/master/padding_ape.go.

[81] Tobias Pulls. "Maybenot: A Framework for Traffic Analysis Defenses". In: *arXiv preprint arXiv:2304.09510* (2023).

[82] Tobias Pulls. "Towards effective and efficient padding machines for tor". In: *arXiv preprint arXiv:2011.13471* (2020).

[83] Mohammad Saidur Rahman et al. "Mockingbird: Defending against deep-learning-based website fingerprinting attacks with adversarial traces". In: *IEEE Transactions on Information Forensics and Security* 16 (2020), pp. 1594–1609.

[84] Jean-Francois Raymond. "Traffic analysis: Protocols, attacks, design issues and open problems". In: *Designing Privacy Enhancing Technologies. LNCS* 2009 (2001), pp. 10–29.

[85] Michael G Reed, Paul F Syverson, and David M Goldschlag. "Anonymous connections and onion routing". In: *IEEE Journal on Selected areas in Communications* 16.4 (1998), pp. 482–494.

[86] Michael G Reed, Paul F Syverson, and David M Goldschlag. “Proxies for anonymous routing”. In: *Proceedings 12th Annual Computer Security Applications Conference*. IEEE. 1996, pp. 95–104.

[87] Michael K Reiter and Aviel D Rubin. “Crowds: Anonymity for web transactions”. In: *ACM transactions on information and system security (TISSEC)* 1.1 (1998), pp. 66–92.

[88] Marc Rennhard and Bernhard Plattner. “Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection”. In: *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society*. 2002, pp. 91–102.

[89] Jody Sankey and Matthew Wright. “Dovetail: Stronger anonymity in next-generation internet routing”. In: *Privacy Enhancing Technologies: 14th International Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings 14*. Springer. 2014, pp. 283–303.

[90] Sajin Sasy and Ian Goldberg. “SoK: Metadata-Protecting Communication Systems”. In: *Cryptology ePrint Archive* (2023).

[91] Andrei Serjantov and George Danezis. “Towards an information theoretic metric for anonymity”. In: *Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers 2*. Springer. 2003, pp. 41–53.

[92] Andrei Serjantov, Roger Dingledine, and Paul Syverson. “From a trickle to a flood: Active attacks on several mix types”. In: *International Workshop on Information Hiding*. Springer. 2002, pp. 36–52.

[93] Claude E Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[94] Vitaly Shmatikov and Ming-Hsiu Wang. “Timing analysis in low-latency mix networks: Attacks and defenses”. In: *Computer Security–ESORICS 2006: 11th European*

Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006. Proceedings 11. Springer. 2006, pp. 18–33.

[95] Payap Sirinam et al. “Deep fingerprinting: Undermining website fingerprinting defenses with deep learning”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security.* 2018, pp. 1928–1943.

[96] Payap Sirinam et al. “Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.* 2019, pp. 1131–1148.

[97] Yixin Sun et al. “{RAPTOR}: Routing attacks on privacy in tor”. In: *24th USENIX Security Symposium (USENIX Security 15).* 2015, pp. 271–286.

[98] Paul Syverson. “A peel of onion”. In: *Proceedings of the 27th Annual Computer Security Applications Conference.* 2011, pp. 123–137.

[99] Paul Syverson. “Why I’m not an entropist”. In: *Security Protocols XVII: 17th International Workshop, Cambridge, UK, April 1-3, 2009. Revised Selected Papers 17.* Springer. 2013, pp. 213–230.

[100] Paul Syverson, Roger Dingledine, and Nick Mathewson. “Tor: The second generation onion router”. In: *Usenix Security.* 2004, pp. 303–320.

[101] Paul Syverson et al. “Towards an analysis of onion routing security”. In: *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings.* Springer. 2001, pp. 96–114.

[102] Paul F Syverson, David M Goldschlag, and Michael G Reed. “Anonymous connections and onion routing”. In: *Proceedings. 1997 IEEE symposium on security and privacy (cat. no. 97CB36097).* IEEE. 1997, pp. 44–54.

[103] Parisa Tabriz and Nikita Borisov. “Breaking the collusion detection mechanism of MorphMix”. In: *International Workshop on Privacy Enhancing Technologies.* Springer. 2006, pp. 368–383.

[104] Can Tang and Ian Goldberg. "An improved algorithm for Tor circuit scheduling". In: *Proceedings of the 17th ACM conference on Computer and communications security*. 2010, pp. 329–339.

[105] *Tor Metrics*. The Tor Project. URL: https://metrics.torproject.org/.

[106] Jelle Van Den Hooff et al. "Vuvuzela: Scalable private messaging resistant to traffic analysis". In: *Proceedings of the 25th Symposium on Operating Systems Principles*. 2015, pp. 137–152.

[107] Tao Wang and Ian Goldberg. "{Walkie-Talkie}: An efficient defense against passive website fingerprinting attacks". In: *26th USENIX Security Symposium (USENIX Security 17)*. 2017, pp. 1375–1390.

[108] Tao Wang and Ian Goldberg. "Improved website fingerprinting on tor". In: *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. 2013, pp. 201–212.

[109] *White Paper xx cMix*. xx.network. Apr. 2021.

[110] Ethan Witwer, James K Holland, and Nicholas Hopper. "Padding-only defenses add delay in Tor". In: *Proceedings of the 21st Workshop on Privacy in the Electronic Society*. 2022, pp. 29–33.

[111] Matthew Wright et al. "Defending anonymous communications against passive logging attacks". In: *2003 Symposium on Security and Privacy, 2003.* IEEE. 2003, pp. 28–41.

[112] Matthew K Wright et al. "The predecessor attack: An analysis of a threat to anonymous communications systems". In: *ACM Transactions on Information and System Security (TISSEC)* 7.4 (2004), pp. 489–522.