KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Evaluation Techniques for Mix Networks

Iness Ben Guirat

Supervisor:
Prof. dr. Claudia Diaz

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

June 2024

Evaluation Techniques for Mix Networks

Iness BEN GUIRAT

Examination committee:
Prof. dr. ir. Paul Sas, chair
Prof. dr. Claudia Diaz, supervisor
Prof. dr. ir. Bart Preneel
Prof. dr. ir. Frank Piessens
Prof. dr. Jan Tobias Mühlberg
(ULB)
Prof. dr. Stefanie Roos
(University of Kaiserslautern-Landau)
Dr. Andrei Serjantov
(BNP Paribas)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

June 2024

Preface

> "The whole is more than the sum of its parts."
>
> Aristotle

I would like to acknowledge the invaluable help and support of many individuals, without whom this thesis could not have been realized.

First and foremost, I wish to express my profound gratitude to my supervisor, Prof. Claudia Diaz, for granting me the opportunity to pursue this Ph.D. Mentoring a Ph.D. student is a challenging task, yet she has managed it excellently. Prof. Diaz has not only equipped me with specific skills but, most importantly, taught me how to think critically, a crucial skill for anyone studying privacy. For her guidance and invaluable insights, I am immensely grateful.

To Prof. Bart Preneel, from whom I have learned immensely, not just from his talks and lectures—which I was fortunate enough to attend some of them—but also through those brief yet insightful conversations in COSIC corridors. These encounters have always brightened my day and enriched my mind. Prof. Preneel has been much more than a researcher and director throughout my PhD; he has been a crucial support for both minor and major challenges. I am deeply thankful to him for agreeing to be part of my committee, for his challenging questions during my various research presentations, and for all his help. I will always be grateful.

To Prof. Mühlberg, who greatly alleviated the stress of my PhD's final phase, making it more enjoyable. Contemplating the next steps after a Ph.D. is one of the journey's most challenging aspects. Offering me a position at Université Libre de Bruxelles (ULB) before I had even finished my Ph.D. illuminated my path. I am eagerly looking forward to collaborating with him, confident that it will be both intellectually stimulating and fulfilling.

I am grateful to Dr. Andrei Serjantov, Prof. Stefanie Roos, and Prof. Frank Piessens for agreeing to be part of my committee, despite their busy schedules. Although my interactions with each of them were brief, they were very enriching. Prof. Roos was the shepherd of my first major paper for PETs, and her feedback was extremely helpful. Prof. Piessens, a member of my supervisory committee, has evaluated my progress throughout the four years and has always provided me with valuable advice. I had the pleasure of meeting Andrei during PETs 2022. I am thankful for his attendance at my talk, as well as the interesting conversations I had with him both during and after the talk. Additionally, I extend my gratitude to Prof. Paul Sas for chairing the committee.

To my incredible co-authors Hadas Zeilberger, Debajyoti Das, Devashish Gosain and Karim Eldefrawi my gratitude is boundless. I have gained invaluable knowledge from each one of them. Navigating the pressures of tight deadlines is challenging, but together, we have turned each project into a unique contribution. Without their collaboration, this thesis would not have been possible.

A special thank you goes to all the COSIC researchers, with whom I had the pleasure of discussing research and privacy, as well as sharing laughter and joy; Younes, Akash, Georgio, Rafa, Lennart, Mahdi,Jeongjeung, Sayon, Hilder, and everyone else not mentioned by name, please know your contributions and camaraderie have not been forgotten.

To my family, who, for reasons I may never fully understand, have always believed in me. To my sister, who has been my outlet for relieving stress and anger in moments where no other outlet seemed possible; her love always reaches me in so many different ways, both in the details and the big things. To my brother who is always my ray of sunshine no matter how dark a situation can be. My mom, with her stories, her food, and her unlimited capacity for always trying to understand the strange life I have chosen. And last but not least, to my dad, who worked constantly from sunrise to sunset to support us. I am profoundly fortunate to have such a family that provides unconditional support. Having them is a comfort beyond measure.

To Asuman, with whom I shared the happy, the sad, and the frustrating moments of a Ph.D. Her iconic breakfasts, extremely funny sense of humor, and her forever readiness to help anytime I needed have been a constant source of calm, strength, and joy. I am forever grateful for her existence in my life.

I would like to thank my Tunisian and international friends—Montassar, Ahmed, Wiem, Marwa, Amelia, Walter and many more—I am deeply grateful for your love. The moments of joy we share always leave a lasting imprint of happiness on my heart. A big thank you to my Brussels friends, who are my reminder that there is something beyond myself (and my Ph.D) worth striving for.

To Harry, Hanna Arendt once said, 'Courage [is] the political virtue par excellence.' and indeed, it is his most distinguished quality. Harry has always been my constant reminder that fear and surrender should have no place in any life choice, but only resistance and resilience. I am thankful for his presence during my Ph.D, and beyond.

To Olfa, my person, with whom laughter is the motto, feminism is our discussion, and her love is my refuge.

Finally, thank you to all the women who came before me, whose courage, resilience, and sacrifice dismantled so many barriers, allowing a woman like me to pursue a doctoral degree.

Abstract

The internet has become an essential part of contemporary society, catering to a variety of human and societal needs—from keeping long-distance relationships to accessing educational resources and facilitating economic activities. Perhaps even more significantly, the internet has also proved to be instrumental for social movements resisting dictatorships, such as in the Arab Spring. However, this unprecedented level of online activities has also led to a widespread mass surveillance and privacy violations, making the internet a tool against democratic values and human rights. Despite the potential of cryptography, it often fails to counter surveillance effectively by itself. Meanwhile, the exploration and deployment of Anonymous Communication Networks (ACNs), such as mixnets, to resist surveillance are still largely unexplored.

A mix network, or mixnet, is an overlay network composed of servers, called mixes, that anonymously route messages from senders to receivers by mixing the order of the messages. Despite their potential, their adoption has been slow, due to challenges such as significant latency, complex configurations, and difficulties in evaluating anonymity. This thesis investigates mixnets, aiming to tackle various open research questions. Despite the broad range of techniques and metrics proposed in the mixnet literature, the diverse array of mixnet building blocks and design choices, such as routing algorithms, network topologies, and mixing strategies, makes the evaluation of the anonymity provided by mixnets a significant challenge.

We first start by identifying the main building blocks of mixnets. We develop a simulation framework that enables the evaluation of anonymity across various mixnet configurations, building blocks, and threat models in order to analyze their impact on anonymity. Then, we explore previously unaddressed research questions, including mixnet parameterization and how various parameters influence the anonymity provided by a mixnet-based system. We identify external parameters, beyond the control of system designers, and we introduce a methodology that enables the fine-tuning of the remaining parameters. Furthermore, a significant contribution of our research is the proposal of a

seemingly straightforward yet complex-to-analyze scheme: blending different types of traffic within a single mixnet. For instance, enabling a network to handle diverse traffic from different applications, such as emails, instant messaging applications, or cryptocurrency transactions, each having a different latency requirement. This novel approach has an immediate real-world impact on mixnet-based systems by increasing anonymity and reducing delays. Finally, we introduce novel methods to evaluate mixnets against unconventional threat models, local adversaries, which have visibility over only a fraction of the network.

By meticulously unraveling the complexities inherent in mixnets and offering novel methods and techniques of anonymity evaluation, this thesis brings several novel contributions to the field of ACNs, especially aimed at achieving widespread deployment of mixnets.

Beknopte samenvatting

Het internet is een essentieel onderdeel geworden van de hedendaagse samenleving en voorziet in een verscheidenheid aan menselijke en maatschappelijke behoeften – van het onderhouden van langeafstandsrelaties tot het verkrijgen van toegang tot onderwijsbronnen en het faciliteren van economische activiteiten. Misschien nog belangrijker is dat het internet ook een instrument is gebleken voor sociale bewegingen die zich verzetten tegen dictaturen, zoals tijdens de Arabische Lente. Dit ongekende niveau van online-activiteiten heeft echter ook geleid tot wijdverbreide massale surveillance en privacyschendingen, waardoor het internet een instrument is geworden tegen democratische waarden en mensenrechten. Ondanks het potentieel van cryptografie slaagt het er vaak niet in om surveillance op zichzelf effectief tegen te gaan. Ondertussen zijn de verkenning en inzet van ACNs, zoals mixnets, om surveillance tegen te gaan, nog grotendeels onontgonnen.

Een mixnetwerk, of mixnet, is een overlay-netwerk dat bestaat uit servers, mixen genoemd, die berichten anoniem van afzenders naar ontvangers routeren door de volgorde van de berichten te mengen. Ondanks hun potentieel is de acceptatie ervan traag verlopen vanwege uitdagingen zoals aanzienlijke latentie, complexe configuraties en problemen bij het evalueren van anonimiteit. Dit proefschrift onderzoekt mixnets, met als doel verschillende open onderzoeksvragen te beantwoorden. Ondanks het brede scala aan technieken en metrieken dat in de mixnet-literatuur wordt voorgesteld, maakt de uiteenlopende reeks mixnet-bouwstenen en ontwerpkeuzes, zoals routeringsalgoritmen, netwerktopologieën en mengstrategieën, de evaluatie van de anonimiteit die door mixnets wordt geboden, tot een aanzienlijke uitdaging.

We beginnen eerst met het identificeren van de belangrijkste bouwstenen van mixnets. We ontwikkelen een simulatieraamwerk dat de evaluatie van anonimiteit over verschillende mixnet-configuraties, bouwstenen en dreigingsmodellen mogelijk maakt om hun impact op de anonimiteit te analyseren. Vervolgens onderzoeken we eerder onopgeloste onderzoeksvragen, waaronder mixnet-parametrisering en hoe verschillende parameters de anonimiteit beïnvloeden die

wordt geboden door een op mixnet gebaseerd systeem. We identificeren externe parameters, buiten de controle van systeemontwerpers, en we introduceren een methodologie die het verfijnen van de resterende parameters mogelijk maakt. Bovendien is een belangrijke bijdrage van ons onderzoek het voorstel van een ogenschijnlijk eenvoudig maar toch complex te analyseren schema: het samenvoegen van verschillende soorten verkeer binnen één enkel mixnet. Hierdoor kan een netwerk bijvoorbeeld divers verkeer van verschillende applicaties verwerken, zoals e-mails, instant messaging-applicaties of cryptocurrency-transacties, die elk een andere latentievereiste hebben. Deze nieuwe aanpak heeft een onmiddellijke reële impact op mixnet-gebaseerde systemen door de anonimiteit te vergroten en vertragingen te verminderen. Ten slotte introduceren we nieuwe methoden om mixnets te evalueren tegen onconventionele dreigingsmodellen, lokale tegenstanders, die slechts zicht hebben op een fractie van het netwerk.

Door de complexiteiten die inherent zijn aan mixnets minutieus te ontrafelen en nieuwe methoden en technieken voor anonimiteitsevaluatie aan te bieden, levert dit proefschrift verschillende nieuwe bijdragen op het gebied van ACNs, vooral gericht op het bereiken van een wijdverbreide inzet van mixnets.

List of Abbreviations

ACN Anonymous Communication Network. v–viii, 7, 40, 42, 43
APV Adversary with Partial Visibility. 7, 14, 35, 36
AS Autonomous System. 14

DP Differential Privacy. 16, 19, 22

GPA Global Passive Adversary. 5, 7, 13, 14, 22, 25, 37

ISP Internet Service Provider. 4, 5, 14

OP Onion Proxy. 5
OR Onion Router. 5

PETs Privacy Enhanced Technologies. 4, 42, 43

VPN Virtual Private Network. 4

Contents

List of Figures

List of Tables

Part I

Evaluating Mixnets

Chapter 1

Introduction

> "Revolutions in technology do not create new societies but they do change the terms in which social, political and economic relations are played out"
>
> Judy Wajcman - TechnoFeminism

The internet has become an integral part of modern society, shaping our personal interactions, information access, education, and economic dynamics. Yet, this widespread adoption comes at the price of increased privacy violations and mass surveillance. While corporations primarily engage in surveillance for financial objectives, including targeted advertising, consumer behavior analysis, and market research, state actors present surveillance practices as a necessity for identifying potential threats and often justify their systematic monitoring of communications as a matter of national security [30]. This constant monitoring, collection, and analysis of data leave individuals vulnerable to self-censorship, profiling, and manipulation as shown in [6]. The situation often worsens and becomes a threat to democracy and human rights when surveillance tools are disproportionately aimed at specific communities, such as activists, journalists, people of color, and other marginalized groups. This has led to wrongful arrests, such as in the case of Robert Williams, [1] diminished bodily autonomy for women as seen in the USA after Roe versus Wade [57], and individuals being included in various targeted lists, such as the no-fly list, with an extremely low level of accuracy as highlighted by Sullivan Gavin in her book *The Law of the List* [80].

[1] `https://www.theguardian.com/us-news/2023/apr/27/california-police-facial-recognition-software`

As a result, academia, free software, and hacker communities have focused their efforts on developing tools, protocols, and cryptographic schemes aimed at protecting individuals' privacy. Open-source libraries like OpenSSL [84], messaging applications such as Signal [66], and encryption standards such as OpenPGP [8] have emerged. These Privacy Enhanced Technologies (PETs) rely on cryptographic mechanisms to encrypt and authenticate data, ensuring that only the intended recipient can access and read it. However, over the last decades, there has been a growing concern over the importance of communication metadata. Metadata is the information that describes data, providing details about it without disclosing the content. It includes different attributes, for example a file's metadata includes the time it was created and modified, its location, and its size. In the context of communications, metadata includes who is sending a message, to whom, at what time, of which size and from where.

The Snowden revelations about the global surveillance programs run by the NSA and its Five Eyes alliance partners—the Government Communications Headquarters (GCHQ) in the United Kingdom, the Communications Security Establishment Canada (CSEC), the Australian Signals Directorate (ASD), and the Government Communications Security Bureau (GCSB) in New Zealand—showed how these entities collect millions of metadata records on a daily basis, store them, and analyze them [59]. While legislation is often vague when it comes to protecting metadata, several studies have shown that metadata can be used to re-identify users as well as determine highly sensitive traits about them [23, 56, 76].

The two main technologies that have been popularized in protecting communications' metadata are (i) Virtual Private Networks (VPNs) and (ii) Tor [39]. VPNs employ tunneling protocols to encrypt data transmitted between the user and the VPN server. This process hides the user's connection destination from their Internet Service Provider (ISP) and masks the user's address from the destination server, making online tracking and surveillance difficult. However, two significant issues remain. First, VPN providers have access to, and may retain records of, identifiable user information, including addresses and card payment details, along with logs of their online activities. Consequently, the *privacy* of the user relies on the trustworthiness of the VPN provider who has oversight over all the user's traffic. Instances where a major VPN provider such as VPN Express being acquired by a data collection company or where VPNs provided logs to government agencies highlight this risk. [2] [3] Second, traffic analysis research shows that patterns in packet sizes and timing, even when encrypted, can easily reveal and identify user activities [34].

[2] `https://www.expressvpn.com/blog/expressvpn-officially-joins-kape/`

[3] `https://www.wired.co.uk/article/ipvanish-vpn-review`

Tor [39], however, distributes trust between multiples intermediaries such that no single server is able to link the user's IP to the destination. Tor is an overlay network where each user runs a local software called an Onion Proxy (OP) to fetch directories, establish circuits across the network, and handle connections from the user to the applications. Users construct circuits preemptively, and then all traffic passes along 3 hops, commonly referred to as Onion Router (OR), in fixed-size cells. This prevents local network adversaries, such as an ISP, from learning the destination of a user's traffic and prevents the applications from identifying users' IP addresses. Provided that at least one OR is honest, correlating a sender to a receiver becomes challenging. Nevertheless, it is crucial to note that Tor is designed as a low-latency system, meaning messages are immediately forwarded to the next OR in the path upon receipt. Additionally, Tor is circuit-based, meaning that upon sending messages, each user constructs a circuit of 3 hops, and all messages traverse these three hops. Although these design strategies minimize end-to-end latency—thereby improving the system usability—it makes Tor users vulnerable against sophisticated traffic analysis. Adversaries are able to monitor the user's and the server's connection and establish communications patterns in order to re-identify users [2,48,60,78,87,88].

While Tor serves as an effective countermeasure, it often falls short against sophisticated traffic analysis techniques used by adversaries who can monitor all network traffic. Recognizing the limitations of the existing privacy tools, we need to pivot our focus towards more robust technologies, such as mix networks or *mixnets*.

Mixnets significantly enhance network-level privacy, particularly against what is known in the literature as a Global Passive Adversary (GPA), who can monitor all network links. Similar to Tor [39], the network is composed of multiple servers called mixes. Messages of fixed sizes, traverse multiple mixes before reaching the final destination. However, there are two main differences between mixnets and Tor. First, unlike Tor's circuit-based routing, mixnets use packet-based routing, meaning that each packet is routed independently. Second, messages are delayed inside each mix for a certain period of time before being forwarded to the next hop or the final destination. This strategy prevents an adversary from correlating the input with the output messages based on their timing and order. The earliest mixnet design by David Chaum [10] used threshold mixing, where each mix accumulates messages until a certain threshold is reached. Upon reaching this threshold, the mix decrypts, reorders and sends the messages to the next entity in the path of the message. Following Chaum's design, a variety of mixnet-based systems have been introduced, such as Vuvuzela [46], Mixminion [18], Loopix [69], and XRD [54]. However, despite the deployment of several mixnet-based systems, achieving widespread adoption has proven challenging. While the inherent latency within mixnets represents a critical barrier, additional challenges also contribute to the difficulty

of deploying them on a large scale. Crucial research questions, such as tuning mixnet parameters—for example, the optimal path length or the total number of mixes—have often been left out of scope in various mixnet papers. Cross-comparison between the different mixnet-based systems is challenging due to the extensive variety of design decisions in the literature, making end-to-end anonymity evaluation difficult.

The goal of this thesis is twofold: first, to empirically and analytically evaluate the anonymity provided by different mixnet configurations across various threat models, and second, to develop and propose methods that can effectively inform and guide the design decisions in a way that optimizes anonymity. This work addresses open research questions in mixnets, hoping to facilitate the development and deployment at a large scale of a mixnet-based system.

1.1 Contributions

To introduce the topic we first summarize the main building blocks of mixnet-based systems that have been proposed in the literature. This overview includes different mixing strategies, topologies, routing algorithms and dummy traffic strategies. Moreover, we introduce a methodology and provide a network simulator, MiXiM, that evaluates the anonymity provided by different mixnet configurations. This contribution was published in:

- Iness Ben Guirat, Devashish Gosain and Claudia Diaz (2021), "Mixim: Mixnet Design Decisions and Empirical Evaluation". In Proceedings of the 20th Workshop on Privacy in the Electronic Society (WPES' 21).

We expand on our evaluation methods in order to be able to parameterize mixnets. This research question has often been left out of scope in previous work. Proposed mixnet-based systems [1, 11, 46, 54, 69] have not only overlooked the detailed parameterization of the network, such as the total number of mixes, the number of layers, and the dummy traffic rates but also the rationale behind choosing these parameters and their impacts on anonymity. Our proposed methodology enables the selection of mixnet parameters in a way that maximizes the anonymity provided by the network. This approach introduces an analytical metric—the fraction of fully compromised paths— and uses empirical evaluations conducted using the network simulator MiXiM. Finally, our methodology takes into account various threat models. This contribution was published in:

- Iness Ben Guirat and Claudia Diaz (2022), "Mixnet optimization methods". In Proceedings on Privacy Enhancing Technologies (PoPETs 2022).

Additionally, we analyze and evaluate the anonymity provided by mixnets when blending two or more traffic types. This scenario is exemplified by the Nym

network [25], where different applications, such as instant messaging applications (Telegram) and cryptocurrency transactions, can use the same mixnet. In this context, Telegram users might tolerate smaller delays compared to those sending cryptocurrency transactions, leading to distinct traffic types. We first provide a theoretical analysis in order to be able to quantify the anonymity in such scenarios. Then, we proceed with empirical evaluations to study the impact of different parameters values on anonymity. These parameters include the message generation rates of different applications, the average delay per traffic type, and the mixnet's parameters, such as the number of layers and the number of mixes per layer. I equally contributed to this work with Debajyoti Das in the following publication:

- Iness Ben Guirat, Debajyoti Das and Claudia Diaz (2024), "Blending Different Latency Traffic With Beta Mixing". In Proceedings on Privacy Enhancing Technologies (PoPETs 2024).

Finally, we address a critical concern often cited by critics of mixnets regarding the default threat model: GPA. Syverson [82] has argued that the concept of a GPA is, first, unrealistic and, second, renders this threat model paradoxically both too powerful and too limited; the GPA can monitor all network links but cannot delay messages on those links. Additionally, Syverson argues that security measures should always account for the resources available to an attacker and hence the evaluation of ACN is not *"'a question of uncertainty in identifying a large set of senders and/or recipients, it is an issue of available resources and their expected effectiveness"* [82]. However, the inherent delays, the packet-based routing, and the diverse design choices in mixnets make the analytical calculation of event probability against an adversary with limited resources virtually impossible due to the large space of possibilities. We therefore introduce a traffic analysis method that allows us to empirically quantify the probability of an event under a rather weaker adversary which we call an Adversary with Partial Visibility (APV). I am the principal author of this contribution which was published in:

- Iness Ben Guirat, Karim Eldafrawi, Claudia Diaz and Hadas Zeilberger (2023), "Traffic Analysis by Adversaries with Partial Visibility". In European Symposium on Research in Computer Security (ESORICS 2023)

This thesis is structured into two main parts. Part I starts with an introduction to the topic, followed by Chapter 2, which offers the necessary background information needed for understanding the contributions of this thesis. Chapter 3 contextualizes these contributions, and Chapter 4 concludes with a synthesis of our work, highlighting potential avenues for future research. Part II includes the publications that have contributed to this thesis.

Chapter 2

Preliminaries

> "I think people know at an instinctual level that a life in which our thoughts, discourse, and interactions are subjected to constant algorithmic or human monitoring is no life at all"
>
> Phillip Rogaway

In his seminal work [10], David Chaum introduced a technique that used public key cryptography to enable both anonymous message sending and voting. This has led to a rich literature on two types of mixnets: *decryption mixnets*, primarily used for messaging, and *re-encryption mixnets*, used for voting. In decryption mixnets, messages are source routed, where the sender of a message selects the route through the network until it reaches the final destination. To prepare messages, senders encrypt messages with the public keys of the mixes in the path of the message in reverse order. Upon receiving a message, mixes use their private keys to strip a layer of encryption and discover the next hop in the route. Messages are then forwarded, following a specified delay determined by the mixing algorithm, to the next hop, which is either another intermediary mix or the recipient. In *re-encryption mixnets*, messages are re-randomized and provably shuffled by a fixed sequence of mixes called cascade before being flushed [7, 40, 53, 61]. Such mixnets are ideal for voting applications since, unlike messaging applications, they have limited and predictable traffic volume, very high latency tolerance, and strict public verifiability requirements. However, given their limited range of applications and well-understood anonymity tradeoffs, we consider re-encryption mixnets as

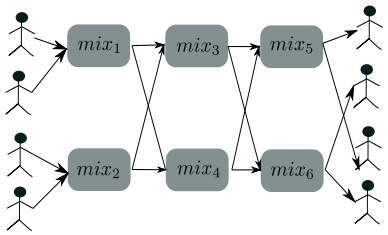


Figure 2.1: A Simplified Mixnet.

out of scope in this thesis. We instead focus on decryption mixnets that are mainly designed for message-based communications [25]. In decryption mixnets, there exist many design decisions and parameters such as the mixing algorithms, routing policies, and dummy traffic strategies, where each of these components and its parameters can have a significant influence on the anonymity and latency tradeoffs provided. Therefore, in order to be able to evaluate the anonymity they provide as well as the best way to choose these parameters, we first need to understand, the main building blocks of mixnets.

2.1 Mixnet's Building Blocks[1]

Mixes are servers that cryptographically transform and reorder messages, such that input messages are unlinkable to outputs both in terms of message appearance and timing. The transformation of message appearance is based on a cryptographic packet format, such as the Sphinx packet [19], while message re-ordering is achieved with different *mixing* algorithms [27]. We highlight the main mixing strategies:

- *Threshold Mix:* A threshold mix [10] buffers messages in an internal memory until a number T (threshold parameter) is reached. The mix, then, decrypts and permutes the T messages and forwards them in a random order.
- *Timed Mix:* A timed mix buffers messages for a period of time t. When the timeout is expired, the mix decrypts and permutes the messages it has collected and flushes them in a random order.
- *Pool Mix:* A pool mix [75] can flush either when hitting a threshold number of messages or upon expiration of a timeout. Pool mixes however do not flush all the messages they have collected. Instead, a subset of messages is flushed in a random order while the rest is kept in an internal memory called *pool* until the next flushing event. The pool algorithm

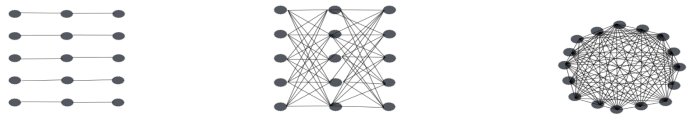
[1]This section reuses some of the texts from our two publications [4, 5].

defines the number of messages that are sent and kept. The choice of *which* messages to send and which to keep is made uniformly at random.

- *Stop-and-Go Mix:* Contrary to the previous types of mixes which are often referred to as *batch mixing*, a Stop-and-Go mix [50] does not send batches of messages at discrete flushing events, but rather processes messages individually and continuously in time. Each message received by the mix is decrypted and kept for a random amount of time and sent out when its timeout has elapsed. Mixnet-based systems that are based on this mixing strategy typically use the exponential distribution for generating these message delays [25, 69]. As shown in [17], the memory-less property of exponential distributions makes this mixing strategy not only simple and straightforward but also capable of achieving an anonymity level similar to other mixing strategies such as pool mixes.

Mixnet topology refers to the interconnection of mixes within the network. The topology outlines the possible routes (sequences of mixes) that messages can traverse. We distinguish the following types of mixnet topologies, each defined by its unique arrangement and the strategic placement of mixes:

- *Cascade:* In this topology, shown in Figure 2.2a all messages pass through a fixed sequence of mixes in a predetermined order, creating a chain where each mix forwards all its messages to the next mix [10]. The scalability of a single cascade is limited by the throughput of its least capable mix. Thus, if more capacity is required, multiple cascades can run in parallel [11, 38].
- *Stratified:* In this topology, also known as *layered* topology, mixes are organized into a fixed number of layers as shown in Figure 2.2b. Each mix in layer i receives messages from mixes in layer $i-1$ and forwards messages to mixes in layer $i+1$. Each mix is assigned to one layer at any given time. Mixes in the first layer receive messages directly from senders, while those in the last layer send messages to the end recipients. Stratified topologies are shown to provide optimal anonymity as well as being efficient to analyze [29]. Variations of stratified topology exist: Fully connected topologies mean that the mixes in layer i can receive messages from all mixes in layer $i-1$ and can send messages to all mixes in layer $i+1$, whereas not fully connected may only receive (respectively forward) messages from/to a subset of mixes in layer $i-1$ (respectively $i+1$). Furthermore, the number of mixes across layers can be the same (a *balanced* network) or can vary (*imbalanced network*), impacting both network throughput and anonymity. For example, having a single mix in one layer not only affects the throughput of the network but also increases the risk of compromise, affecting the overall likelihood of the compromise of a path. Detailed analysis and evaluations on these considerations are discussed in Part II, Publication 2.



(a) Multiple cascades (b) Stratified (c) Free routes

Figure 2.2: Mixnets' Topologies.

- *Free routes:* Free route networks are composed of mixes that are fully connected, *i.e.* they have a network graph in which mixes are connected to all other mixes (Figure 2.2c). Free routes allow maximum freedom in terms of choosing messages' paths and unlike in cascades or stratified networks that implicitly fix the number of hops. With free routing, message path lengths may be variable [33], though they may also be fixed by the routing algorithm. While free routes offer good anonymity and scalability properties, they are extremely complex to analyze [83] and are also known to perform worse than stratified networks in terms of anonymity [26], without any discernible advantage.

Dummy traffic strategies: Dummy or *cover traffic* comprises fake messages sent through the mixnet, devoid of any real data payload and intended to be discarded by the ultimate recipient. Thus, they not only increase the network's traffic volume, enhancing the system's anonymity, but also serve as an essential defense against certain active attacks, e.g., the *n-1* attack [75]. The parameters defining a dummy traffic strategy include:

- *Who* generates the dummies and *when*. Dummy messages can be produced by end-users, mixes, or both. The generation algorithm should specify the timing and frequency of dummy creation, such as at fixed intervals, during flushing events, when traffic load is minimal, etc.
- Dummy message *routing* and *destination*: The dummy traffic algorithm must delineate the routing process, including the selection of intermediate hops and the final destination. For instance, dummy messages might be dropped at the next hop, routed in a loop back to their origin, or directed to a user's mailbox.

Figure 2.3 shows three examples of dummy strategies. Client-based dummies are dummies that are generated and sent and received by clients. This strategy is used in systems like Vuvuzela and Loopix [46, 69] in order to achieve the unobservability property [67], meaning that it is not possible to tell whether a user is idle or actively communicating. Mix-based dummies are generated in the mixes and can either be sent to multiple mixes in the network or can be dropped at the next hop. When dummy messages traverse multiple mixes, the amount of traffic in the network increases after each hop resulting in an increased level

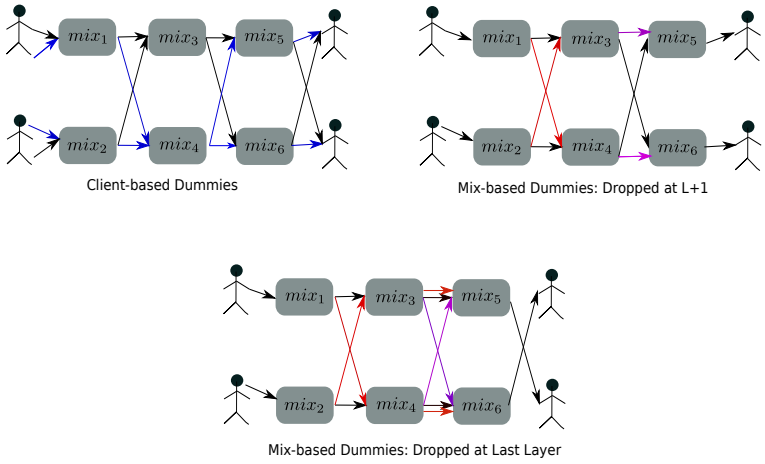


Figure 2.3: Different Dummy Strategies.

of anonymity. However, the approach of dropping dummy messages at the next hop is more cost-effective because the network performs significantly fewer public key cryptography operations. In multi-hop paths, whether generated by clients or mixes, dummies are indistinguishable from actual messages at intermediate hops and towards the GPA; only the source and destination know that it is a dummy message. We discuss the impact of the different generation rates and the types of dummy traffic types on anonymity in Part II, Publication 2.

2.2 Threat Model

Mixnets are designed to protect against a GPA, a powerful network adversary with visibility over the whole network. The primary goal of the adversary is to identify the sender or the receiver of a specific message. This adversary's capability is however limited to the monitoring of messages as they traverse the network. The adversary lacks the capability to *see* inside the mixes, meaning it cannot trivially know the mapping between inputs and outputs. Furthermore, the GPA abstains from any active attacks, including the dropping, modifying or injecting messages. Typically, in mixnet literature, it is assumed that message format throughout the transmission remains unaltered, meaning that attacks where the adversary tags messages in order to follow them are prevented. This is achieved through cryptographic schemes, using packet formats such as Sphinx [20], which are designed to provide various security features such as

bitwise unlinkability, meaning that an adversary who is observing a mix cannot link the incoming messages to the outgoing messages based on the format. In addition, Sphinx hides the route's length and the message's position within the path. Finally, it is assumed that the adversary cannot break secure encryption, ensuring the confidentiality of the data transmitted through the mixnet.

Throughout this work, we also consider adversaries that have the capability to compromise a subset of mixes. These mixes provide no anonymity to the messages routed through them. Contrarily to honest mixes, where the adversary can only probabilistically correlate between the inputs and outputs based on its observation of message arrivals and departures [17, 49], a compromising adversary has knowledge of the mapping between the inputs and outputs of these malicious mixes. In volunteer-based networks like the Nym network [25], where participants offer their servers to function as mixes, there is an inherent risk of malicious behavior. It is crucial to highlight that the anonymity of messages passing through the network relies on the presence of at least one honest mix in its path. Assuming the existence of a number of malicious mixes within the network can, depending on the network's topology, result in certain message paths being entirely compromised by the adversary. The topology of the mixnet as well as other parameters such as the number of layers and the number of mixes per layer play a critical role in determining the likelihood of a message traversing a fully compromised path. This is further explained in Part II, Publication 2.

Lastly, we additionally consider adversaries that can only monitor or compromise a portion of the network, which we call an APV. Such adversaries could include an ISP, Autonomous System (AS), or state actors that collaborate without full network observation. Despite their significant capabilities, these entities may have blind spots unlike the GPA. Given the complexity of mixnets, considering such an adversary is a challenging task. We detail our approach in modeling the APV, with its prior knowledge, capabilities, and goals in Part II, Publication 4.

2.3 Metrics and Evaluation in mixnets

Metrics are quantifiable measures of privacy that enable us to evaluate and compare various systems. We outline in this section some of the common metrics used in mixnet-based systems.

2.3.1 Entropy

Entropy provides an *average measure* of the number of candidate messages that the adversary confuses with a target message [29, 74]. For instance, an entropy

of 8 bits suggests a message's anonymity is equivalent to its indistinguishability among $2^8 = 256$ messages, and 9 bits to $2^9 = 512$ messages. This logarithmic scale implies that each increase of one bit in entropy *doubles* the set size for perfect indistinguishability, whereas a decrease *halves* it. The entropy metric accounts for the probabilistic information obtained by network adversaries in addition to corrupt adversarial nodes. To calculate this metric, the adversary chooses a target message, m_{target}, and monitors all the network links. Upon a message m_i exiting a mix, it is assigned a probability $Pr[m_i = m_{target}]$ of being the target message. H is then computed as follows:

$$H = -\sum_{n=1}^{N} Pr[m_i = m_{target}] \cdot log_2(Pr[m_i = m_{target}]) \tag{2.1}$$

Note that, the computation of the entropy metric, a crucial measure in evaluating the anonymity provided by mixnets, relies on determining the probability distribution that associates a given target message with all potential output messages, or vice versa. This distribution's calculation depends on the specific mixing strategy employed by the mix, as detailed in [24]. For instance, for a threshold mix with a parameter T, the likelihood of any output message being a particular target input message is $\frac{1}{T}$ since all collected and flushed messages by the mix are equally likely to be a specific target message. For a Poisson mixing strategy, we make use of the memory-less property of the exponential distribution. Thus, for the first message exiting the mix after a target input arrives, the probability is $\frac{1}{N}$, where N represents the total number of messages inside the mix at the time of the message leaving. The calculation for the probability of the next messages exiting the mix needs additional steps to account for the residual probability of the target message still being inside the mix. However, we lose the memory-less property when the mix delays messages with samples drawn from exponential distributions with different parameters. Such a scenario can happen in networks like such as the Nym network [25] that accommodates diverse traffic types, such as instant messaging applications and cryptocurrency transactions, each requiring different latency considerations. Consequently, messages may be delayed considering the latency tolerance of their applications, leading to a mix routing messages with delays drawn from multiple distributions. The challenge then becomes computing the probability of a message being the target without the simplification afforded by the memory-less property. We outline this analysis in Part II, Publication 3.

Given the complexity of mixnets when considering multiple mixes in the network arranged in a specified topology, obtaining the relevant distributions cannot be done in a straightforward analytical form. Prior work [26, 28, 69, 77], has therefore resorted to empirical evaluation by developing their own network

simulators such as in Loopix [2] and Vuvuzela [3]. However, these simulators typically include unique design choices of their respective systems, for example verifiable shuffle or mailboxes. While they do address specific challenges, these design choices can be peripheral to core mixnet concerns especially with regard to anonymity. We therefore need to develop a generic mixnet simulator that helps us evaluate the above building blocks and their intersections together. We describe this work in Part II, Publication 1.

2.3.2 Differential Privacy (DP)

Differential Privacy (DP) [35] is a mathematical definition of privacy commonly used in analyzing datasets. It guarantees that by adding noise using distributions such as Laplace or Gaussian, the output of a query does not allow distinguishing whether or not a specific record is part of the dataset [35]. Dwork emphasizes that bad disclosures can still occur, however the user is guaranteed that the presence of their individual data is not leaked [35]. Differential Privacy has been adapted and used in some mixnet-based systems such as Vuvuzela [46] to achieve precise privacy guarantees and bound the privacy leakage. This guarantees that even if only two users are communicating, by adding noise, the adversary is not able to determine whether or not these two users are communicating with each other. However, the privacy budget is a well-known problem in the differential privacy literature. As the authors of XRD argue in [54], this is especially a problem when applying this technique to message-based systems. The budget could run out quickly if a user sends messages frequently, and once their privacy budget is exhausted, it is unclear how to proceed.

2.3.3 Fraction of Fully Compromised Paths

A network consisting of a single mix would offer optimal anonymity if it remains honest, as this would maximize the number of messages mixed together and hence the anonymity set. Additionally, unlike a cascade topology, this reduces the propagation delays and the per-mix delays, resulting in a smaller end-to-end latency. However, trusting a single mix is not ideal. Therefore, having multiple mixes in the network is crucial so that even if some are compromised, the anonymity of the message is preserved as long as there is at least one honest mix in the path. Computing the fraction of compromised paths is a critical metric for evaluating a mixnet. This metric indicates the fraction of messages going through an entirely compromised route. To compute this metric, we need to take into consideration the topology, the number of mixes in the network,

[2] https://github.com/UCL-InfoSec/loopix
[3] https://github.com/vuvuzela/vuvuzela/tree/master/mixnet

and the number of compromised mixes. A detailed analysis is presented in Part II, Publication 2.

2.3.4 Expected Difference in Likelihood

The authors of Loopix [69] introduce the Expected Difference in Likelihood metric which aims to quantify the knowledge of the adversary in learning whether a received message is sent by a sender compared to another's. Given the probabilities p_1 (respectively p_2) of a message being sent by S_1 (respectively S_2), this metric is given by:

$$\epsilon = \left| log_2(\frac{p_1}{p_2}) \right| \tag{2.2}$$

Similar to the entropy metric, this metric also needs to be computed empirically using a network simulator: Users generate messages and send them to their corresponding recipients. Messages leaving the different mixes have three probabilities of being sent by S_1, S_2 or others. At the end of the simulation, using these probabilities the expected likelihood metric is computed using Equation (2.2).

2.3.5 Traffic Analysis in Mixnets[4]

In order to calculate the probability of a certain event in mixnets, for example linking a sender to a receiver, previous work has used several traffic analysis techniques that primarily examine patterns of messages [16, 51, 52, 72]. In [21], Danezis and Troncoso argue that at the heart of traffic analysis lies an inference problem. Applying Bayesian techniques provides a framework on which to build attacks and algorithms to estimate different probabilities of events of interest. Their significant contribution lies in introducing Bayesian inference to traffic analysis. They introduce an inference engine that samples from the distribution of network states consistent with a given set of observations to estimate the probability of an event of interest, such as the probability of a specific sender communicating with a specific receiver. Given that the core defense of mixnets to hide metadata lies in the mixing strategies that disable an adversary from mapping the inputs of mixes with their outputs, these studies have focused on the edges of mixnets where messages are sent and received. In our work, we have extended the default mixnet threat model to adversaries with partial visibility who can compromise and/or monitor only a portion of the network. By modeling the portions of the network that the adversary cannot see by a set of

[4]This sections reuses some texts from our publication 4.

Hidden States, we introduce a novel mixnet model capable of capturing various mixnets with distinct design choices and adversaries with different capabilities. This methodology is further detailed in Part II, Publication 4.

2.4 Mixnet-based systems

In this section, we introduce various mixnet-based systems that have been proposed in the literature, focusing on the main design decisions they each employ.

2.4.1 Mixmaster and Mixminion

One of the first mixnet-based systems to achieve widespread deployment was Mixmaster [58]. This system uses a Timed Dynamic Pool Mix strategy [75], in which received messages are stored in a pool. Periodically, each mix (referred to as a *remailer* in Mixmaster) randomly selects and forwards a percentage of these messages to the next mix in the network or to their final destination. Each user selects a path through up to 20 different mixes, and messages are of equal sizes. In version 3.0, Mixmaster uses dummy traffic. Mixes generate dummy messages whenever a new message enters the pool, and the number of dummy messages is drawn from a geometric distribution. Additionally, Mixmaster's documentation suggests that senders should also send dummy traffic. To avoid statistical traffic analysis by observing the timing of sending and receiving messages (and hence correlating senders to receivers), users should send messages at regular intervals and include dummy messages whenever appropriate.

Mixminion [18] builds upon Mixmaster and addresses some of its limitations. Notably, it introduces support for anonymous reply messages, with the same processing as forward messages, thus making them indistinguishable from one another and part of the same anonymity set. Reply blocks in Mixminion are of single use and enable recipients to respond without knowledge of the sender's identity. When sending an anonymous message, senders can include a reply block, outlining a pre-constructed path but in reverse. Upon receipt of a message with a reply block, the recipient, can respond using this block as the reply's route back to the sender. Mixminion accommodates three anonymity scenarios: sender-only anonymity, recipient-only anonymity, or both.

2.4.2 cMix

cMix [11] is based on batch mixes [75]. Mixes are arranged in a fixed cascade. cMix introduces dummy messages to prevent indefinite delays due to the

threshold mixes' firing condition, especially when there are few active users. The authors argue that a significant drawback of mixnets is the delay introduced by real-time public key cryptography, required each time a mix receives a message. cMix addresses this by introducing a pre-computation phase, thereby avoiding the need for users to perform public key cryptography operations and allowing lightweight mixnet applications to be deployed on small Android devices. Public key cryptography is replaced by a pre-computation phase and symmetric key cryptography in the real-time phase. Users only need to establish a shared secret with each mix during the enrollment phase, a process performed infrequently. While this strategy enhances cost efficiency and reduces latency, the per-mix delays in mixnets remain a fundamental challenge. These delays arise from the waiting time for mixes to reach the firing condition, which depends on the network traffic load. Details on the precomputation phase are omitted as they are orthogonal to our research questions.

2.4.3 Vuvuzela

Vuvuzela [46] is a mixnet-based system that prevents an adversary from determining which pairs of users are communicating. Vuvuzela's topology is a cascade with a timed mixing algorithm. Messages, whether real or fake, are encrypted sequentially with the public key of each mix in the cascade. Vuvuzela's threat model considers an adversary capable of controlling all servers except one, with the ability to monitor, delay, or inject traffic. It uses two main protocols: First, the *dialing protocol* enables users to send invitations to other users for communication. An invitation includes the sender's public key and is deposited at the recipient's *dead drop* (a virtual location on a server). During the dialing phase, users check their dead drops by downloading and decrypting the messages to find any invitations. Upon accepting an invitation, the user uses the public key they found on the dead drop for communication. The *conversation protocol* follows, where both users, having agreed to communicate, establish a shared secret and select a new dead drop for message exchange during the session. Each mix decrypts incoming messages and forwards them in a randomized order. The final mix decrypts all messages, matches the access to each dead drop, and then sends messages back through the chain in reverse order, ensuring the first mix delivers them to the intended recipients. Additionally, each server introduces cover traffic, leveraging DP to provide precise privacy guarantees. This cover traffic accesses random dead drops at the last mix, obscuring the distinction between active messaging and one-way communication (e.g., Alice sending messages without receiving any from Bob, possibly due to his offline status).

The two main drawbacks of Vuvuzela are: first, Vuvuzela assumes that users must be online all the time in order to be able to communicate, which is unrealistic. Second, it allows users to send only one message per round.

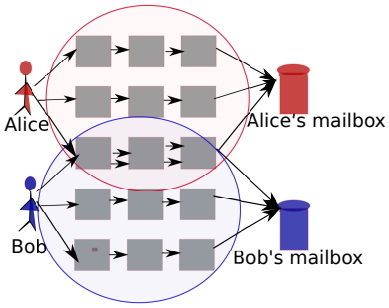
2.4.4 XRD

XRD, short for Crossroads, is a round-based protocol that consists of parallel cascades. XRD uses an algorithm that ensures that every user pair shares at least one common cascade. When a sender wants to send a message to a recipient, they encrypt the message using the public keys of the mixes in the shared cascade and the recipient's key in reverse order, routing it through this cascade. Additionally, the sender sends dummy messages to all other cascades assigned to them, directing these to its own mailbox. Should a user choose not to engage in communication, they send dummy messages through all cascades. Similar to Vuvuzela, XRD works in synchronous rounds where each user submits only one message per round. Upon receiving all the messages from the users, each chain shuffles, decrypts, and then routes the messages to the correct mailboxes. Recipients then collect their messages from their own mailboxes. The number of messages each user receives in a round is equal to the number of cascades assigned to them, hiding whether or not a user is actively communicating, as shown in Figure 2.4a.

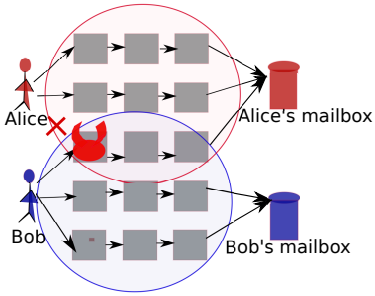
The privacy guarantees of XRD, mainly due to how each user selects the cascades, hold if there's at least one honest server at each cascade. During each round XRD mixes provably shuffle the messages in order to prevent malicious mixes from dropping messages which would not only impact the usability of the system but also the anonymity provided by the system; while provable shuffle is cryptographically expensive, it prevents the adversary from learning if a user is in conversation with another user or not. In Figure 2.4, we depict three scenarios where Alice and Bob each belong to three cascades, sharing one. A GPA cannot know whether Alice and Bob are conversing in Figure 2.4a. However, if an adversary compromises the input mix of the shared cascade, drops Alice's message, and notes from the output mix that Alice received a message (Figure 2.4b), it indicates Bob is conversing with Alice. Conversely, if no message is forwarded to Alice's mailbox, the adversary deduces that Alice is not in conversation with anyone (Figure 2.4c).

2.4.5 Karaoke

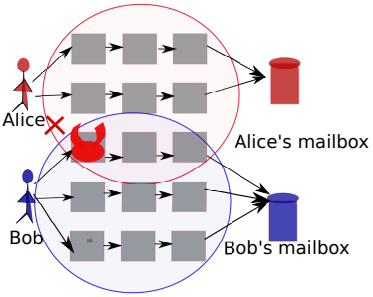
Similar to Vuvuzela and XRD, Karaoke is a synchronous round-based protocol. Karaoke's topology is a layered topology that provides horizontal scalability. There is a coordinator server that is not trusted for privacy but rather its role is



(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

Figure 2.4: XRD: Three scenarios illustrated: In the first scenario, the adversary is a GPA and he cannot distinguish whether Alice is communicating with Bob. In the second and third scenarios, the adversary additionally compromises the first mix in the common chain of Alice and Bob and drops Alice's message. In the second scenario, the adversary knows that Alice is communicating with Bob while in the third scenario, the adversary knows that Alice is not communicating with Bob.

to announce the start of a new round. At each round's start, users send a real or fake message. Mixes collect these messages and forward them to the next mix in the path which is chosen randomly by the senders. Echoing Vuvuzela, Karaoke uses a dialing protocol for users to establish a shared secret, used to agree on dead drops located in the last mix for message exchange. However, each pair of users communicating in Karaoke agrees on two dead drops where they send duplicate messages. If a user is not actively communicating, she sends two dummy messages which meet at a single dead drop. This strategy of sending two messages, whether real or fake, during each round guards against an adversary monitoring dead drop accesses and distinguishing scenarios where two dead drops are accessed by only one message each (bottom left in Figure 2.5)

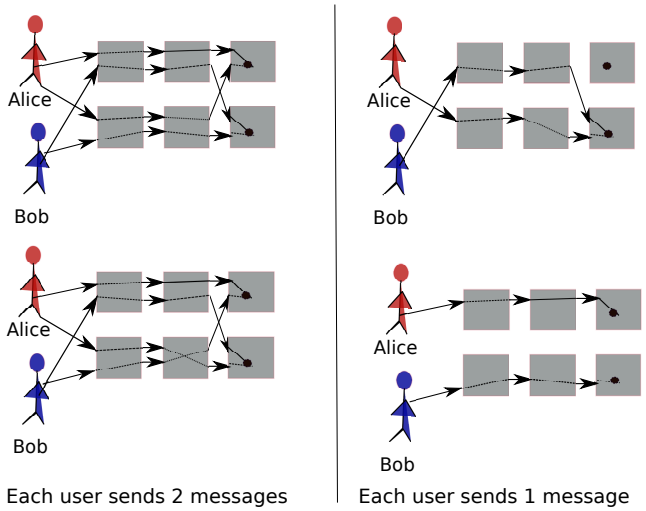


Figure 2.5: Karaoke: Two scenarios depicted: On the left, each user sends two messages per round and on the right each user sends only one message.

from those accessed by two messages, indicating communication between users (top right in Figure 2.5). As shown on the left of Figure 2.5, when users send two messages each, a GPA cannot differentiate between cases where Alice is communicating with Bob or when both users are sending two dummy messages to a random dead drop. However, as shown in the [22], by observing the number of messages in each link the adversary can still achieve a significant leakage and thus able to distinguish between the scenarios shown in Figure 2.5.

Karaoke also uses dummy traffic not only to increase the overall number of messages in the system but also to mask dead drop access patterns in the case of message dropping by an active adversary, meaning that legitimate user messages did not form a pair of accesses to the same dead drop. Furthermore, each mix generates and routes dummy messages through random paths to random dead drops. Dummy traffic messages are generated via a Poisson distribution and the amount of dummy traffic is determined by the desired DP parameters.

2.4.6 Loopix

The topology of Loopix [69] is a layered topology where each mix is assigned a position at each layer and can only receive messages from the mixes in the previous layers and send messages to mixes in the next layer. Messages are source-routed, with the sender uniformly choosing a mix from each layer and

wrapping the messages in layers of encryption, starting with the public key of the final destination. Loopix is a continuous time mixnet, meaning that the per-mix delays are drawn from an exponential distribution, allowing for tunable latency. Unlike similar mixnet-based systems, messages in Loopix do not need to wait for a threshold number to be collected by the mixes before delivery, a strategy that enhances usability. Loopix introduces service providers as offline message storage for users. There are three types of dummy traffic: *drop cover traffic*, similar to real messages but discarded by the service provider; *loop cover traffic*, sent by clients to themselves through the mixnet to detect $N-1$ attacks, where the adversary drops all messages but one message. This attack can be detected by the client. The last dummy traffic introduced in Loopix is the *loops* generated by the mixes, and their final destination is the same mix. Senders store messages in a buffer and periodically check, following a Poisson process, if there is a scheduled message to send. If so, the message is sent; otherwise, a drop cover message is generated and sent. Similarly, when recipients retrieve messages, providers send messages to the clients following a Poisson process, introducing dummy traffic if no real message is available.

These strategies of dummy traffic, along with the Poisson process-based sending and receiving by users, prevent adversaries from linking a sender to a recipient. Loopix's anonymity is evaluated using the entropy and the likelihood difference metrics across various delay rates, layers, and traffic volumes, showing the system's ability to balance latency, anonymity, and bandwidth by adjusting latency and dummy traffic generation rates.

Chapter 3

Contributions

> "We will be victorious if we have not forgotten how to learn."
>
> Rosa Luxembourg

In this chapter we contextualize and summarize the contributions of the work compiled in this thesis.

3.1 MiXiM

Our first objective in this thesis is to comprehensively explore the various building blocks of mixnets, as well as their parameters in order to analyze and evaluate their impact on anonymity. To achieve this objective, we primarily rely on the entropy metric as it offers a measure of the average anonymity provided by the mixnet. For our analysis, we have implemented MiXiM, a mixnet simulator that incorporates the core building blocks of mixnets. This simulator enables us to construct mixnets with different combinations of building blocks and calculate the entropy metric accordingly. While there are existing network simulators available [46, 54, 68], they are often tailored for evaluating specific systems, which complicates comparisons between systems and makes systematic evaluation of anonymity challenging.

Figure 3.1, shows an overview of the mixnet simulator MiXiM. We start by defining the network's building blocks and parameters. This includes setting the number of users and message sending rates, selecting the number of mixes, the mixing strategy and its parameters (*e.g.*threshold for threshold mixing, timeout

for time mixing), and deciding on the mixnet topology and associated parameters, such as the number of parallel cascades or number of layers. Additionally, we specify the simulation duration and any optional features such as dummy traffic types and parameters. Upon configuration, the simulator launches the network, with clients sending messages to recipients over a specified period. The simulation finishes by providing the distribution of entropy results, indicating the mixnet's average anonymity.

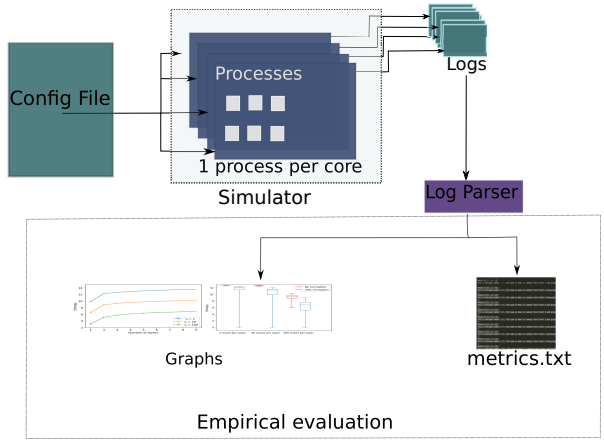


Figure 3.1: MiXiM architecture.

To compute the entropy metric under our default GPA threat model, we track all message information accessible to a GPA as messages traverse the network, logging input and output times at honest mixes, while compromised mixes function merely as links. Upon reaching a stable state, the simulator marks a *target message* and tracks it and any message that was mixed with it in the honest mixes, updating their probability of being the target message. At the end of the simulation, each message has a probability of being the target message. We use these probabilities to compute the entropy metric. To mitigate corner cases, such as messages traversing fully compromised paths, we choose N target messages, and we average the resulting entropy values.

Using this methodology, we are able to analyze various mixing strategies under similar traffic conditions, showing that Poisson mixing provides the highest degree of anonymity. Additionally, we evaluated the anonymity levels of different mixnet topologies—including Fully connected stratified, Restricted proposed in [15], and the topology proposed by the XRD mixnet-based system [54]. It is important to note, however, that our focus on XRD was specifically on its proposed topology rather than the full characteristics and features.

This comparative analysis was made possible by our custom-built simulator, designed to accommodate the different parameters and building blocks of mixnets. Furthermore, we explore how the number of layers in a mixnet impacts anonymity when having a compromising adversary. We show that increasing the number of layers beyond three does not significantly enhance anonymity as measured by entropy when there are no compromised mixes. However, in scenarios where a substantial portion of mixes (up to 40%) are compromised, adding more layers within a stratified topology can effectively reduce the likelihood of messages traversing through compromised routes. Additionally, our evaluation of mix-based dummy strategies, with different generation rates, shows that dummy traffic can indeed improve anonymity in low-traffic conditions, however, in high-traffic scenarios, the introduction of dummy messages does not provide substantial additional anonymity benefits. Further details on these contributions are discussed in Part II, Publication 1.

3.2 Parameterization of Mixnets

Building on our previous findings, we focus on mixnets that use a Poisson mixing strategy and a stratified topology. One of the main challenges in mixnet design is the optimal selection of mixnet parameters in a way that optimizes anonymity. We first need to differentiate between adjustable parameters and those that are imposed by external constraints. We start by identifying external constraints: First, the traffic load is defined by the number of users and their message generation rates. Other systems such as Karaoke and XRD [54, 55] increase the amount of traffic by forcing users to send dummy messages at each round if they are not sending real messages. This method is not ideal due to potential user offline status and bandwidth costs. User tolerance for delay influences system use, for example, users may stop using a messaging app if the end-to-end latency is too large and hence, end-to-end latency, dictated by user preferences, is an external factor. Similarly, the number of compromised mixes is considered beyond the designer's control, especially in volunteer-based networks such as Nym [25], where individuals contribute to the network with their own servers raising the potential malicious behavior. On the other hand, the parameters that can be chosen include the number of mixes, the number of layers, the width of the network (the number of mixes per layer), and the rate of dummy traffic generation.

3.2.1 Mixnet's external factors

Given a traffic load, a rate of mix compromise, and latency requirements, we provide a methodology aimed at systematically exploring mixnet parameters

in a way that allows selecting mixnet parameters that optimize the anonymity provided by the network. We first explain these external constraints.

Mix Compromise: When an adversary compromises a portion of the mixes of the network, it not only reduces the average level of anonymity provided by the mixnet, but also may lead to certain *unlucky* messages passing through entirely compromised paths, thereby loosing all anonymity for those messages. However, it is important to note, that the number of compromised mixes as well as their individual positions are significant factors in de-anonymizing messages. For instance, if the compromised mixes are all in one layer in a topology of let's say 3 layers, this adversary's strategy does not lead to any message having zero anonymity. Therefore we use in our methodology another metric besides entropy, which is the *fraction of fully compromised routes* which we denote by $\alpha_{\mathcal{F}}$. The fraction $\alpha_{\mathcal{F}}$ of fully compromised routes focuses on *worst-case scenarios*, i.e. messages for which all anonymity is lost as the adversary can determine *with certainty* the <sender, time, receiver> of the message. We analytically compute $\alpha_{\mathcal{F}}$ in a mixnet with L layers of width W and $N = LW$ mixes, of which A mixes are honest and B mixes are adversarial. $\alpha_{\mathcal{F}}$ is a weighted average of the fraction of compromised routes over all possible valid topologies $\mathcal{T}_v$:

$$\alpha_{\mathcal{F}} = \sum_{\mathcal{T}_v} \Pr(\mathcal{F}|\mathcal{T}_v) \Pr(\mathcal{T}_v) \tag{3.1}$$

A valid topology $\mathcal{T}_v = (\mathcal{A}, \mathcal{B})$ is defined by the number of honest and malicious mixes present in each layer, $\mathcal{A} = \{a_1, a_2, .., a_L\}$ and $\mathcal{B} = \{b_1, b_2, .., b_L\}$ such that it meets the following constraints:

$$\forall i \quad 0 \leq a_i \leq A, \quad 0 \leq b_i \leq B$$
$$\sum_{i=1}^{L} a_i = A, \qquad \sum_{i=1}^{L} b_i = B$$

We evaluate this metric for different values of L as well as different ratios of compromise for two types of network *balanced networks* are those where the number of mixes per layer is the same for all layers while *imbalanced network* are those where the number of mixes per layer can be different. We show that $\Pr(\mathcal{F}|\mathcal{T}_{opt}) = (\frac{B}{N})^L$ is a close approximation to $\alpha_{\mathcal{F}}$ and that increasing the number L of layers exponentially decreases the fraction of compromised routes for both types of network. The full analysis is explained in Part II, Publication 2. We emphasize that the inverse $\frac{1}{\alpha_{\mathcal{F}}}$ expresses the *average number of messages that need to be sent to choose one fully compromised route.* Combined with the message-sending rate of users, $\alpha_{\mathcal{F}}$ determines the de-anonymization of messages over time. For example, if $\alpha_{\mathcal{F}} = 0.0001$ and $\lambda_u = 10$ messages per second for a user, it will take on average $\frac{1}{\alpha_{\mathcal{F}} \cdot \lambda_u} = 1000$ seconds for one of the user's messages

to be routed via a fully compromised route.

Non-uniform mix capacities: While we chose to focus on uniform routing (all mixes in each layer are equally likely to be chosen in a path of a message) when computing $\alpha_{\mathcal{F}}$, we also consider the case when mixes have different capacities. When users choose the mixes to be included in the path of each message, these mixes have an assigned probability of being chosen. This 'biased (capacity-based) routing' can reflect the throughput each mix has or the reputation of mixes. We compare anonymity for both types of routing (uniform and biased) using the entropy metric and α_F and adversaries with 0% and 10% corruption rates. For example, we consider an adversary who is able to introduce mixes with 6x more capacity than the average honest mix, meaning in each layer of $W = 10$ mixes the adversarial mix has 40% of the layer's capacity and is thus chosen for 40% of the routes. This is in contrast to the uniform routing scenario where each mix routes on average 10% of the messages. With a biased routing, the adversary is able to fully compromise $\alpha_F = 6.4\%$ of routes, in contrast to $\alpha_F = 0.1\%$ of routes in the case of uniform routing. We then evaluate the effect of uniform vs biased routing on average anonymity using the entropy metric. We show that biased routing enables the adversary to not only compromise many more routes but also diminish average anonymity for the remaining messages. Based on these results we conclude that uniform routing is the best choice from an anonymity perspective.

End-To-End Latency: The average end-to-end latency in a multi-hop overlay routing aggregates the latencies at each layer the message traverses. Given that the latency at each hop is composed of network propagation time τ, packet processing time δ, and the additional time μ the message spends in the mix for anonymity—where μ is the mean of an exponential distribution representing the mix delay— the overall latency can be calculated as follows:

$$D_{e2e} = \mu L + (\tau + \delta)(L + 1) \tag{3.2}$$

We, therefore, need to adjust the per-mix latency μ when choosing the number of layers L, in order to have a fair comparison of mixnets configurations all providing the same average end-to-end latency D_{e2e}. The time τ required for messages to traverse the internet at each hop can vary significantly. Mixes within an overlay network can be distributed globally, making network propagation latency subject to various factors, including characteristics of the geographical distance, transmission medium, routing devices, and network congestion. We study the anonymity impact of propagation latency variability by comparing three scenarios with the same average τ: (i) constant propagation latency of $\tau = 50$ms for all links; (ii) variable latency per link sampled from a uniform distribution $\mathcal{U}[10, 90]$ms; and (iii) a different propagation delay per mix. These three simplified network propagation models provide a sense of the impact of inter-mix propagation variability on anonymity. We refer to other work that

provides a thorough model of network propagation latency and its impact on anonymity in mixnets [71]. We show that when μ is larger than the propagation latency τ the average anonymity is the same regardless of whether network propagation times are considered fixed or variable. On the other hand, if μ is orders of magnitude smaller than τ (Figure 5b), the variation of τ has an anonymity impact that makes message tracing harder for an adversary, and this impact is exacerbated with the number of layers in the mixnet.

3.2.2 Methodology[1]

Optimizing the number of layers and the width of the mixnet: Our proposed method for optimizing mixnet design parameters consists of three main steps. First, we set the deployment scenario and the external constraints (average end-to-end latency D_{e2e} and traffic volume $\lambda_{\mathcal{U}}$). Second, we set a threshold β ($0 < \beta < 1$) that defines the maximum tolerable fraction of compromised routes. Note that $\frac{1}{\beta}$ represents the average number of messages that need to be sent to have one fully compromised route: if we set the worst case threshold at 'one in a thousand' messages then $\beta = 0.001$, while lowering the tolerance to 'one in a million' messages corresponds to $\beta = 10^{-6}$. Given the range of values of the number L of layers and the number W of mixes per layer , we want to optimize, we compute $\alpha_{\mathcal{F}}$ for each of the values. We then discard parameter values that result in $\alpha_{\mathcal{F}} > \beta$, while keeping those that result in $\alpha_{\mathcal{F}} \leq \beta$ as candidates for the next step. The third and final step computes entropy-based anonymity, in order to find the value that maximizes *average anonymity* in addition to satisfying *worst-case anonymity* constraints. For this third step, we consider two adversaries of interest: the constant fraction adversary and the constant budget adversary. The constant fraction adversary controls a subset of B mixes that is a constant fraction b of the total number of mixes N. When considering this adversary, the number B of malicious mixes grows proportionally to network size. The constant budget adversary on the other hand controls a constant number B of malicious mixes that do not change when the network grows, with adversarial mixes therefore becoming a smaller fraction of the total network as it scales up. This adversary is of interest for systems such as the Nym network where competition among mixes for finite resources may impose a practical cap on the number of new mixes the adversary can introduce when the network grows. In a nutshell, our methodology is as follows: given the deployment scenario (average end-to-end latency D_{e2e} and traffic volume $\lambda_{\mathcal{U}}$), we find the number of layers and the number of mixes that optimize anonymity (according to the entropy and the fraction of fully compromised routes metrics) under different threat models.

[1] This section reuses text from our publication [4].

Our method includes (i) an analytical framework to compute the rate of fully corrupted paths given a level of adversarial compromise, which defines *worst-case anonymity*; (ii) an empirical method for selecting the network width and number of layers to maximize *average anonymity*, and (iii) an evaluation of the effectiveness of mix-based dummy traffic strategies to support anonymity levels in low-traffic scenarios. Our results show that the optimal number L of mixnet layers depends on the combination of adversarial compromise and end-to-end latency. Tighter latency constraints lower L, while higher adversarial compromise increases L. We note that L remains small (maximum six layers) in all considered scenarios, due to the harmful effect of thinning traffic per mix when layers are added as end-to-end latency remains constrained.

In terms of network width, narrower networks are better towards adversaries that compromise a fraction of the network, while slightly wider networks become optimal when adversaries are limited in the number of mixes they can compromise.

Dummy traffic: Throughout this work, we emphasized that the volume of traffic, determined by the number of users and their message generation rates, is considered an external factor, and hence mixnet size should be chosen accordingly. However, it is important to note that in most applications, users do not send messages at a constant rate. Instead, the flow of traffic varies according to users' habits and is influenced by the specific application in use, leading to fluctuations in traffic volume. These fluctuations can cause traffic to either peak or significantly decrease at different times throughout the day. Although mixnet size, including the number of layers and number of mixes per layer, can be updated regularly, these adjustments typically occur in epochs rather than on an ad-hoc basis in order to avoid problems such as message loss and overhead of distributing updated network information to all clients.

To address the challenge of maintaining anonymity during periods of sudden low traffic in an oversized network, we explore the strategies of dummy traffic. We evaluate two simple mix-based dummy traffic strategies, the first involves dummies that travel just one hop, offering a cost-effective solution, while the second strategy involves dummies that follow multi-hop routes and are discarded at the final hop. We evaluate these two strategies under a GPA with no corruption and with various rates of corruption (10%, 20% and 30%). We find that under rather small rates of corruption, inexpensive link-based dummies significantly improve anonymity up to the same level as the multi-hop dummies. However, when adversarial control of the mixnet is expected to be more than 20%, the more computationally expensive multi-hop dummies offer more anonymity. Our contributions are further elaborated in Part II, Publication 2.

3.3 Different traffic types

The current state-of-the-art in mixnet research focuses on individual use cases for each mixnet. This approach results in traffic from particular applications, such as email clients, remaining separate from traffic originating from other types of applications, for example, messaging apps. Alternatively, when blending different traffic from different applications does occur, all messages' delays are adjusted to have the same average end-to-end latency. However, this approach is not ideal. For applications that can tolerate latency, reducing delay might decrease their anonymity. Conversely, applications with strict latency requirements may suffer from excessive end-to-end latency, affecting the usability of the system. It is evident that users exhibit varying levels of latency tolerance depending on the application. For instance, users display higher latency tolerance for cryptocurrency transactions, with a transaction reaching up to 10 minutes for Bitcoin.[2] On the other hand, users demand lower latency for instant messages, emails, and are even more strict when it comes to web page loading, where response times should not exceed 2 seconds [37]. While studies indicate that other factors beyond speed, such as user experience and the different application features [63], influence users' choice of one application over another, the delivery speed of messages remains a critical factor of users' satisfaction [12, 65].

In [32], Dingledine et al. suggest that users can choose between security and speed when they send messages. The scheme that the authors propose works for round-based protocols by giving users a parameter α which they can increase if they want more anonymity and decrease it if they want more speed. This parameter will determine the number of rounds each message stays inside the mix before it is being flushed. The authors show that by assigning different α values, the overall anonymity of the system increases. This work was not initially proposed to blend various traffic types from different applications. However, this method closely resembles a scheme in which messages originating from different applications have distinct average delays. These different delays would not stem from users selecting a specific security score (and hence a different delay) for each message they send, but rather, would occur naturally because the messages originate from different applications. Within continuous mixnets, this approach can be implemented by sampling the per-mix delays from different exponential distributions, each having a different parameter.

The primary challenge in blending different traffic types within continuous mixnets is that the anonymity evaluation becomes complex. As discussed previously, with a single traffic type, all messages exiting a mix are equally

[2] https://medium.com/klaytn/a-comparison-of-blockchain-network-latencies-7508509b8460.

likely of being an input target message [4, 5, 24, 69]. However, blending multiple traffic types disrupts the memoryless property of the exponential distribution, complicating the computation of probabilities. Thus, we need a novel analytical method in order to be able to calculate the probability that links a target input to an output message. While the complete theoretical analysis is detailed in Part II, Publication [3], we provide an overview of the intuition behind deriving these probabilities using a simplified example of a single mix in the network in the following section.

3.3.1 Analysis for one mix:

In this work, we assume that the adversary knows the number of traffic types blended together within the mixnet, which we denote by d. It is also assumed that the adversary is able to identify the types of input messages from the client side. Although it might seem counterintuitive, given that mixnet-based systems typically employ cryptographic schemes to obscure such details, user behavior and traffic patterns could still inadvertently reveal the traffic types. Let's consider a scenario where the adversary observes two messages, m_1 and m_2, entering the mixnet at different times, with m_1 being the target message. If we assume both messages belong to the same traffic type, the probability of the first message leaving the mix is $\frac{1}{2} = 0.5$, due to the exponential distribution's memoryless property. However, if we introduce two distinct traffic types, where each message belongs to a different traffic type the memoryless property does not apply. This needs a new method to calculate the probability of an output message being the targeted input which is given by:

$$\Pr[m'_1 = m_1 | m_{\texttt{target}} \in \mathcal{T}_1 \wedge \mathcal{O}] = \frac{\lambda_1}{\sum_{j=1}^{d} k_j \lambda_j} \tag{3.3}$$

Given that the adversary knows the type of traffic of all incoming messages, this means that he also knows that there are k_j input messages of each type $\mathcal{T}_j$, $j \in \{1, 2, \ldots, d\}$ inside the mix. The numerator denotes the rate delay λ_1 of the type of traffic the target input belongs to ($\mathcal{T}_1$) and the denominator is the aggregate of the number of messages from different types of traffic (k_j) weighted by their respective delay rates. This means that there is a bias for the message m'_1 to be a specific incoming message depending on the per-mix delay values of the different traffic types. For instance, let's say we have two types of traffic: $\mathcal{T}_1$ with an average delay $d_1 = 1$ (meaning that $\lambda_1 = 1$) and the second message belongs to the second type of traffic $\mathcal{T}_2$ with an average delay $d_2 = 100$ (meaning that $\lambda_2 = 0.01$). Suppose that these two messages arrive at the same time, intuitively the first message is more likely to leave first

due to its shorter delay rate compared to the second, which is reflected by the probability calculation of $Pr[m_1 = m_{target}] = 0.99$ using Equation (3.3).

Generalisation: After the first message leaves the mix, we need to calculate the probability of the next outgoing message being the target message. At this phase, the adversary lacks information about the remaining number of messages that belong to each traffic type (k_j). Additionally, any message arriving after the first departure influences these quantities and, consequently, the probability calculations. Hence, we account for every possible combination of the different k_j. Moreover, when the network includes multiple mixes, the adversary lacks knowledge about the type of traffic of each message, resulting in probabilistic methods to estimate the probability of each outgoing message being a certain traffic type. Furthermore, we explore situations where the adversary does not know the traffic type of incoming messages, as well as cases where the traffic type is identifiable at the recipient's end. Upon completing our probability analysis, we investigate the anonymity implications of blending traffic types with various latency requirements in a single mixnet, as opposed to dedicating a different network per traffic type.

3.3.2 Impact of Blending Different Traffic Types

We updated the simulator from [5] by including the above probability derivations. We evaluate the impact of blending traffic types, each characterized by a distinct average end-to-end latency, with per-mix delays sampled from different exponential distributions, on the overall anonymity provided by the mixnet. Additionally, users generate and send these messages according to different generation parameters each corresponding to a different traffic type. Following the approach of previous work in this thesis, our anonymity evaluation uses the entropy metric, selecting a target message, m_{target}, from each traffic type. At the end of the simulation, every received message is assigned a probability of being m_{target}, which is then used in the entropy metric.

In our evaluations, we explored the impact of various generation rates per traffic type, delay parameters, threat models, and mixnet configurations on the anonymity provided by the mixnets for messages from each traffic type. We found that mixing different types of traffic within one mixnet improves anonymity compared to using separate mixnets each dedicated to one traffic type; in other words, adding messages with longer delays to a mixnet already handling messages with short delays enhances the anonymity for both types of traffic. We also examined the impact of changing traffic volumes. For example, reducing the amount of one type of traffic and replacing it with another type that has a longer average delay generally increases the anonymity for the first type of traffic. However, the degree of improvement in anonymity varies, especially

as the difference in delays between the traffic types grows. When the delay difference is very large, the benefit in terms of anonymity to the traffic with shorter delays decreases. This is due to the adversary's near ability to distinguish between the two traffic types solely based on the differences in the per-mix delays. While the anonymity for any traffic type never falls below what it would be in a mixnet dedicated to just that type of traffic, the optimal anonymity needs to be evaluated accordingly as the increase of anonymity for each type of traffic depend on various factors such as the amount of traffic of each type, the rate delays and the number of layers. The delay parameter assigned to a particular traffic type influences not just the anonymity of messages within that type, but also the messages from different traffic types. Finally, our analysis included scenarios where a single message from one traffic type is mixed with messages exclusively from another type. In such cases, this solitary message gains a notable increase in anonymity, an important difference from having no anonymity at all when it remains unblended.

3.4 Adversaries with Partial Visibility

So far our threat model exclusively considered a GPA that may additionally compromise mixes. However, as Syverson argues, in security analysis most adversaries are subject to some constraints and the success of an adversary's attack depends on their available resources and effectiveness [81]. Currently, there is a notable gap in the literature regarding methods to model and analyze anonymity properties with respect to an adversary who has partial visibility of the network. For instance, consider the case of Nym [25] in its current deployment. Even though the majority of ISPs may share information among themselves, some might not.[3] Hence the questions of how to analyze the anonymity properties in scenarios where specific message paths partially go through uncovered areas remain open. One challenge in evaluating anonymity under such an adversary is that their advantages in inferring information about messages in a network vary greatly depending on which portions of the network they have visibility over. Deriving the probability of an event analytically, considering such an adversary, requires the use of conditional probabilities based on the different assumptions about the portions of the network visible to the adversary. Even in the case of a simple network, the complexity of deriving such probabilities increases substantially.

[3] `https://www.dailydot.com/debug/sonic-isp-privacy/`.

3.4.1 A matrix-based model

In order to perform traffic analysis under an APV, we first need to model the mixnet in a way that captures this adversary. As explained earlier the adversary is modeled by its *goal*, *prior knowledge* and *capacity*. In other words, the adversary, having a prior knowledge about the network and its constraints, observes the network for an amount of time t, and tries to infer a probability of an event she is interested in and which happened either entirely or partially outside of her visibility. We denote the prior knowledge of the adversary about the network constraints by $\mathcal{C}$, by $\mathcal{O}$ its observation (the parts of the network where the adversary has the ground truth) and by $\mathcal{HS}$ the Hidden State which is the part of the network the adversary does not have under its visibility. The probability that the adversary is trying to estimate is therefore the probability $Pr[HS|O, C]$. We then need to model the entire trace of the network which is the full set of paths of all messages that traversed the network during the adversary's observation in a way that allows us to perform traffic analysis from the perspective of adversaries with partial visibility.

We note that ($\mathcal{HS}$,$\mathcal{O}$) represents the full trace of the network. We split each message path into segments so that we can differentiate between the parts of each path that an adversary can see and the parts that they cannot see. We model each mix mix_i by two set of vertices; $\mathcal{V}_I(mix_i)$ represents the input messages and $\mathcal{V}_O(mix_i)$ represents the output messages. The connections between these inputs and outputs provide a matrix that represents the mix mapping of input and output messages. Additionally, we must also represent the connections between mixes where messages are routed, since the adversary is a partial adversary and hence may not be able to see the full path. Therefore we also use the two sets of vertices $\mathcal{V}_I(mix_i)$ and $\mathcal{V}_O(mix_i)$ to represent the connections between the different entities (which can be either a mix, a sender or a receiver) of the network. In other words, we consider two types of matrices (i) inner-mix matrices that represent the *inside* of the mix and (ii) the inter-entities matrices that represent the connections between two entities. The full network trace is then represented by a set of adjacency matrices.

Definition 3.4.1 (Adjacency Matrix)**.** Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Then an adjacency matrix Mx is a matrix with $|\mathcal{V}|$ columns and $|\mathcal{V}|$ rows, where row i and column i are both labeled by vertex $v_i \in \mathcal{V}$, for $i \in [1, |\mathcal{V}|]$. Element (i, j) of Mx is 1 if the edge $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise.

The motivation behind this modeling is two-fold: First, using this model we are able to trace any message by simply following the matrices with value 1 that connects the edges in the different matrices. Second, it allows us to accommodate the adversary in our model; the inter-entity matrices represent

monitoring capability of the adversary and the inner-mix matrices represent compromising capability. For example if an adversary is corrupting a mix, we put the corresponding matrix in the observation $\mathcal{O}$ and the rest of the matrices representing other mixes in the hidden state $\mathcal{HS}$. For the inter-entity matrices however, depending on which portions of the network the adversary monitors, the same matrix has to be split into $\mathcal{O}$ and $\mathcal{HS}$.

3.4.2 Methodology

We explain in this section how to conduct traffic analysis in mixnets under an APV.

Metropolis Hastings: The Metropolis–Hastings method samples states by conducting a random walk across a state space. In our context, a state is the Hidden State $\mathcal{HS}$ of the trace. The main component of the Metropolis–Hastings method is the ratio α_{next_move} with which we decide whether or not to accept the proposed state $\mathcal{HS}_p$. α_{next_move} is given by:

$$\alpha_{next\_move} = \frac{Pr[\mathcal{HS}_c|\mathcal{O},\mathcal{C}] * Q(\mathcal{HS}_p|\mathcal{HS}_c)}{Pr[\mathcal{HS}_p \mid \mathcal{O},\mathcal{C}] * Q[\mathcal{HS}_c|\mathcal{HS}_p]} \tag{3.4}$$

In order to move from a current Hidden State $\mathcal{HS}_c$ to a proposed Hidden State, $\mathcal{HS}_p$ we first choose a matrix from the modeled trace at random. This matrix can be an inner-entity matrix (a mix) or an inter-entity matrix. Then we choose two columns from this matrix (which corresponds to vertices in the network) at random to swap. This provides a new Hidden State $\mathcal{HS}_p$ which we then use to compute α_{next_move}.

Network Representation: Recall that the adversary's goal is to be able to estimate $Pr[\mathcal{HS} \mid \mathcal{O},\mathcal{C}]$. However, this is not obvious. We need to rewrite $ratio_{next\_move}$ as follows:

$$\begin{aligned}
\alpha_{next_move} &= \frac{Pr[\mathcal{HS}_c \mid \mathcal{O},\mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p \mid \mathcal{O},\mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} && \triangleright \textit{ Bayes's rule} \\
&= \frac{Pr[\mathcal{HS}_c, \mathcal{O} \mid \mathcal{C}]/Pr[\mathcal{O} \mid \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p, \mathcal{O} \mid \mathcal{C}]/Pr[\mathcal{O} \mid \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} \\
&= \frac{Pr[\mathcal{HS}_c, \mathcal{O} \mid \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p, \mathcal{O} \mid \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} \\
&= \frac{Pr[\mathcal{TR}_c \mid \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{TR}_p \mid \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]}
\end{aligned} \tag{3.5}$$

Computing the α_{next_move} is therefore reduced to computing the probabilities of the different traces $\mathcal{TR}$ given the network constraints $\mathcal{C}$.

In a simplified stratified mixnet assuming uniform routing, each network trace is simply the set of message paths:

$$Pr[\mathcal{TR}|\mathcal{C}] = \prod_{P \in Tr} Pr[P|\mathcal{C}] \tag{3.6}$$

Any system can have a variety of constraints, such as paths lengths [83], user constraints in choosing nodes based on geo-locations, having a biased routing, etc. These constraints need to be accounted for when computing $Pr[\mathcal{TR}|\mathcal{C}]$. We compute this probability for various network constraints in Part II, Publication 4.

Traffic analysis in mixnets under an APV: Putting everything together, we update the simulator used throughout this work to include this method. We start by defining the network constraints, number of mixes, type of routing etc. Users send messages to their corresponding receivers using these constraints which result into the ground truth. We translate these messages' paths into a set of inner-entity and inter-entity matrices. We define the adversary by its goal (for example, if sender S_1 is communicating with receiver R_1) as well as the parts of the network the adversary has visibility over and the parts of the network that belong to the hidden state. We divide the matrices into two sets $\mathcal{O}$ and $\mathcal{HS}$ accordingly and finally we execute the Metropolis Hastings algorithm in order to sample the different hidden states. Each time a hidden state is accepted by the algorithm we check if the goal of the adversary is reached. We evaluate our model for two types of adversaries, one with compromising capabilities and one with monitoring capabilities. Unlike the previous work presented in this thesis, this is the first time we do not consider a GPA, rather the adversary is able to control only a portion of the network.

We assess the accuracy by running the Metropolis Hastings algorithm on N_{tr} number of traces for the same adversary each time using a different ground truth trace. After each run of the algorithm, we record the probability p_{tr} to the event of interest and $P(\mathcal{TR}_{GT})$ as an occurrence. We call an occurrence positive when $P(\mathcal{TR}_G) = 1$ and we call an occurrence negative when $P(\mathcal{TR}_G) = 0$. Once we have our full data set $(p_{tr_i}, occurrences)$ tuples, we compare the average inferred probability to the confidence score on the occurrences. We use the Wilson score interval [89], as it has been shown to be the most accurate [85] of binomial confidence intervals. The Wilson score takes a number of successes and a number of failures as inputs, and outputs the probability of success along with the 95% confidence interval. We show that our model is effective when the adversary is rather strong, meaning have access to a large portion of the network. However when the adversary is very weak, the probabilities of a given goal depend greatly on where the event happened. In this case, these probabilities are no different from the a priori knowledge. We elaborate on this analysis in Part II, Publication 4.

Chapter 4

Conclusion and Future Work

> "It may be necessary to have unanswered questions in order to continue living, struggling, and searching."
>
> Ghada Al-Samman

4.1 Conclusion

Online activities have become essential to society's needs; however, unfortunately, to meet these needs, technology has been developed in a way that has escalated privacy violations and surveillance to a critical level. This reliance on privacy-invasive technologies is likely to worsen, particularly with proposals like eIDAS [13] and CSAM [14].

Shoshana Zuboff in [90] argues that this level of surveillance is driven by what she calls *surveillance capitalism*, where major companies such as Google, Amazon, and Facebook created an economic model based on the collection, analysis, and commercialization of data. We have reached a point where even the NSA acknowledges that users are essentially paying for their own surveillance.[1]

Gürses et al. in [43] go beyond software consumption and emphasize that to better understand privacy, we should also examine the software production processes. The authors argue that the introduction of agile methods has resulted

[1] `https://billmoyers.com/story/smartphone-users-paying-for-own-surveillance/`

in software production being carried out in shorter iterations with continuous code reviews making software increasingly data-intensive.

The discussion around technology and surveillance however should not revolve around the binary choice of living without technology or accepting it as it currently stands. Instead, the critical question we must address is: how can we ensure privacy?

While we have managed to write the first part of this thesis without defining privacy, it is now time to conclude with an attempt to explain what privacy means and entails. As noted by Fiebig in [36] "Privacy is an elusive term with myriad facets and interpretations". Solove emphasizes that the definition of privacy evolves with technological advancements and cultural contexts [79]. To accurately define privacy is hard, however, researchers, lawyers, and policymakers have developed several principles that constitute what we call "privacy by design" [9, 42, 45, 73]. While there is not, and perhaps should not be, a definitive checklist of what privacy by design is, Seda Gürses in [41] identifies three paradigms: privacy as contextual integrity, privacy as control, and privacy as confidentiality. Privacy as contextual integrity, articulated by Helen Nissenbaum in [62], focuses on the appropriateness of information flows within specific contexts. She argues that privacy is maintained when personal information is shared according to the norms governing various social contexts and violations occur when information is shared in ways that deviate from these established norms. Privacy as control suggests that when there are mechanisms such as transparency and accountability in place, users can make informed decisions and have greater control over the collection and flow of their personal information. However, as Jaap-Henk Hoepman notes in [45], this paradigm implies that personal information is binary, suggesting that some data is either personal information or not, which is a misconception often held by lawyers.

The confidentiality paradigm considers any exposure of information as a loss of privacy. Privacy is holistic, meaning that where information leaks in one part of a system it can violate the privacy expectations of different parts of the system [44]. We note two main principles in this paradigm: (i) no single point of failure, meaning trust minimization and (ii) data minimization, meaning that the system design should only include data that is strictly necessary for it to function. This paradigm highlights the intrinsic connection between privacy and security; when systems centralize trust and collect data on a single entity, it is assumed that a data breach will occur at some point, compromising both security and privacy. As Preneel emphasizes "privacy is a security property" [70].

ACNs are an important part of this paradigm and represent a crucial step forward. ACNs, such as mixnets, hide metadata, which is essential for providing privacy and protecting against mass surveillance.

The work presented in this thesis contributes to the systematic evaluation of mixnets across various configurations, design choices, and threat models. We have explored research questions in mixnets by developing empirical and analytical methods. We started our work by creating a network simulator that evaluates the anonymity provided by various mixnets, each characterized by distinct design choices. This tool facilitated a deeper understanding of how these design decisions intersect. Furthermore, using this simulator, we introduced a methodology that enables us to choose mixnet parameters, such as the number of layers, the total number of nodes, and the rates of dummy traffic, in a way that optimizes anonymity. This approach is grounded in two metrics: entropy and the fraction of fully compromised paths.

We also provided the first study of the anonymity provided by continuous mixnets when blending multiple traffic types. We calculate the probabilities of linking an input message to an output message under different threat models including an adversary who has knowledge about incoming messages, outgoing messages, and compromising a portion of the mixnet. This method was incorporated into our simulator to evaluate the anonymity under varying traffic distributions, delay averages, and network setups.

Lastly, we presented a generic matrix-based model for mixnets suitable for traffic analysis against adversaries with partial visibility over the network. Using a Bayesian approach and the Metropolis-Hastings algorithm, we presented the first traffic analysis technique that captures such an adversary. It is our hope that these contributions lay a foundational framework that informs the future design of mixnets and deployment on a large scale.

4.2 Future Work

Building upon the contributions of our research, we present below a list of potential avenues for further exploration and investigation. We propose two short-term projects and two long-term projects, acknowledging that privacy is holistic and should consider technological, ethical, and legal aspects.

Bridging Theoretical Assumptions with Practical Implementations: In this thesis, we made several simplifying assumptions in order to concentrate on core unanswered questions of mixnets. Nonetheless, future research should aim to bridge the gap between theory and practice by incorporating real-world scenarios into mixnet evaluations. This would enhance their practical applicability and scalability. Specifically, investigating the impact of traffic distribution patterns and geographic diversity on mixnet performance and anonymity is crucial. Additionally, examining usage patterns of data could inform more accurate mixnet parameterization. To accomplish this, future

work could involve collecting and analyzing traffic data from diverse sources to gain a deeper understanding of actual traffic behaviors. Although we believe our findings will remain valid, integrating real-world scenarios could provide additional insights.

Towards an Advanced Network Emulator: Enhancing MiXiM for Advanced Research in Mixnets: Shadow [47] is a tool for Tor [39] research, blending simulation and emulation by executing applications as Linux processes and running them as a discrete-event network simulation. Released in 2011, it has since then been endorsed by the Tor community for research and experimentation, to facilitate the testing of design proposals and attacks. Without such a simulator, experimenting on the live Tor network can be problematic due to deployment complexities and especially privacy concerns. Having such a tool prevents the redundancy of each researcher creating their own Tor network simulator. However, given that Tor is fundamentally different from mixnets, adapting Shadow for mixnet simulations is a difficult task. Our research began with the development of MiXiM, incorporating mixnet building blocks while abstracting away several network layers, such as cryptographic operations, message buffering, and network protocols. Future work on mixnets should further develop MiXiM, aspiring for it to become a tool akin to Shadow. Such an effort would require effort from the broader community.

AI and Mixnets: In the era of AI and deep learning, the need for privacy is more critical than ever. Despite the data-intensive nature of AI algorithms, we argue that these technologies can also be leveraged to enhance privacy. In [64], the authors provide the authors demonstrate one of the first AI attacks against mixnets, by using deep learning algorithms in correlating packet flows as well as proposing countermeasures to such attacks. Given that mixnets were initially developed before the widespread adoption of AI, it is crucial to further enhance their effectiveness and resilience in the face of sophisticated deep learning analyses and attacks. This line of research is essential for advancing the integration of powerful AI algorithms with mixnet research, aiming for clearer and more impactful outcomes.

ACNs for At-Risk Communities: The lessons learned in this thesis are valuable for improving mixnets in general scenarios; however, we need to consider specific use cases to truly enhance their anonymity, utility, and future deployment. Exploring the use of ACNs for vulnerable groups is crucial. This includes evaluating how mixnets and ACNs can serve activists, individuals in women's shelters, and other at-risk populations, whose privacy needs and threat models may differ significantly from the broader public. It is essential to develop PETs tailored to unique contexts, emphasizing user-friendly designs, strong anonymity, and defenses against specific threats. By organizing workshops with various activists and human rights organizations as well as engaging in

multidisciplinary fields we can have a better understanding of the various unique privacy needs and challenges. Wang et al. set the standard for future designs in PETs [86]. By combining rigorous research and collaborations with key organizations active in the field, they present a safe and secure digital aid-distribution system, demonstrating the practical application of privacy by design principles. Similarly, Tor has exemplified how to build a project for at-risk communities such as activists and journalists by organizing workshops, publishing blogs, and continuously enhancing usability so that it does not differ significantly from other major browsers. These efforts have made Tor more accessible and user-friendly, thereby increasing its adoption and effectiveness in providing privacy on the network level against local adversaries.

"Anonymity loves company" as put by Dingledine et al. in [31] and hence for a mixnet-based system to be deployed and used at a large scale, lessons should be learned from several projects. Privacy on the network level achieved by ACNs is only one piece of the puzzle to achieve privacy and resist mass surveillance, nevertheless a crucial one. In almost every privacy and security conference, the importance of metadata is frequently discussed, highlighting the challenges for deploying ACNs. We hope that this thesis has answered several important questions and that future research will build upon it.

Bibliography

[1] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. MCMix: Anonymous Messaging via Secure Multiparty Computation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1217–1234, Vancouver, BC, August 2017. USENIX Association.

[2] Lamiaa Basyoni, Noora Fetais, Aiman Erbad, Amr Mohamed, and Mohsen Guizani. Traffic Analysis Attacks on Tor: A Survey. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 183–188. IEEE, 2020.

[3] Iness Ben Guirat, Debajyoti Das, and Claudia Diaz. Blending Different Latency Traffic with Beta Mixing. *Proceedings on Privacy Enhancing Technologies*, 2024.

[4] Iness Ben Guirat and Claudia Diaz. Mixnet optimization methods. *Proceedings on Privacy Enhancing Technologies*, page 456–477, 2022.

[5] Iness Ben Guirat, Devashish Gosain, and Claudia Diaz. MiXiM: Mixnet Design Decisions and Empirical Evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, 2021.

[6] Hal Berghel. Malice Domestic: The Cambridge Analytica Dystopia. *Computer*, 51(5):84–89, 2018.

[7] Dan Boneh and Philippe Golle. Almost Entirely Correct Mixing With Application to Voting. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, pages 68–77, November 2002.

[8] Jon Callas, Lutz Donnerhacke, Hal Finney, David Shaw, and Rodney Thayer. OpenPGP message format. Technical report, 2007.

[9] Ann Cavoukian et al. Privacy by design: The 7 foundational principles. *Information and privacy commissioner of Ontario, Canada*, 5:12, 2009.

[10] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[11] David Chaum, Farid Javani, Aniket Kate, Anna Krasnova, Joeri Ruiter, Alan T Sherman, and Debajyoti Das. cMix: Anonymization by high-performance scalable mixing. Technical report, 2016.

[12] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. Quantifying skype user satisfaction. *ACM SIGCOMM Computer Communication Review*, 36(4):399–410, 2006.

[13] European Comission. Regulation of the european parliament and of the council amending regulation (eu) no 910/2014 as regards establishing a framework for a european digital identity, 2021.

[14] European Comission. Proposal for a regulation of the european parliament and of the council laying down rules to prevent and combat child sexual abuse, 2022.

[15] 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*, pages 1–17. Springer, 2003.

[16] George Danezis. Statistical disclosure attacks. In *IFIP International Information Security Conference*, pages 421–426. Springer, 2003.

[17] George Danezis. The traffic analysis of continuous-time mixes. In *International Workshop on Privacy Enhancing Technologies*, pages 35–50, Berlin, Heidelberg, 2004. Springer.

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

[19] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *2009 30th IEEE Symposium on Security and Privacy*, pages 269–282. IEEE, 2009.

[20] George Danezis and Ian Goldberg. Sphinx: A Compact and Provably Secure Mix Format. In *2009 30th IEEE Symposium on Security and Privacy*, pages 269–282, 2009.

[21] George Danezis and Carmela Troncoso. Vida: How to use Bayesian inference to de-anonymize persistent communications. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 56–72. Springer, 2009.

[22] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Divide and funnel: a scaling technique for mix-networks. Cryptology ePrint Archive, Paper 2021/1685, 2021. https://eprint.iacr.org/2021/1685.

[23] Yves-Alexandre De Montjoye, Laura Radaelli, Vivek Kumar Singh, and Alex "Sandy" Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, 2015.

[24] Claudia Diaz. *Anonymity and Privacy in Electronic Services.* PhD dissertation, KU leuven, December 2005.

[25] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym Network. https://nymtech.net/nym-whitepaper.pdf, February 2021.

[26] Claudia Diaz, Steven J. Murdoch, and Carmela Troncoso. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 184–201, Berlin, Heidelberg, 2010. Springer-Verlag.

[27] Claudia Diaz and Bart Preneel. Taxonomy of mixes and dummy traffic. In *Information Security Management, Education and Privacy*, pages 217–232. Springer, 2004.

[28] Claudia Diaz, Len Sassaman, and Evelyne Dewitte. Comparison between two practical mix designs. In *Proceedings of ESORICS 2004*, LNCS, September 2004.

[29] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards Measuring Anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 54–68, Berlin, Heidelberg, 2002. Springer-Verlag.

[30] Whitfield Diffie and Susan Landau. *Privacy on the line: The politics of wiretapping and encryption.* The MIT Press, 2010.

[31] Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *WEIS*, 2006.

[32] Roger Dingledine, Andrei Serjantov, and Paul Syverson. Blending different latency traffic with alpha-mixing. In *Privacy Enhancing Technologies: 6th International Workshop, PET 2006, Cambridge, UK, June 28-30, 2006, Revised Selected Papers 6*, pages 245–257. Springer, 2006.

[33] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *Proceedings of Privacy Enhancing*

Technologies workshop (PET 2004), volume 3424 of *LNCS*, pages 186–206, May 2004.

[34] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of encrypted and VPN traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pages 407–414, 2016.

[35] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[36] Tobias Fiebig, Martina Lindorfer, and Seda Gürses. Position paper: Escaping academic cloudification to preserve academic freedom. *Privacy Studies Journal*, 1(1):49–66, 2022.

[37] Dennis F Galletta, Raymond Henry, Scott McCoy, and Peter Polak. Web site delays: How tolerant are users? *Journal of the Association for Information Systems*, 5(1):1–28, 2004.

[38] Nethanel Gelernter, Amir Herzberg, and Hemi Leibowitz. Two cents for strong anonymity: The anonymous post-office protocol. In *International Conference on Cryptology and Network Security*, pages 390–412. Springer, 2017.

[39] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In *Proceedings of the First International Workshop on Information Hiding*, page 137–150, Berlin, Heidelberg, 1996. Springer-Verlag.

[40] Marcin Gomulkiewicz, Marek Klonowski, and Miroslaw Kutylowski. Rapid mixing and security of chaum's visual electronic voting. In *Proceedings of ESORICS 2003*, October 2003.

[41] Seda Gürses. Can you engineer privacy? *Commun. ACM*, 57(8):20–23, aug 2014.

[42] Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design. *Computers, Privacy & Data Protection*, 14(3):25, 2011.

[43] Seda Gurses and Joris V J van Hoboken. Privacy after the agile turn, May 2017.

[44] Harry Halpin. Holistic privacy and usability of a cryptocurrency wallet. *arXiv preprint arXiv:2105.02793*, 2021.

[45] Jaap-Henk Hoepman. *Privacy is hard and seven other myths: Achieving privacy through careful design*. MIT Press, 2021.

[46] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[47] Rob Jansen and Nicholas Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Proceedings of the 19th Symposium on Network and Distributed System Security (NDSS)*. Internet Society, February 2012.

[48] Rob Jansen, Marc Juarez, Rafa Galvez, Tariq Elahi, and Claudia Diaz. Inside Job: Applying Traffic Analysis to Measure Tor from Within. In *NDSS*, 2018.

[49] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop- and- Go-MIXes providing probabilistic anonymity in an open system. In David Aucsmith, editor, *Information Hiding*, pages 83–98, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[50] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-And-Go-MIXes Providing Probabilistic Anonymity in an Open System. In *International Workshop on Information Hiding*, pages 83–98. Springer, 1998.

[51] Dogan Kesdogan, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Breaking anonymity by learning a unique minimum hitting set. In *International Computer Science Symposium in Russia*, pages 299–309. Springer, 2009.

[52] Dogan Kesdogan and Lexi Pimenidis. The hitting set attack on anonymity protocols. In *International Workshop on Information Hiding*, pages 326–339. Springer, 2004.

[53] Joe Kilian and Kazue Sako. Receipt-free MIX-type voting scheme - a practical solution to the implementation of a voting booth. In *Proceedings of EUROCRYPT 1995*. Springer-Verlag, May 1995.

[54] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable Messaging System with Cryptographic Privacy. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 759–776, Santa Clara, CA, February 2020. USENIX Association.

[55] 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)*, pages 711–725, Carlsbad, CA, October 2018. USENIX Association.

[56] Jonathan Mayer, Patrick Mutchler, and John C Mitchell. Evaluating the privacy properties of telephone metadata. *Proceedings of the National Academy of Sciences*, 113(20):5536–5541, 2016.

[57] Michela Meister and Karen Levy. Digital security and reproductive rights: Lessons for feminist cyberlaw. *Feminist Cyberlaw (Meg Leta Jones and Amanda Levendowski, eds.), University of California Press, Forthcoming*, 2022.

[58] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster protocol—version 2. 2003.

[59] Joseph D Mornin. NSA metadata collection and the Fourth Amendment. *Berkeley Tech. LJ*, 29:985, 2014.

[60] Steven J Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 183–195. IEEE, 2005.

[61] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, 2001.

[62] Helen Nissenbaum. Privacy as contextual integrity. *Wash. L. Rev.*, 79:119, 2004.

[63] Solomon O Ogara, Chang E Koh, and Victor R Prybutok. Investigating factors affecting social presence and user satisfaction with mobile instant messaging. *Computers in Human Behavior*, 36:453–459, 2014.

[64] Lennart Oldenburg, Marc Juarez, Enrique Argones Rúa, and Claudia Diaz. Mixmatch: Flow matching for mixnet traffic. *Proceedings on Privacy Enhancing Technologies*, 2024.

[65] Piyusha Paradkar. User tolerance of message transmission delay in electronic mail: A cross-national comparison. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 44:368–371, 07 2000.

[66] Trevor Perrin and Moxie Marlinspike. The double ratchet algorithm. *GitHub wiki*, page 112, 2016.

[67] Andreas Pfitzmann and Marit Hansen. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer, 2001.

[68] 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*, pages 39–44, 2021.

[69] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix Anonymity System. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1199–1216, Vancouver, BC, August 2017. USENIX Association.

[70] Bart Preneel. The future of cryptography. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9665:XVI–XVII, 2016.

[71] Mahdi Rahimi, Piyush Sharma Kumar, and Claudia Diaz. Larmix: Latency-Aware Routing in Mix Networks. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*. Internet Society, February 2024.

[72] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Designing Privacy Enhancing Technologies*, pages 10–29. Springer, 2001.

[73] Peter Schaar. Privacy by design. *Identity in the Information Society*, 3(2):267–274, 2010.

[74] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 41–53, Berlin, Heidelberg, 2002. Springer-Verlag.

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

[76] Yan Shi and Subir Biswas. Website fingerprinting using traffic analysis of dynamic webpages. In *2014 IEEE Global Communications Conference*, pages 557–563. IEEE, 2014.

[77] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of ESORICS 2006*, September 2006.

[78] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1928–1943, 2018.

[79] Daniel J Solove. *Understanding privacy*. Harvard university press, 2010.

[80] Gavin Sullivan. *The Law of the List*, page i–i. Global Law Series. Cambridge University Press, 2020.

[81] Paul Syverson. Why i'm not an entropist. In *International Workshop on Security Protocols*, pages 213–230. Springer, 2009.

[82] Paul Syverson. Why I'm not an entropist. In *Proceedings of Security Protocols XVII: 17th International Workshop, April 2009, Revised Selected Papers*, pages 231–239. Springer-Verlag, LNCS 7028, 2013.

[83] Carmela Troncoso and George Danezis. The bayesian traffic analysis of mix networks. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, page 369–379, New York, NY, USA, 2009. Association for Computing Machinery.

[84] John Viega, Matt Messier, and Pravir Chandra. *Network security with openSSL: cryptography for secure communications.* " O'Reilly Media, Inc.", 2002.

[85] Sean Wallis. Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods. *Journal of Quantitative Linguistics*, 20(3):178–208, 2013.

[86] Boya Wang, Wouter Lueks, Justinas Sukaitis, Vincent Graf Narbel, and Carmela Troncoso. Not yet another digital id: privacy-preserving humanitarian aid distribution. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 645–663. IEEE, 2023.

[87] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, 2014.

[88] Tao Wang and Ian Goldberg. On Realistically Attacking Tor with Website Fingerprinting. *Proceedings on Privacy Enhancing Technologies*, (4):21–36, 2016.

[89] Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[90] Shoshana Zuboff. Surveillance capitalism and the challenge of collective action. In *New labor forum*, volume 28, pages 10–29. SAGE Publications Sage CA: Los Angeles, CA, 2019.

Part II

Publications

List of Publications

International Journals

1. Ben Guirat, I., and Diaz, C. Mixnet optimization methods. *Proceedings on Privacy Enhancing Technologies* (2022), 456–477
   - See p. 73
2. Ben Guirat, I., Das, D., and Diaz, C. Blending Different Latency Traffic with Beta Mixing. *Proceedings on Privacy Enhancing Technologies* (2024)
   - See p. 115

International Conferences and Workshops with Proceedings

1. Ben Guirat, I., Gosain, D., and Diaz, C. MiXiM: Mixnet Design Decisions and Empirical Evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society* (2021)
   - See p. 57
2. Ben Guirat, I., Diaz, C., Eldefrawy, K., and Zeilberger, H. Traffic Analysis by Adversaries with Partial Visibility. In *28th European Symposium on Research in Computer Security (ESORICS)* (2023), Springer
   - See p. 155

International Workshop

Ben Guirat, I., Gosain, D., and Diaz, C. MiXiM: A general purpose simulator for mixnet. *13th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2020)* (2020)

Publication

Mixim: Mixnet design decisions and empirical evaluation

Publication Data

Ben Guirat, I., Gosain, D., and Diaz, C. MiXiM: Mixnet Design Decisions and Empirical Evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society* (2021)

Contributions

- Principal author.

MiXiM: Mixnet Design Decisions and Empirical Evaluation

Iness Ben Guirat[1], Devashish Gosain[2], and Claudia Diaz[1]

[1] COSIC, KU Leuven, Belgium
[2] Max Planck Institute for Informatics

Abstract. In this paper we present MiXiM, a simulation framework for mixnets that allows researchers to evaluate different design options and their tradeoffs. This framework is flexible and allows to quickly run experiments to assess combinations of mixnet building blocks, such as mixing strategies and network topologies, as well as study the effect of different parameters related to each component. The framework provides results for a number of metrics including anonymity, end-to-end latency and traffic overhead.

1 Introduction

In the last decades, a variety of overlay networks [2, 12, 15, 18, 20] have been proposed to provide communication anonymity, meaning that in these networks it is not possible to find out who is communicating with whom. Mixnets are a type of anonymous communication network that aims to be secure against *global network adversaries*, who observe all communications in the underlying network. Even though the concept of mixnets [1] predates onion routing [10] by more than a decade, and early mixnet deployments [3, 16] were operative before Tor, their uptake has remained far behind for years, mainly due to their higher computational requirements, added latency, and lack of industrial-quality implementations. In recent years however a number of mixnet designs have been proposed [12, 15, 18] and currently the Nym network is implementing a mixnet-based anonymity network already deployed as a testnet prototype [6].

Like other types of anonymity networks, mixnets are complex systems with many components and parameters, including mixing strategies, routing policies, and dummy traffic strategies. Questions like—what is the best combination of mixing strategies and topologies, or what is the impact of the number of mixes per layer on anonymity, *etc.* have not been explored systematically.

Each of these components and its parametrization can have a significant impact on the anonymity provided by a network. In order to select components and tune the parameters of a mixnet configuration, designers need to be able to evaluate the anonymity resulting from possible design decisions in different conditions. Simulation is commonly used in research studies on the Tor network, particularly in instances where the experiments that need to be conducted would endanger actual users if they were deployed in the live network. Instead, many attacks on Tor can be evaluated with Shadow [13] in a safe manner. Shadow provides an isolated environment for simulating Tor network communications. It produces traces and logs that can be further analyzed to assess what an adversary would be able to learn with access to different subsets of the logged data.

In this paper we introduce MiXiM, a generic mixnet simulation framework to evaluate anonymity in different designs and configurations. MiXiM's level of abstraction focuses on elements core to anonymous routing, including mixnet topology, routing policy, mixing algorithms, cover traffic and mix corruption. We focus on documenting the metadata such as packets sources, destinations and timings exposed by the mixnet, while making abstraction of data payloads and cryptographic operations. In addition, MiXiM captures a number of relevant metrics (latency, bandwidth overhead etc.) that can be used to study tradeoffs between anonymity, performance and cost. The simulator and all related tooling are publicly available.[3]

2 Background and related work

A *mixnet* is an overlay network of **mixes**. Mixes are servers that cryptographically transform and reorder messages, such that inputs are unlinkable to outputs both in terms of message appearance and timing—even by an adversary who can monitor *all* the messages going in and out of the mix. Message reordering can be achieved with different "mixing" strategies [8]. MiXiM presently supports four popular algorithms:

- *Threshold Mix:* A threshold mix [1] buffers messages in the queue until a set number T (threshold parameter) is reached. At that point the mix permutes the T messages, flushing them in a random order.
- *Timed Mix:* A timed mix buffers messages in the queue for a set time interval (timeout). When the time is elapsed, the mix permutes the messages it has collected in the interval, flushing them in a random order.
- *Pool Mix:* These are a variation of both *threshold* and *timed* mixes where, instead of flushing all messages, a fraction of messages is kept [3]. MiXim

[3] `https://gitlab.esat.kuleuven.be/Iness.BenGuirat/mixim`

supports pool mixes but they are left out of the experiments presented here due to space limitations.

- *Stop-and-Go Mix:* a SG-mix [14] processes messages individually and continuously in time. Each received message is kept for a random amount of time (sampled from an exponential distribution) and sent out when the time has elapsed. Due to the memory-less properties of exponential distributions, individual delays result in a reordering of the sequence of messages that has high entropy [4].

Mixnet topology is the arrangement that defines how mixes are interconnected in the mixnet and thus which routes (sequences of mixes) exist for messages to follow.

- *Cascade:* In this topology, all messages traverse a fixed sequence of mixes forming a chain with a predetermined order: each mix sends messages to the next mix in the cascade [1]. A single cascade cannot scale to handle more throughput than that of a single mix. Thus, if more capacity is required, multiple cascades can run in parallel [2, 11].
- *Stratified:* In stratified or *layered* topologies mixes are arranged in a fixed number of layers where each mix, at any given time, is assigned to only one specific layer. The layers are interconnected such that each mix in layer i receives messages from mixes in layer $i-1$ and sends messages to mixes in layer $i+1$. There are two variants of stratified topology, (i) Fully connected where mixes in layer i can send messages to all mixes in layer $i+1$, and (ii) Not Fully connected where mixes in layer i can send messages to only a subset of mixes in layer $i+1$.

3 MiXiM Framework

The MiXiM framework consists of configuration files that define the mixnet environment, a discrete-event simulator that instantiates and executes the network, producing observations that are logged to files, and analysis scripts to process the log files and extract empirical results.

3.1 Configuration

Configuration files define the topology of the mixnet, routing policy, client and network traffic characteristics, and optional features such as cover traffic and mix corruption. These configuration values are input to the simulator that instantiates the network and its components (as processes) and then simulates its execution by generating and processing events.

C	Total number of clients
λ_C	Mean rate of message generation per client
mix-type	Timed, threshold, pool, or Poisson
T_0	Timeout between mix flushes (Timed mix)
T	Threshold parameter (Threshold mix)
μ	Mean message delay (Poisson mix)
net-type	Cascades, Stratified Fully Connected (FC) or Restricted
l	Path length
N_M	Number of mixes
λ_D	Mean rate of dummy traffic per mix
α	Fraction of corrupted mixes

Table 1: Summary of notation for Publication 1.

Client information: Sets the total number of clients C, the message generation rate per client λ_C, and the duration of the simulation.

Mixing information: Sets the mix types with their parameters *e.g.,* μ for Poisson mix, *timeout* for Timed mix, and *threshold* for Threshold mix.

Topology information: Sets the topology with its parameters *e.g.,* number of layers, the number of mixes per layer, the number of cascades, *etc.*

Optional information: If desired, a system designer can also generate dummy traffic. In MiXiM, dummy traffic is generated according to a Poisson process with parameter λ_D. In addition, MiXiM supports mix corruption where α is the fraction of corrupted mixes. Table 1 provides a handy reference for all the parameters.

3.2 Implementation

The MiXiM simulator is a discrete event simulator. It is built with SimPy[4] and models all the building blocks of a mixnet as described in Section 2. The components (mixes, clients, and messages) are processes that exist in the same environment. Once the mixnet parameters are chosen, MiXiM loads the configuration file and instantiates the simulation environment where all processes live in. They interact with the environment and with each other via *events*.

The simulator instantiates the mixnet as follows: first it creates the *Network* process according to the configuration, including the number of mixes, the topology, the types of mixes and the number and positions of corrupt mixes. Then MiXiM instantiates each of the *Mixes* with their specific descriptions

[4]A python discrete event simulation library, https://simpy.readthedocs.io/en/latest/

(types and parameters). After this step, *Client* processes are created that in turn generate and send *Messages*. We model client message sending behaviour as a Poisson process[5] with the parameter λ_C. MiXiM can be extended to implement alternative distributions to model other client sending behaviours that may be of interest.

The simulator runs for the specified time and then computes the evaluation metrics from the log files. Since MiXiM is a discrete event simulator, simulation time and "wall-clock" time do not progress in-step. We denote the simulation time unit of SimPy as t_u. Furthermore, upon start-up the network requires a **burn-in** period time to become **stable**. Stability means that each mix in the mixnet is receiving and transmitting the expected number of messages following the specified parameters in the configurations files.

MiXiM evaluates the anonymity of messages that are transiting the network once it is stable. The simulator logs all relevant information (probability distributions over messages, latencies, etc.) to files. The final stage of the workflow is the analysis step where scripts[6] parse these log files to compute the results. We use entropy as metric for anonymity [9].

3.3 Entropy

Entropy is an information theoretic measure of the uncertainty associated with a random variable. In anonymous communications, entropy captures the uncertainty of an adversary about which is the target message of interest amongst all of the other messages in the network that it could be confused with. The larger the size of this set and the more uniform the probability distribution of being the target among all the messages, the more anonymous is the target [9, 19].

In MiXiM when all the mixes are stable, the simulator automatically chooses a target message m_t. The simulated adversary starts to track all the M messages in the network, including the target. Because the target message, as well as all the messages in the network, goes through multiple mixes—where they are delayed and shuffled with other messages—the adversary assigns a probability, $0 \leq \Pr[m_i = m_t] \leq 1$, to each message of being the target message, with $i = 1 \ldots M$. After the experiment ends, the simulator computes the entropy of

[5] Poisson processes are extensively used to model natural phenomena, such as user message sending habits.

[6] The framework provides scripts for all the analysis conducted in this paper. However, the logging feature is extensible and can allow additional evaluations with new scripts.

the probability distributions defined by $\Pr[m_i = m_t]$ to evaluate anonymity, as:

$$-\sum_{i=1}^{M} \Pr[m_i = m_t] \cdot \log_2(\Pr[m_i = m_t])$$

4 Mixnet building blocks

In this section, we demonstrate the usefulness of the simulator by walking through a rather simple yet complete mixnet-based system assessing the different parameters and design decisions.

4.1 Mixing

In this experiment we study the impact of different mixing strategies on anonymity (ref. Section2 for more details on mixing strategies).

Experimental setup: We set the average end-to-end latency $L_e = 1$s, the number of clients $C = 100$ and two values of λ_C *i.e.,* $\lambda_C = 1$ and $\lambda_C = 10$. In a Poisson mix $L_e = 1$s translates to $\mu = 1$, while in a timed mix it corresponds to a timeout $T_0 = 2$s. Given the considered message rates λ_C, we adjust the threshold parameter T to achieve $L_e = 1$s and perform a fair comparison. This results in the following parameter values:
For $C \cdot \lambda_C = 100$: $\mu = 1$, $T_0 = 2$s, $T = 200$.
For $C \cdot \lambda_C = 1000$: $\mu = 1$, $T_0 = 2$s, $T = 2000$.

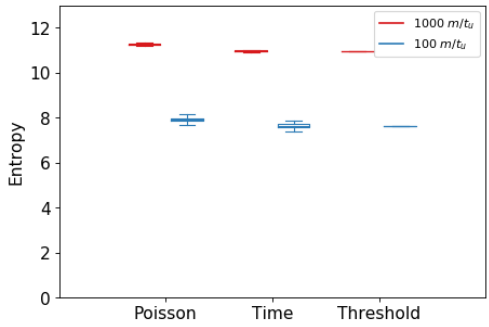


Figure 1: Impact of mixing strategies on entropy.

Result: In Figure 1 it is evident that the three types of mixing provide approximately the same entropy, with a slight advantage for Poisson mixing.

4.2 Topologies

Mix cascades are a classical design [1] that has been used in many proposals, such as XRD [15], Vuvuzela [12], and cMix [2]. Stratified topologies have been recommended as optimal topologies for anonymous routing in prior studies [7] and are used in systems such as Loopix [18]. Although XRD [15] and Loopix [18] seem different, they are both designed for scalability, aim at hiding metadata, and are capable of storing messages. The main differences between the two types of designs are (1) topology: Vuvuzela, cMix and XRD's topology consists of organizing the mixes into small chains (cascades) each acting as a local mixnet, while Loopix is a fully connected stratified network; and (2) mix types: Threshold vs Poisson. Inspired by these designs, we experimentally compare three types of topologies *i.e.,* fully connected stratified, not fully connected stratified, and multiple cascades.

Multiple cascades have the drawback of splitting the anonymity set: users who are connected to one cascade are completely distinguishable from users who are connected to another cascade. To avoid this problem, XRD [15] implements a rather sophisticated scheme of cascade selection. The scheme guarantees that every pair of users have at least one cascade in common, as shown in Figure 3. Since our aim is to analyze the impact of underlying topology on anonymity, we do not include all the features of Loopix and XRD, and only compare their topologies. For example in XRD, every client connected to N_c cascades, sends $N_c - 1$ dummy messages each time they send a real message. Moreover, every message (either real or dummy) goes through a different chain. This is very expensive in terms of bandwidth and thus we chose to ignore this feature.

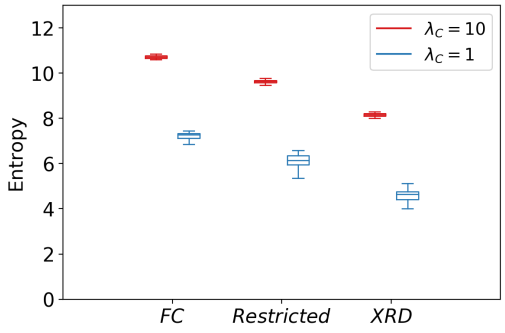


Figure 2: Impact of different topologies on entropy.

Experimental setup: Since we already studied the impact of different mixing strategies and our aim in this experiment is to analyze the impact of topologies on anonymity, we consider Poisson mixing for all the experiments and only

vary the network topology. We evaluate the following network topologies for $C = 100$ and two values of $\lambda_C = 1$ and $\lambda_C = 10$:

- Fully connected (FC) stratified: 3 layers of 6 mixes each.
- Restricted (not fully connected) stratified: 3 layers of 6 mixes each, but each mix is only connected to 2 mixes in the adjacent layers.
- XRD (multiple cascades): 6 cascades of 3 mixes each.

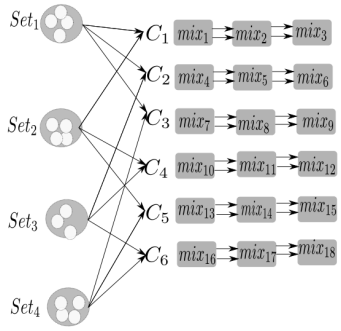


Figure 3: Arrangement of cascades in XRD.

Figure 3 represents the scenario where we simulate $C = 100$ clients, that results in 4 sets of 25 users each (as per XRD scheme). Every set is connected to one group of 3 cascades. Suppose we have 6 cascades C_i, $i = 1 \dots 6$, then users' sets are connected to the following cascades respectively: $Set_1 = \{C_1, C_2, C_3\}$, $Set_2 = \{C_1, C_4, C_5\}$, $Set_3 = \{C_2, C_4, C_6\}$, $Set_4 = \{C_3, C_5, C_6\}$. It must be noted that each pair of user sets intersects with every other set in at least one cascade. This scheme guarantees that the set of users is not split into disjoint subsets.

Results: Figure 2 shows that even though we have the same amount of traffic, same mix types and same number of nodes, the network topology greatly influences the level of anonymity: stratified topologies, and in particular fully connected, provide more anonymity than multiple cascades. This is due to the fact that the second layer of mixing aggregates all messages in one large anonymity set, while in cascades the users of each cascade remain partitioned.

4.3 Layers and Average Delay

To build a mixnet based system, another important parameter to decide on is the the number of layers. On one hand adding layers will make the messages mix with each other more therefore anonymity will increase, but on the other hand adding layers increases the risk of packet loss. Additionally, to increase

anonymity one might also opt for increasing the average delay of each message per mix [5].

Experimental setup: In Figure 4, we show results for $C = 100$ with $\lambda_C = 10$, a stratified topology of 10 mixes per layer, with all mixes as Poisson mixes. We study the impact of the number layers l and the average delay for each message per mix μ on anonymity.

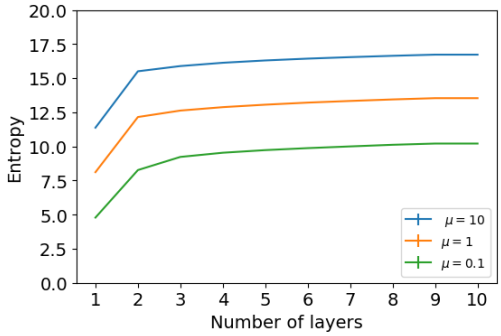


Figure 4: Variation of entropy depending on the number of layers and the average delay of each message per mix.

Result: Increasing the latency of messages in each mix has a higher impact on anonymity than adding layers. In fact, after 4 layers the entropy barely increases. However going from an average delay of $0.1t_u$ to $10t_u$ per mix in 3 by 10 topology will increase anonymity by more than 6 bits which means an increase of anonymity set size that is $2^6 = 64$-fold.

4.4 Mix Corruption

We now evaluate anonymity for different fractions mixes that are controlled by the adversary. For these mixes the adversary knows with full certainty the correspondence between inputs and outputs, rather than having probabilistic information on possible correspondences.

Experimental setup: In Figure 5 we show results for a fully connected stratified topology with 10 Poisson mixes per layer with $\mu = 0.1$, and $C = 100$ clients with sending rate $\lambda_C = 10$. We increase the number of layers and study the impact of different fractions of mix corruption on anonymity.

In the previous subsection we showed that, with zero corruption, anonymity does not substantially increase after 3 layers. However, when considering mix corruption, we observe that more layers results in better anonymity. For instance

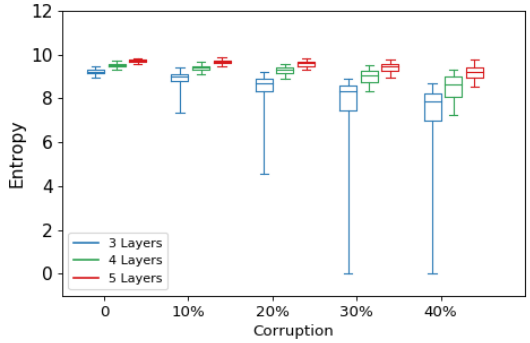


Figure 5: Impact of fraction of corruption (α) on entropy.

in Figure 5, for 40% corruption, anonymity in a three layer topology has a high standard deviation compared to a four or five layer topology. This is because the chances of a message traversing a fully corrupted path (*i.e.,* encountering a corrupted mix in each layer) are higher when the number of layers is small. With three layers a non-negligible fraction of messages (6%) traverse a fully corrupted path.

4.5 Cover Traffic

Oya et al. [17] described two types of cover traffic: (i) Client-based dummy traffic and (ii) Mix-based dummy traffic. Since we assume that all messages are cryptographically indistinguishable, client-based dummy traffic is considered the same as real traffic. Therefore any change in the amount of traffic, real or client-based dummy, will impact the anonymity in exactly the same way. Mix-based dummy traffic has a different effect, and next we evaluate its impact on anonymity.

Experimental setup: We evaluate anonymity while varying the rate of client traffic λ_C (for $C = 100$ clients) and mix-based dummy traffic λ_D. In Figure 6 we show results for a 4 (layers) by 10 (mixes) fully connected stratified topology of 40 Poisson mixes with $\mu = 0.1$.

We observe in Figure 6 that dummy traffic significantly increases anonymity when the amount of real traffic is low. *E.g.,* when $\lambda_C = 0.1$ (real traffic is $C \cdot \lambda_C$, *i.e.,* 10 messages/t_u), the median entropy increases from 0 to 5 bits with sufficient dummy traffic. However, when the amount of real traffic is high (when $\lambda_C = 10$) adding dummies does not have any major impact on anonymity.

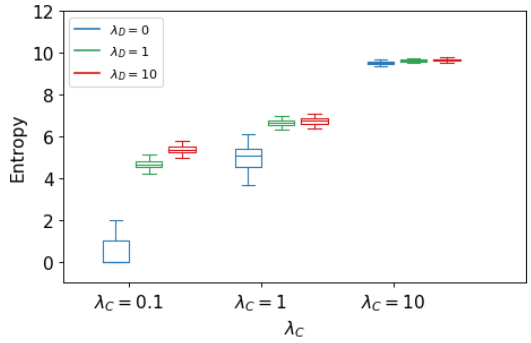


Figure 6: Impact of mix dummy traffic on entropy.

5 Conclusion

Mixnets are anonymous communication networks that provide anonymity against a passive global network adversary. Mixnets have many parameters, *e.g.,* number of layers, types of mixes, underlying topological structure, real and cover traffic rates, number of clients, *etc.* These parameters interplay in complex ways to provide a certain level of anonymity to routed messages. Thus, in this paper we proposed MiXiM, a framework with which one can i) efficiently understand the interplay of these parameters and, ii) evaluate and design mixnet configurations. MiXiM already supports a variety of design options and configurations that can be easily extended to conduct further studies and perform systematic evaluations of mixnet designs.

Acknowledgments

We would like to thank Tariq Elahi for early discussions and feedback on this work. This research is partially supported by the Research Council KU Leuven under the grant C24/18/049, by CyberSecurity Research Flanders with reference number VR20192203, and by DARPA FA8750-19-C-0502. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the funders.

References

[1] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[2] David Chaum, Farid Javani, Aniket Kate, Anna Krasnova, Joeri Ruiter, Alan T Sherman, and Debajyoti Das. cMix: Anonymization by high-performance scalable mixing. Technical report, 2016.

[3] Lance Cottrell. Mixmaster and remailer attacks, 1995.

[4] George Danezis. The traffic analysis of continuous-time mixes. In *International Workshop on Privacy Enhancing Technologies*, pages 35–50, Berlin, Heidelberg, 2004. Springer.

[5] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity Trilemma: Strong anonymity, Low Bandwidth Overhead, Low latency - Choose two. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 108–126, 2018.

[6] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym Network. `https://nymtech.net/nym-whitepaper.pdf`, February 2021.

[7] Claudia Diaz, Steven J. Murdoch, and Carmela Troncoso. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 184–201, Berlin, Heidelberg, 2010. Springer-Verlag.

[8] Claudia Diaz and Bart Preneel. Taxonomy of mixes and dummy traffic. In *Information Security Management, Education and Privacy*, pages 217–232. Springer, 2004.

[9] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards Measuring Anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 54–68, Berlin, Heidelberg, 2002. Springer-Verlag.

[10] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, page 21, USA, 2004. USENIX Association.

[11] Nethanel Gelernter, Amir Herzberg, and Hemi Leibowitz. Two cents for strong anonymity: The anonymous post-office protocol. In *International Conference on Cryptology and Network Security*, pages 390–412. Springer, 2017.

[12] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[13] Rob Jansen and Nicholas Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Proceedings of the 19th Symposium on Network and Distributed System Security (NDSS)*. Internet Society, February 2012.

[14] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-And-Go-MIXes Providing Probabilistic Anonymity in an Open System. In *International Workshop on Information Hiding*, pages 83–98. Springer, 1998.

[15] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable Messaging System with Cryptographic Privacy. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 759–776, Santa Clara, CA, February 2020. USENIX Association.

[16] Nick Mathewson and Roger Dingledine. Mixminion: Strong anonymity for financial cryptography. In *Proceedings of Financial Cryptography (FC '04)*, pages 227–232. Springer-Verlag, LNCS 3110, February 2004.

[17] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. Do dummies pay off? limits of dummy traffic protection in anonymous communications. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 204–223. Springer, 2014.

[18] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix Anonymity System. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1199–1216, Vancouver, BC, August 2017. USENIX Association.

[19] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 41–53, Berlin, Heidelberg, 2002. Springer-Verlag.

[20] Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. A survey on routing in anonymous communication protocols. *ACM Computing Surveys (CSUR)*, 51(3):1–39, 2018.

Publication

Mixnet Optimization Methods

Publication Data

Ben Guirat, I., and Diaz, C. Mixnet optimization methods. *Proceedings on Privacy Enhancing Technologies* (2022), 456–477

Contributions

- Principal author.

Mixnet Optimization Methods

Iness Ben Guirat and Claudia Diaz

COSIC, KU Leuven, Belgium

Abstract. We propose a method to optimally select mix network parameters for a given deployment context and adversarial model. Our method considers both worst-case and average-case anonymity and selects configurations that meet worst-case constraints while maximizing average anonymity. We apply our methods to mixnet size optimization to determine the number and width of mixnet layers, and provide results for various deployment and adversarial scenarios. For cases where the deployment context suddenly changes (drop in user traffic) we evaluate countermeasures based on mix-generated dummy traffic and show that inexpensive link dummies can significantly boost protection in some of these cases.

1 Introduction

A mix network, or *mixnet*, is an overlay network of mix nodes that routes messages anonymously from senders to receivers [7, 8, 15, 21, 31, 37, 40, 45]. Messages are encrypted by senders multiple times using, e.g., the Sphinx packet format [16], and then routed through a sequence of mix nodes. Each of the mix nodes decrypts, pads and re-randomizes messages to make its output messages *cryptographically unlinkable* to its input messages. Mix nodes also retain messages for a randomized amount of time to alter their flow and make node inputs and outputs *probabilistically unlinkable* with respect to message order and timing.

Even though the concept of mixnets [7] predates onion routing [26, 28, 29] by more than a decade, and early mixnet deployments [10, 40] were operative before Tor,[1] their uptake has remained far behind for years, mainly due to their higher computational requirements, added latency, and lack of industrial-quality implementations. In recent years however, given a renewed interest in anonymity systems that resist global network adversaries, novel mixnet-based anonymity network designs have been proposed [31, 35, 45] and currently Nym[2]

[1] `https://www.torproject.org`
[2] `https://nymtech.net`

is developing a mixnet-based anonymity network [21], already deployed as a testnet prototype that counts thousands of nodes.[3]

Traffic analysis is a collection of statistical methods to make inferences from available metadata, in particular: a data transmission's source, destination, size and timing [14]. The traffic analysis of mixnets yields probabilistic information that describes the likelihood of input messages corresponding to outputs [12, 20, 32, 45, 47, 48, 51, 52, 53]. This likelihood is greatly affected by the mixnet parameters, with some configurations providing significantly better protection from traffic analysis than others. Notably, this includes the network topology parameters, which describe the network size, how nodes are connected, and how multi-hop routes are selected [4, 50]. A suboptimal design can provide very poor privacy – or even no protection at all: while the *anonymity trilemma* [19] is informative of the *theoretical upper bound* on the anonymity that an abstract system can offer given conditions of traffic volume and end-to-end latency, the *practical lower bound* for those very same traffic and latency conditions is actually *zero* anonymity if the network is inadequately parametrized, e.g., if it is grossly oversized.

Given an expected volume of user traffic, constraints on end-to-end latency, and a threat model of concern, we currently lack methods to optimally select mixnet parameters, e.g., to decide how many nodes the mixnet should have and how they should be arranged to maximize protection from traffic analysis. The observation that a limited network size is desirable so that traffic density per mix node is sufficient for the mixing to be effective has been made in prior work [11, 51]. We are however the first to propose a methodology to systematically select mixnet parameter values given deployment and adversarial constraints, which results in configurations that respect worst-case anonymity bounds while maximizing average anonymity.

Our methods apply to a class of mixnets broadly defined by continuous-time mixes [32] arranged in a layered network topology [21,45], considering adversaries that observe all network connections in addition to controlling a subset of mixes. We specify and justify our system model in Section 2, where we also describe the considered adversarial capabilities.

We consider two anonymity metrics: (i) *worst-case anonymity* (expressed as the probability of selecting a fully compromised route), and *average anonymity* (given by the entropy of the probability distribution that relates a target input to the mixnet outputs [25, 49]). In Section 3 we provide analytical methods to compute worst-case anonymity and empirical methods to compute average anonymity. Given these two metrics, we propose a mixnet parametrization methodology in Section 4 that maximizes average anonymity while respecting worst-case anonymity constraints.

[3] `https://github.com/nymtech/`

We discuss our experimental setup in Section 5. We first show that variability in network propagation delays and multi-core message processing can help prevent message tracing in practice when the per-mix added latency is very small (these effects become negligible as per-mix added latency grows larger). We then show that uniform node selection per layer offers better anonymity than biased capacity-based selection, which allows modest-budget adversaries to arbitrarily increase the fraction of fully compromised routes.

Section 4 presents our optimization results. We first apply our methods to optimizing the number of mixnet layers and show results for various end-to-end latency constraints, considering different adversarial capabilities and worst-case anonymity thresholds. We then apply the method to the mixnet width, again considering various adversarial models. We finally consider scenarios where a network optimized for a certain level of user traffic suffers a large and sudden drop in traffic volume. We study two countermeasures based on mix-generated dummy traffic. We find that link-based dummies are a cheap yet effective strategy to support anonymity levels in scenarios with moderate mixnet compromise. We expand by considering scenarios with higher levels of adversarial compromise. Finally, we review prior work on mixnet optimization in Section 7 and offer our conclusions in Section 8.

2 System and threat model

2.1 System model

There are two basic types of entities in a mixnet: *end users* who anonymously send and receive messages, and *mix nodes* that act as intermediaries, routing messages between senders and receivers. We model the user population as sending messages with a rate that follows a Poisson process, considering high and low traffic load scenarios. We consider *source-routed decryption* mixnets of *continuous-time mixes*, with *fully connected layered* network topologies, and three strategies for *dummy traffic*. The rest of this section explains and justifies these choices.

2.1.1 Source routing

We consider *decryption mixnets* that are source routed, i.e., where the sender of a message selects the route through the network until it reaches the receiver. Preparing a message for sending requires encrypting it with public key material of the mix nodes selected by the sender as intermediaries in the route. The encryption is done in reverse order: starting with the recipient, adding a layer

of encryption for each predecessor in the route, ending with the first mix node that receives the message directly from the sender. Upon receiving a message, mix nodes use their private keys to strip a layer of encryption and discover the next hop in the route. After a randomized delay, the message is forwarded to the next hop, which is either another intermediary mix node or the end receiver. Sphinx [16] is the best known cryptographic packet format for source-routed mixnet messages [21, 45, 46].

An alternative to decryption mixnets is *re-encryption mixnets*, which are typically cascades where batches of encrypted messages are re-randomized and provably shuffled multiple times before being threshold-decrypted [5, 30, 33, 38]. Such mixnets are specially tailored to voting applications as use cases that have limited and predictable traffic volume, very high latency tolerance, and strict public verifiability requirements. The anonymity provided by such re-encryption cascades is essentially proportional to the size of the batch where a message is mixed, considering the number of voting choices and distribution of votes (e.g., if *all* voters vote for the same candidate, then there is no voting privacy for anyone as everyone's voting choice is revealed by the tally). Given their limited range of application and straightforward anonymity tradeoffs (simply dependent on batch size), we consider re-encryption mixnets as out of scope in this paper, which focuses on the optimization of decryption mixnets for scalable, general-purpose message-based communications [21].

2.1.2 Topology

The topology of a mixnet defines how mix nodes are inter-connected and which routes (sequences of mixes) messages can follow. The earliest mixnet proposals considered **mix cascades**, where a batch of messages goes through a fixed sequence of mixes [7,43]. Cascades have however two main drawbacks: scalability and fault tolerance. A single server has a performance limit, and thus parallel cascades must be created in order to serve more users. As parallel cascades are disjoint, they do not combine all users in one large anonymity set, failing to take advantage of user growth to offer better anonymity [11, 22]. This makes cascades rather uninteresting for anonymity optimization. Furthermore, the failure of a single node invalidates the whole cascade, making cascades very vulnerable to server failures compared to other topologies [4, 27].

The other traditional anonymity network topology is **free routes** [10, 26, 36], where nodes form a fully connected graph and any random walk in the path (up to a maximum path length) is a valid message route. The evaluation of anonymity in free route networks requires complex and inefficient analysis methods, even for simple threshold-mix based mixnet designs [52]. Moreover,

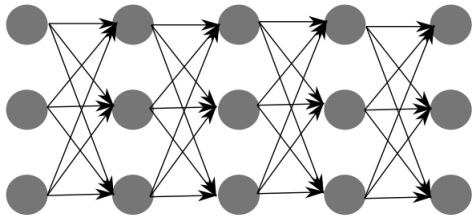


Figure 1: Layered mixnet with $N = 15$, $L = 5$, $W = 3$.

free route networks have been shown to offer worse anonymity than *layered* (or *stratified*) topologies when compared in the same conditions [22, 51].

In **layered** topologies mixes are arranged in a number of layers where each mix, at any given time, is assigned to exactly one layer. The layers are interconnected such that each mix in layer i receives messages from mixes in layer $i - 1$ and sends messages to mixes in layer $i + 1$, as shown in Figure 1. Mixes in the first layer receive messages from senders, while those in the last layer send messages to end recipients. The path length of message routes is fixed and determined by the number of layers. Valid message routes traverse a mix of each layer in the correct order. Layers can be *fully connected*, meaning that all nodes of a layer are connected to all predecessors in the previous layer and all successors in the next layer, or *restricted*, i.e. subject to constraints where nodes connect to a subset of predecessors and successors rather than all of them. Prior work has found no significant difference between the anonymity provided by fully connected and restricted layers [22].

In this paper we focus on *layered networks with fully connected layers* of the same size. We assume the topology is periodically reshuffled to allow for churn and adjust to changes in network scale. We also assume the assignment is neither predictable nor biasable by an adversary, who cannot influence the placement of malicious nodes in the layers. This can be achieved for example by relying on a public random beacon and a verifiable random function as proposed in [21]. We say a layered network has *balanced layers* if all L layers have the same number of nodes or *width* W, with the total number of nodes being $N = LW$. We choose networks with balanced layers as our baseline for their better load balancing properties. For completeness, we also include results for *imbalanced layers*, where some layers may have more nodes than others.

2.1.3 Mixing

In our model we consider continuous-time mixes with exponential delay [32] as they are known to offer excellent anonymity properties [13] and also allow

fine-tuning the added latency per mix node to offer predictable end-to-end latency [21, 45]. This is in contrast to threshold and pool mixes [7, 37] where latency varies with the traffic volume per mix (the more traffic, the less latency), making them impractical for use cases that require latency to be within certain bounds [23]. The per-mix delays are sampled by the message sender and encoded in the Sphinx headers. Upon receiving and decrypting a message, a mix extracts the delay from the header, keeps the message in its internal memory for that amount of time, and then forwards it to its next destination.

2.1.4 Dummy traffic

Dummy or *cover* traffic are automatically generated messages introduced for privacy purposes. Dummy messages contain no data payload and are discarded by their final recipient. If the dummy messages follow multi-hop paths, they are considered indistinguishable from actual messages at intermediate hops as well as towards the underlying network, i.e., only the source and destination of a dummy message know that it is a dummy message. Intuitively dummy traffic enables unobservability properties [42], meaning that it is not possible to tell whether a user is idle or actively communicating. In addition, by virtue of increasing the mixnet traffic dummies also contribute to higher anonymity for actual messages.

In this paper we consider three types of cover traffic, illustrated in Figure 2. First, *user-generated dummies* destined to themselves (*loops*, as in Loopix [45]) or to others. We note that for all network purposes this traffic is equivalent to real user traffic. Second, *link-based dummies*, this type of dummy traffic is generated by mixes and it is dropped at the next hop. Third, *partial-route dummies* are also generated by the mixes in all but the last layer, and dropped at the last mixnet layer. This dummy traffic causes a linear increase of traffic load transiting the network after each layer. Other dummy strategies are possible, e.g., dummies can be generated by mixes and dropped by end users [39]. We however consider this impractical as it requires mixes to maintain a list of end user keys and addresses.

2.1.5 User traffic

We consider a user population $\mathcal{U}$ that, as a whole, generates messages following a Poisson process with parameter $\lambda_{\mathcal{U}}$ messages per second, i.e., messages arrive to the mixnet at intervals that follow an exponential distribution of mean $\frac{1}{\lambda_{\mathcal{U}}}$ seconds. We note that only messages generated by honest users (not controlled by the adversary) are relevant to anonymity, and thus $\mathcal{U}$ and $\lambda_{\mathcal{U}}$ exclude malicious users. Furthermore, users may generate end-to-end dummy traffic

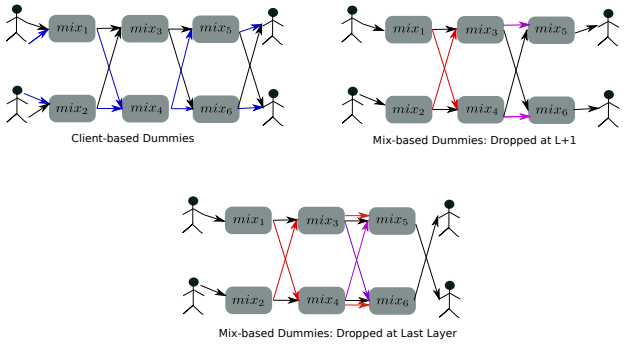


Figure 2: Types of Dummy Traffic.

destined to themselves or to other users. Since this end-to-end traffic follows the same mixnet routes as real traffic and it is fully indistinguishable, we consider that $\lambda_{\mathcal{U}}$ accounts for *all* honest-user-generated traffic, whether real or dummy. Internet traffic varies per day and per hour of the day and large variations in the amount of user traffic arriving to a mixnet are possible. We consider two scenarios: (i) *High volume* user traffic ($\lambda_{\mathcal{U}} = 5000$ messages per second) and (ii) *Low Volume* user traffic ($\lambda_{\mathcal{U}} = 100$ messages per second). We note that $\lambda_{\mathcal{U}}$ is an external deployment constraint (not a parameter chosen by the mixnet designer), and our methodology can be applied to any concrete value of $\lambda_{\mathcal{U}}$.

2.2 Threat model

We consider **global passive network adversaries** that have a *global view* on the network, meaning that they can observe all network links and take into account all messages sent between any two participants (end users or mixes) with their timing information. We assume messages have the same size, and thus only timing information is exploitable to correlate a node's inputs to outputs. If the adversary in addition controls a set of **malicious users**, any messages generated by those users are excluded from $\lambda_{\mathcal{U}}$.

In addition, the adversary controls a subset of mix nodes. Mix nodes controlled by the adversary provide no anonymity to the messages they route, as the adversary knows the mapping between the inputs and outputs of malicious nodes. This is contrast to honest mixes, for which the adversary can only obtain probabilistic information linking their inputs and outputs based on message arrival and departure observations [13, 32]. We consider two adversaries of interest: the *constant fraction adversary* and the *constant budget adversary*.

The **constant fraction adversary** controls a subset of B mix nodes that is a constant fraction b of the total number of nodes N, i.e., $B = bN$, while $A = N - B$ denotes the number of honest nodes. When considering this adversary, the number B of malicious mixes grows proportionally to network size N. The **constant budget adversary** on the other hand controls a constant number B of malicious mix nodes that does not change when the network grows, with adversarial nodes therefore becoming a smaller fraction of the total network as it scales up. This adversary is of interest for systems such as the Nym network [21], where competition among mix nodes for finite resources (representing node reputation) may impose a practical cap on the number of new nodes the adversary is able to introduce when the network grows.

3 Anonymity metrics

We evaluate anonymity in mixnets using two metrics: (i) **fraction of fully compromised routes** [1, 6], and (ii) **entropy** [25, 49]. The metrics express different aspects of anonymity, with the first focusing on worst-case scenarios and the second on average-case scenarios. Table 1 provides a summary of the notation used for the various relevant parameters.

3.1 Fraction of fully compromised routes

The fraction $\alpha_{\mathcal{F}}$ of fully compromised routes focuses on *worst-case scenarios*, i.e. messages for which all anonymity is lost as the adversary can determine *with certainty* the <sender, time, receiver> of the message. This happens when the message passes through a fully compromised route, meaning that at every mixnet layer, the node in the message's route is adversarial. Note that the inverse $\frac{1}{\alpha_{\mathcal{F}}}$ expresses the *average number of messages that need to be sent to choose one fully compromised route.*

We analytically compute the expected fraction $\alpha_{\mathcal{F}}$ of fully compromised message routes in a mixnet with L layers of width W and $N = LW$ mix nodes, of which A nodes are honest and B nodes are adversarial ($A + B = N$). We consider networks of equal-capacity mix nodes where the topology is periodically reshuffled, so that the adversary cannot choose where malicious nodes are placed (in which layer). Let $\mathcal{F}$ denote the event of a fully compromised route. We compute $\alpha_{\mathcal{F}}$ as a weighted average of the fraction of compromised routes over all possible valid topologies $\mathcal{T}_v$:

Notation	Description
N	Total number of nodes
L	Number of layers
W	Width of the network
W_{min}	Minimum width of the network
b	Fraction of adversarial nodes ($\frac{B}{N}$)
D_{e2e}	Average end-to-end latency
μ	Average per-mix delay
τ	Link propagation time
δ	Per-mix processing time
$\lambda_{\mathcal{U}}$	Rate of user-generated traffic
$\lambda_{\mathcal{M}}$	Rate of mix-generated dummy traffic
A	Number of honest nodes
a_i	Number of honest nodes in layer i
B	Number of corrupted nodes
b_j	Number of corrupted nodes in layer j
$\mathcal{F}$	Event of fully compromised route
$\alpha_{\mathcal{F}}$	Fraction of fully compromised routes
β	Maximum tolerated $\alpha_{\mathcal{F}}$
$\mathcal{T}_v$	Valid topology
$\mathcal{T}_e$	Topology with at least one empty layer
$\mathcal{T}_{opt}$	Optimal adversarial topology
m_i	$i-$th message
m_t	Target message
$\Pr_L[m_i = m_t]$	Probability that output m_i is the target
$\Pr_{Mix}[m_t]$	Probability that m_t is in the mix

Table 1: Summary of notation for Publication 2.

$$\alpha_{\mathcal{F}} = \sum_{\mathcal{T}_v} \Pr(\mathcal{F}|\mathcal{T}_v) \Pr(\mathcal{T}_v) \tag{1}$$

A valid topology $\mathcal{T}_v = (\mathcal{A}, \mathcal{B})$ is defined by the number of honest and malicious nodes present in each layer, $\mathcal{A} = \{a_1, a_2, .., a_L\}$ and $\mathcal{B} = \{b_1, b_2, .., b_L\}$ such that it meets the following constraints:

$$\forall i \quad 0 \leq a_i \leq A, \quad 0 \leq b_i \leq B \tag{2}$$

$$\sum_{i=1}^{L} a_i = A, \qquad \sum_{i=1}^{L} b_i = B \tag{3}$$

$$\forall i \quad a_i + b_i = W \tag{4}$$

$$N = A + B = LW \tag{5}$$

$\Pr(\mathcal{T}_v)$ expresses the likelihood of occurrence of a certain valid topology $\mathcal{T}_v$, and given $\mathcal{T}_v$, $\Pr(\mathcal{F}|\mathcal{T}_v)$ expresses the probability of choosing a fully compromised route in that topology. In layered networks this means choosing a malicious mix node at *every* layer. The choice of nodes in a message's route is made uniformly at random and independently per layer, and thus $\Pr(\mathcal{F}|\mathcal{T}_v)$ corresponds to the product of the fraction of compromised nodes in each layer:

$$\Pr(\mathcal{F}|\mathcal{T}_v) = \prod_{i=1}^{L} \frac{b_i}{a_i + b_i} \tag{6}$$

Note that the adversary does not compromise any full route if he fails to populate one of the mixnet layers, i.e. if $b_i = 0$ for any layer i. At the other end of the spectrum, the optimal topology $\mathcal{T}_{opt}$ for the adversary (with highest fraction of compromised routes) is when adversarial nodes are equally distributed across layers, i.e. when $b_i = \frac{B}{L}$, $i = 1..L$. In this adversarial best case, the fraction of fully compromised routes is:

$$\Pr(\mathcal{F}|\mathcal{T}_{opt}) = \prod_{i=1}^{L} \frac{\frac{B}{L}}{W} = (\frac{B}{N})^L \tag{7}$$

3.1.1 Balanced layers

To compute the likelihood $\Pr(\mathcal{T}_v)$ of a valid topology $\mathcal{T}_v = (\mathcal{A}, \mathcal{B})$, we note that in networks with balanced layers $\mathcal{A}$ and $\mathcal{B}$ are not independent. The mapping of honest nodes $\mathcal{A} = \{a_1, a_2, .., a_L\}$ is deterministic with respect to $\mathcal{B}$ as $a_i = W - b_i$, and thus $\Pr(\mathcal{B})$ fully determines the likelihood of a topology $\Pr(\mathcal{T}_v)$ (the inverse is equivalent: fixing $\mathcal{A}$ fully determines $\mathcal{B}$ as $b_i = W - a_i$).

$\Pr(\mathcal{B})$ is modeled by a hypergeometric distribution that initially has a population of size N, with B objects of interest, and W draws without replacement. The number b_1 of malicious nodes selected for the the first layer is given by:

$$\Pr(b_1) = \frac{\binom{B}{b_1}\binom{N-B}{W-b_1}}{\binom{N}{W}} \tag{8}$$

The number b_j of malicious nodes in the subsequent layers $j = 2 \dots L-1$ is given by a hypergeometric distribution with updated parameters to account for the (honest and malicious) nodes already assigned to the previous layers:

$$N_j = N - W(j-1) \tag{9}$$

$$B_j = B - \sum_{k=1}^{j-1} b_k \tag{10}$$

$$\Pr(b_j|b_1,..b_{j-1}) = \frac{\binom{B_j}{b_j}\binom{N_j - B_j}{W - b_j}}{\binom{N_j}{W}} \tag{11}$$

The last layer is deterministically composed by the leftover nodes:

$$b_L = B - \sum_{k=1}^{L-1} b_k \tag{12}$$

Thus, the probability of a valid topology $\mathcal{T}_v = (\mathcal{A}, \mathcal{B})$ with an assignment of nodes to layers $\mathcal{B} = \{b_1, b_2, .., b_L\}$ and $\mathcal{A} = \{a_1, a_2, .., a_L\}$ with $a_i = W - b_i \ \forall i$ is given by:

$$\Pr(\mathcal{T}_v) = \Pr(\mathcal{B}) = \prod_{j=1}^{L-1} \frac{\binom{B_j}{b_j}\binom{N_j - B_j}{W - b_j}}{\binom{N_j}{W}} \tag{13}$$

Putting everything together, we obtain:

$$\alpha_{\mathcal{F}} = \sum_{\mathcal{B}} \left(\prod_{i=1}^{L} \frac{b_i}{W}\right) \prod_{j=1}^{L-1} \frac{\binom{B_j}{b_j}\binom{N_j - B_j}{W - b_j}}{\binom{N_j}{W}} \tag{14}$$

Figure 3 shows $\alpha_{\mathcal{F}}$ in networks of a hundred nodes organized in two to five layers, considering 10% to 30% adversarial nodes. We depict with stars the value given by $\Pr(\mathcal{F}|\mathcal{T}_{opt}) = (\frac{B}{N})^L$ and find that it is a close approximation of $\alpha_{\mathcal{F}}$ due to the small variance of the distribution (by the law of big numbers, the variance of $\alpha_{\mathcal{F}}$ becomes smaller as the network size grows). Given that $\frac{B}{N} < 1$, increasing the number L of layers exponentially decreases the fraction of compromised routes, e.g., in a network where the adversary controls 10% of the nodes, 1% of messages are compromised with 2 layers, one in a thousand messages with 3 layers, one in ten thousand with 4 layers, and so on. Combined with the message sending rate of users, $\alpha_{\mathcal{F}}$ determines the de-anonymisation of messages over time. For example, if $\alpha_{\mathcal{F}} = 0.001$ and $\lambda_u = 5$ messages per second for a user u, it will take on average $\frac{1}{\alpha_{\mathcal{F}} \cdot \lambda_u} = 200$ seconds for one of u's messages to be routed via a fully compromised route.

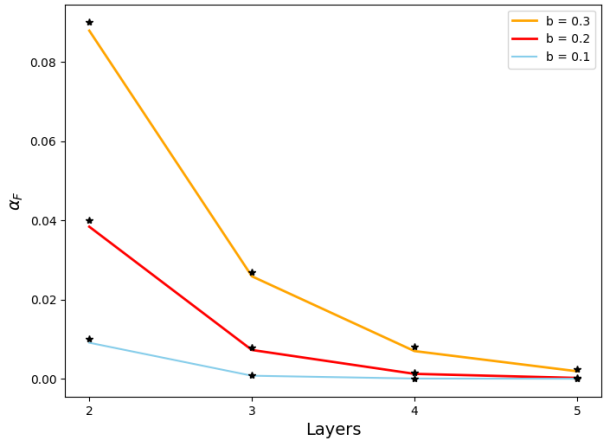


Figure 3: Fraction $\alpha_{\mathcal{F}}$ for different values of L and B in a network of a hundred nodes.

3.1.2 Imbalanced layers

In this section we develop an analysis analogous to the one in Section 3.1 for the case of layered mixnets with imbalanced layers, i.e., mixnets where layers have variable width. This is the case if the algorithm assigns nodes to layers independently for each node, e.g., deriving the layer from the node's public key and a public random beacon [21]. We represent layer widths with a vector $(w_1, w_2, ..., w_L)$, where w_i is the width of layer i. Since the choice of layer is made independently per node, the assignment is modeled by a multinomial distribution with N trials, L categories and uniform probability $\frac{1}{L}$ over the categories. The probability of having a layer size distribution $(w_1, w_2, ..., w_L)$ is subject to the constraint that $\sum_{i=1}^{L} w_i = N$ and given by:

$$Pr(w_1, w_2, ..., w_L) = \frac{N!}{\prod_{i=1}^{L} w_i!}(\frac{1}{L})^N \tag{15}$$

We consider that only mixnet topologies with at least one node per layer are considered valid, i.e., we require that $\forall i\ w_i > 0$, to ensure that messages always go through L mixes. If a topology selection results in a mixnet where $w_i = 0$ for any layer i, the selection is discarded and the assignment is re-sampled with updated randomness.

We consider that the N nodes are split into A honest nodes and B malicious nodes, $N = A + B$, with distribution over the layers given by vectors $\mathcal{A} = \{a_1, a_2, .., a_L\}$ and $\mathcal{B} = \{b_1, b_2, .., b_L\}$, where $w_i = a_i + b_i$. The fraction of

compromised routes $\alpha_{\mathcal{F}}$ is computed with Eq. (1), subject to the constraints expressed in Eq. (2), Eq. (3), and Eq. (5). The difference in node assignment however does invalidate the constraint in Eq. (4), which is instead substituted by:

$$\forall i, \quad a_i + b_i > 0 \tag{16}$$

Given a valid mixnet topology $\mathcal{T}_v = (\mathcal{A}, \mathcal{B})$, the fraction $\Pr(\mathcal{F}|\mathcal{T}_v)$ is computed with Eq. (6). We now derive the distribution $\Pr(\mathcal{T}_v)$ of valid topologies $\mathcal{T}_v = (\mathcal{A}, \mathcal{B})$, defined by the number of honest and malicious nodes in each layer, $\mathcal{A} = \{a_1, a_2, .., a_L\}$ and $\mathcal{B} = \{b_1, b_2, .., b_L\}$, subject to the already mentioned constraints.

In imbalanced networks each node's layer assignment is done independently, and thus the probability $\Pr(\mathcal{A}, \mathcal{B})$ of a topology $(\mathcal{A}, \mathcal{B})$ can be computed as the probability of two independent assignments $\mathcal{A}$ and $\mathcal{B}$, i.e., $\Pr(\mathcal{A}, \mathcal{B}) = \Pr(\mathcal{A}) \Pr(\mathcal{B})$. $\Pr(\mathcal{A})$ and $\Pr(\mathcal{B})$ are each described by a multinomial:

$$\Pr(\mathcal{A} = \{a_1, a_2, .., a_L\}) = \frac{A!}{\prod_{i=1}^{L} a_i!} (\frac{1}{L})^{\sum_{i=1}^{L} a_i} \tag{17}$$

$$\Pr(\mathcal{A} = \{a_1, a_2, .., a_L\}) = \frac{A!}{L^A \prod_{i=1}^{L} a_i!} \tag{18}$$

$$\Pr(\mathcal{B} = \{b_1, b_2, \ldots, b_L\}) = \frac{B!}{L^B \cdot \prod_{i=1}^{L} b_i!} \tag{19}$$

We recall that topologies with empty layers are discarded. We define $\mathcal{T}_e$ as the set of topologies with at least one empty layer, i.e., topologies that meet the constraints in Eq. (2), Eq. (3), and Eq. (5), but violate the constraint in Eq. (16) for at least one layer. We define a normalization factor Z that accounts for the aggregate probability of choosing a topology that is discarded due to empty layers:

$$Z = \frac{A!B!}{L^N} \sum_{(\mathcal{A},\mathcal{B}) \in \mathcal{T}_e} (\prod_{j=1}^{L} a_j! \prod_{k=1}^{L} b_k!)^{-1} \tag{20}$$

The probability of selecting a valid topology $\mathcal{T}_v = (\mathcal{A}, \mathcal{B})$ that meets all constraints is re-normalized considering $(1 - Z)$, to account for discarded topologies $\mathcal{T}_e$:

$$\Pr(\mathcal{A},\mathcal{B}) = \frac{A!B!}{(1-Z)L^N} \sum_{(\mathcal{A},\mathcal{B})\in\mathcal{T}_v} (\prod_{j=1}^{L} a_j! \prod_{k=1}^{L} b_k!)^{-1} \tag{21}$$

Putting everything together we obtain:

$$\alpha_{\mathcal{F}} = \frac{A!B!}{(1-Z)L^N} \sum_{(\mathcal{A},\mathcal{B})\in\mathcal{T}_v} \prod_{i=1}^{L} \frac{b_i}{a_i+b_i} (\prod_{j=1}^{L} a_j! \prod_{k=1}^{L} b_k!)^{-1} \tag{22}$$

The results for $\alpha_{\mathcal{F}}$ in this case are nearly identical to those obtained for balanced networks and shown in Figure 3, meaning that the expected fraction of fully corrupt paths is the same regardless of whether layers are balanced or imbalanced. Furthermore, in networks of a hundred nodes the variance is so low that $\alpha_{\mathcal{F}}$ can be safely approximated by $(\frac{B}{N})^L$.

As side observation, note that the optimal topology for the adversary in *imbalanced* networks is a corner case where all mixnet layers but one have a single node, which happens to be adversarial, with a lone layer containing all the rest of nodes. The fraction of compromise in this case would be $\frac{B-(L-1)}{N-(L-1)}$. Such scenarios are rare but possible in toy-sized networks but their likelihood quickly becomes negligible for any realistic network sizes. Large networks have overwhelming probability of being close to balanced for the same reason that casting a fair die many times yields roughly the same counts for each side, with relative variance only affecting small sample sizes.

We finally note that imbalanced layers do not present any advantage over balanced layers, and as disadvantage they typically incur in a small loss of overall mixnet throughput, which is bounded by the layer with the least capacity. Furthermore, particularly in small networks, imbalanced layers present worst-cases that provide more advantage to the adversary than the worst-case of balanced networks. Therefore, we argue that network topologies with balanced layers should be preferred when designing a mixnet, as they minimize capacity waste caused by imbalances in the sizes of different layers and avoid scenarios that could give outsize advantage to the adversary.

3.2 Entropy

Instead of a worst-case metric, entropy provides an *average measure* of the number of candidate messages that the adversary confuses with a target message [25, 49]. Entropy metrics capture network scaling as their maximum possible value grows with the number of users. An entropy of, e.g., 10 bits, indicates that a message is as anonymous as if it was perfectly indistinguishable

among about a thousand ($2^{10} = 1024$) other messages, while 11 bits correspond to perfect indistinguishability among $2^{11} = 2048$ messages. Note that the scale is logarithmic, and that an increase of one bit of entropy *doubles* the size of the equivalent perfect indistinguishability set, while a drop of one bit *halves* it.

Compared to worst-case metrics that only account for rates of fully compromised routes, entropy metrics account for anonymity in all possible scenarios, weighted by their likelihood of occurrence. For example, from a worst-case perspective it does not matter whether the adversary can guess the sender of an anonymous message with probability 1% or 99.9%; it only matters whether the adversary can *fully determine* the sender (100% certainty), or not. Entropy metrics are not as blunt as this binary determination and instead consider that messages can be *more* or *less* anonymous depending on the uncertainty of the adversary about the real sender. Thus, with entropy metrics a message for which the adversary is *almost* certain of the sender is considered very similarly to a message for which the adversary is *completely* certain of the sender – in contrast to worst case metrics where the *'almost'* case is considered adversarial failure and the *'completely'* case adversarial success. Furthermore, entropy metrics account for the probabilistic information obtained by network adversaries in addition to corrupt adversarial nodes, while the worst-case metric is only dependent on adversarial nodes and disregards probabilistic inferences made by network adversaries (because nothing short of full route compromise is relevant to the worst case).

Computing entropy metrics requires obtaining the probability distribution that links a target input message to all possible output messages, or conversely one target output to all possible inputs. Given the complexity of mixnets, obtaining the relevant distributions cannot be done in a closed analytical form. In line with prior work [22, 23, 45, 51], we resort to using a discrete-event mixnet simulator [2] that given an experimental setup generates message traces, defines a subset of the traces as adversarial observations, and computes anonymity given those observations.

We consider a user population that generates messages following a Poisson process with rate $\lambda_{\mathcal{U}}$ messages per second sent to the mixnet. Messages are routed through the mixnet until they reach their destination, and in the process they leave traces that are used for anonymity evaluation. The simulation environment allows the adversary to choose a target message m_t and compute the probability $0 \leq \Pr_L[m_i = m_t] \leq 1$ linking that target input to all possible outputs m_i after the last mixnet layer L, as illustrated in Figure 4.

For each target input m_t, we are interested in the probability that each output message m_i may correspond to that target. We do so by associating a probability $\Pr_l[m_i = m_t]$ to each message at layer l. Message probabilities are computed iteratively per layer and updated each time a message enters and leaves a

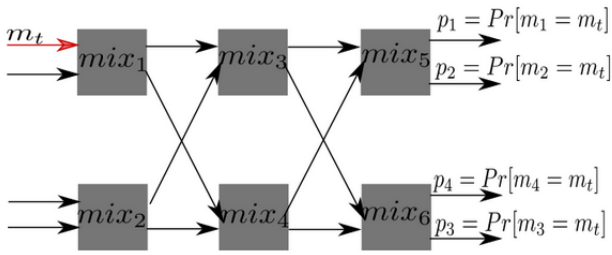


Figure 4: Probability distribution $\Pr_L[m_i = m_t]$ for a target input m_t and all output messages m_i.

non-adversarial mix, as described in Algorithm 1. Messages that go through an adversarial mix do not alter their associated probability, i.e., $\Pr_l[m_i = m_t] = \Pr_{l-1}[m_i = m_t]$ if the mix at layer l is adversarial.

Algorithm 1: Per-mix entropy update for mix at layer $l = 1 \dots L$

Result: Updated $\Pr_l[m_i = m_t]$.
Initialize:
$\Pr_{Mix}[m_t] = 0$;
$\Pr_0[m_i = m_t] = 1$ if $m_i = m_t$;
$\Pr_0[m_i = m_t] = 0$ if $m_i \neq m_t$;
while *Simulation running* **do**
  if *event(receive(m_i))* **then**
    $\Pr_{Mix}[m_t] += \Pr_{l-1}[m_i = m_t]$;
    $pool += 1$;
  end
  if *event(send(m_i))* **then**
    $\Pr_l[m_i = m_t] = \frac{Pr_{Mix}[m_t]}{pool}$;
    $\Pr_{Mix}[m_t] -= \Pr_l[m_i = m_t]$;
    $pool -= 1$;
    Forward Message (m_i)
  end
end

Before entering the first layer, $\Pr_0[m_i = m_t]$ is one for the target input m_t and zero for the rest of the input messages sent by users. $\Pr_{Mix}[m_t]$ denotes the probability that the target is one of the messages in the current internal memory (pool) of the mix and its initial value (before receiving messages that

may be the target) is zero. The *pool* variable simply denotes the number of messages that are currently inside a mix, waiting to be forwarded.

When a message m_i is received by a mix in layer l, its associated probability $\Pr_{l-1}[m_i = m_t]$ is added to $\Pr_{Mix}[m_t]$ to account for the increased probability of m_t being now in that mix. When a message m_i leaves the mix, its updated $\Pr_l[m_i = m_t]$ is a fraction of $\Pr_{Mix}[m_t]$, which is evenly divided by the number of messages currently in the mix's internal pool. This is because in continuous-time mixes with exponential delays all of the messages inside a mix are equally likely to be sent next, and thus the probability that a message in the pool is the target, is uniformly distributed across all messages in the mix at any given time [13]. The outputs of a mix in layer l have an associated updated probability $\Pr_l[m_i = m_t]$ that is the input probability of m_i for the receiving mix in the next layer $l+1$. If the message is being delivered to its final recipient, the probability $\Pr_L[m_i = m_t]$ is the value needed for the entropy calculation. The anonymity of the target message is computed considering the probabilities associated to the mixnet outputs m_i sent to final recipients, as:

$$H = -\sum_i Pr_L[m_i = m_t] * \log_2(Pr_L[m_i = m_t]) \tag{23}$$

We generate message traces using the open source MiXiM discrete event simulator [3] (run on an Intel(R) Core(TM) i9-9920X with 3.50GHz CPU and 132 GB RAM) and compute entropy for hundreds of targets m_t. We treat each target as an anonymity sample and then show average values or full distributions (as boxplots). The number of targets depends on the scenario: for pure network adversaries ($b = 0$) we choose 200 targets, while for adversaries that corrupt a fraction of the mixnet ($b > 0$) we increase to 1000 targets. This is because scenarios with corrupted nodes have outliers for messages going through corrupt nodes that need to be properly sampled.

4 Methodology

Our proposed method for optimizing mixnet design parameters consists of three main steps. First, we set the variables that define the considered adversary (selection of b corrupted fraction or B corrupted nodes) and the deployment scenario (average end-to-end latency D_{e2e} and traffic volume $\lambda_{\mathcal{U}}$). These variables represent external constraints that the system design needs to optimize for. Second, we set a threshold β ($0 < \beta < 1$) that defines the maximum tolerable fraction of compromised routes. Note that $\frac{1}{\beta}$ represents the average number of messages that need to be sent to have one fully compromised route: if we set the worst case threshold at 'one in a thousand' messages then $\beta = 0.001$,

while lowering the tolerance to 'one in a million' messages corresponds to $\beta = 10^{-6}$. Given the range of values for a mixnet design parameter we want to optimize, we compute $\alpha_{\mathcal{F}}$ for each of the values. We then discard parameter values that result in $\alpha_{\mathcal{F}} > \beta$, while keeping those that result in $\alpha_{\mathcal{F}} \leq \beta$ as candidates for the next step. The third and final step computes entropy-based anonymity, in order to find the value that maximizes *average anonymity* in addition to satisfying *worst-case anonymity* constraints.

5 Experimental setup

5.1 Baseline parameters

As part of the first step of the methodology previously outlined, we select baseline values for the adversary and deployment models as follows:

- As baseline, we consider scenarios where adversaries control 10% of the nodes, i.e., where $b = 0.1$. In specific experiments, we also consider scenarios where adversaries do not control any nodes ($b = 0$), scenarios with larger fraction of corrupted nodes ($b = 0.2$ and $b = 0.3$), and scenarios with a constant number of corrupted nodes ($B = 9$, $B = 15$ and $B = 30$).
- In terms of end-to-end latency, we consider as baseline that $D_{e2e} = 1$ second. In specific experiments however we also consider end-to-end latencies of 2, 5, 0.5, and 0.25 seconds.
- In terms of traffic load, the baseline scenario considers $\lambda_{\mathcal{U}} = 5000$ messages per second. When evaluating dummy traffic we also study scenarios where the traffic suddenly drops to just $\lambda_{\mathcal{U}} = 100$ messages per second.

In terms of parameter choices, we argue that $D_{e2e} = 1$ second is a tolerable end-to-end average latency for, e.g., broadcasting transactions to be included in a blockchain or for email applications. In terms of volume, Mastercard processes 5000 transactions per second,[4] which gives a sense of the volume that could be expected in a broadly used payment application if transactions would be routed via a mixnet. Besides these baseline parameters we test other values for comparison (e.g., lower traffic volumes of just 100 m/s, and latencies between 0.25s and 5s). We note that our main contribution is a method that can be used for any latency and volume constraints of interest in concrete deployments, rather than specific results for a specific configuration.

[4] https://applevisaservices.com/blog/faq-how-many-visa-transactions-per-second.html

5.2 Per-mix exponential delay

In multi-hop overlay routing, the end-to-end latency is the aggregation of the latencies incurred at the intermediate hops in the route, each corresponding to a layer in the mixnet. In turn, the latency at each hop is composed of three elements: the network propagation time τ, the packet processing time δ, and the time that the packet dwells in the mix for anonymity purposes, which is sampled from an exponential distribution with mean μ seconds. Given a mixnet with L layers, the message passes by L mix nodes and $L+1$ links, and needs to be processed by L mixes in addition to the final recipient. The average end-to-end latency can be expressed as:

$$D_{e2e} = \mu L + (\tau + \delta)(L + 1) \tag{24}$$

In Section 6.1 we adjust the per-mix latency μ when comparing mixnets with different number of layers L, to fairly compare configurations that provide the same average end-to-end latency D_{e2e}. For this we consider average network propagation and packet processing times $\tau + \delta = 50$ms, and set μ as:

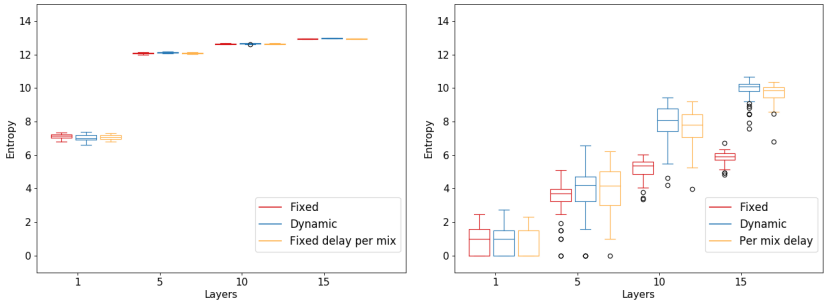
$$\mu = \frac{D_{e2e} - (\tau + \delta)(L + 1)}{L} \tag{25}$$

In the next two sections we study the impact of variable propagation and packet processing times on anonymity calculations.

5.3 Network propagation delay

In practice, the time τ taken by messages to travel through the internet in each hop may be highly variable. Nodes in an overlay network may be located all around the world, and network propagation times are proportional to geographical distance (ultimately bounded by the speed of light and in practice by a fraction of that speed). For example, distances of 500 Km can be covered in just 10ms while intercontinental distances may take over 100ms [34]. Thus, the propagation latency of a route is dependent on the relative geolocations of the nodes in the route. Furthermore, varying transmission medium characteristics, asymmetric and dynamic routing, congestion, and other effects introduce further variance in network propagation latency. Building a model of network propagation latency into a simulator that accurately predicts specific real-world deployment scenarios is a challenging task. Thus, we study the anonymity impact of propagation latency variability by comparing three scenarios with the same average τ: (i) constant propagation latency of $\tau = 50$ms for all links; (ii) variable latency per link sampled from a uniform distribution $\mathcal{U}[10, 90]$ ms;

and (iii) a different propagation delay per mix that is randomly assigned but kept constant for all the received messages throughout the simulation. These three simplified network propagation models provide a sense of the impact of inter-mix propagation variability on anonymity.



(a) Fixed vs variable propagation delay when $\mu = 100$ms

(b) Fixed vs variable propagation delay when $\mu = 0.1$ms

Figure 5: Anonymity with fixed vs variable propagation delays τ.

Our results are shown in Figure 5 for mixnets with different per-mix average latency μ and number of layers L (and thus various D_{e2e} latencies). When μ is larger than the propagation latency τ (Fig. 5a) the average anonymity measured in simulations is the same regardless of whether network propagation times are considered fixed or variable. On the other hand, if μ is orders of magnitude smaller than τ (Fig. 5b), the variation of τ has an anonymity impact that makes message tracing harder for an adversary, and this impact is exacerbated with the number of layers in the mixnet. The main effect leading to this anonymity increase when considering variable propagation times whether changing per message or just per mix, is that more output messages are possible matches for a target input, since the window of likely output matches starts earlier (the message could have been lucky to travel via links with low propagation delay) and ends later (the message could have been unlucky and travel via links with a lot of delay).

Based on these results, we conclude that considering constant propagation delays is a conservative assumption that seems to provide a lower bound on anonymity. Considering variable τ risks overestimating anonymity if the modelled variance is larger than the actual variance present in a concrete real-world mixnet deployment.

5.4 Non-uniform mix capacities

So far we have assumed 'uniform routing', i.e., that routing choices per layer are uniform in the number of nodes W in the layer, spreading the traffic load equally over all mix nodes in the network. In this section we consider networks with 'biased (capacity-based) routing', i.e., that allow for different node capacities and that select nodes for a route proportionally to the share of capacity that each node contributes to its mixnet layer. Capacity-based routing has two advantages: first, it is more inclusive, as even participants with limited resources can contribute to the network; and second, it better utilizes available resources, as some mix nodes are able to process more packets than others, and their extra capacity is wasted with uniform routing.

We compare anonymity for both types of routing (uniform and biased) in a small mixnet of $N = 30$ nodes organized as a Wx$L = 10$x3 network, considering the baseline parameters provided in Section 5.1: $\lambda_{\mathcal{U}} = 5000$ messages per second, $D_{e2e} = 1$ second, and $b = 0.1$ fraction of corrupted nodes, i.e., $B = 3$ adversarial nodes.

Regardless of the type of routing, from a worst-case anonymity perspective the adversary compromises zero routes whenever no adversarial nodes are present in a layer. In this example $\alpha_{\mathcal{F}} = 0$ whenever the network topology is different from the optimal adversarial topology $\mathcal{T}_{opt} = (\mathcal{A}, \mathcal{B})$ that corresponds to $\mathcal{A} = \{9, 9, 9\}$ and $\mathcal{B} = \{1, 1, 1\}$. We thus focus our comparison on topologies $\mathcal{T}_{opt}$.

Next, we observe that adversaries can be expected to introduce high-capacity nodes in order to maximize route captures ($\alpha_{\mathcal{F}}$). Introducing many nodes in a staking-based system such as the Nym network can be very costly, as the adversary may need to spend millions of dollars to acquire enough stake to control a high percentage of nodes, or otherwise build enough reputation to persuade other stakeholders to delegate millions to support adversarial nodes [21]. In contrast, the additional cost of computing and bandwidth resources that provide significantly larger-than-average node capacity is in the range of thousands of dollars, orders of magnitude less and within the budget of a broader set of adversaries.

We consider that the adversary introduces nodes with 6x more capacity than the average honest node. Thus, in each layer of $W = 10$ nodes the adversarial node has 40% of the layer's capacity and is thus chosen for 40% of the routes. This is in contrast to the uniform routing scenario where each node, including the adversary's, routes 10% of the messages. Using Eq. (7), we can see that in this example biased routing allows the adversary to fully compromise $\alpha_F = 6.4\%$ of routes, in contrast to $\alpha_F = 0.1\%$ of routes in the case of uniform routing; i.e., a 6-fold increase in adversarial bandwidth and computing resources yields a 64-fold increase in the rate of worst-case compromise.

As final step we examine the effect of uniform vs biased routing on average anonymity, and show the results in Figure 6. The red bloxplots show the entropy distribution when considering a network adversary that does not control any mixnet nodes. In this case both uniform and biased routing provide the same level of average anonymity. The blue boxplots show results when the mixnet contains three adversarial nodes, which route 10% of messages per layer in the uniform case and 40% in the biased case. In this case we can see that compared to uniform routing, biased routing enables the adversary to not only compromise many more routes (worst-case anonymity) but also diminish average anonymity for the remaining messages. Based on these results we conclude that uniform routing is the best choice from an anonymity perspective and consider uniform routing policies in our remaining experiments. We note that volunteer-based networks like Tor [29] benefit from flexibility as that allows everyone to contribute even if they have limited capacity – and thus enforcing uniform routing in such networks comes with the cost of excluding prospective node operators with capacity limitations. In commercial networks like Nym [21] however, nodes are rewarded for operating the network, and it is thus possible to set a minimum capacity requirements and penalize with lower rewards the nodes that fail to perform.

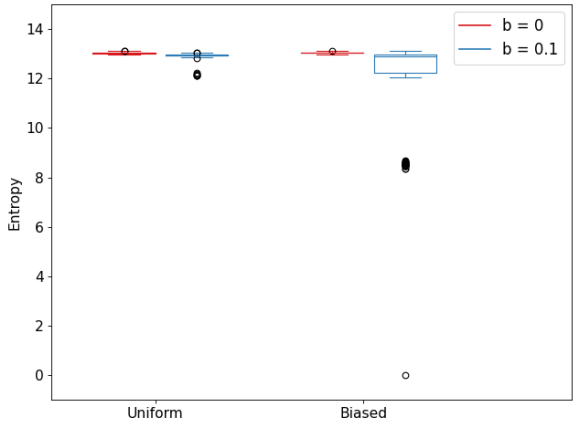


Figure 6: Anonymity with uniform vs biased routing considering no adversarial nodes (red) and 10% of adversarial nodes (blue).

6 Optimization results

6.1 Optimizing the number of layers L

We first apply the methodology outlined in Section 4 to address the question: *given a deployment scenario and adversary model, what is the optimal number L of mixnet layers?* We consider a mixnet that routes $\lambda_{\mathcal{U}} = 5000$ messages per second and that has a width of $W = 10$ mixes per layer, meaning that each mix routes on average 500 messages per second.[5] We consider that the average end-to-end latency D_{e2e} is fixed per optimization experiment and evaluate anonymity for a range of possible D_{e2e} values, from 0.25 to 5 seconds. We first consider a global network adversary that can observe all links but does not control any mix nodes, i.e., $B = b = 0$. Next, we consider an adversary that, in addition to globally observing the network, also controls $b = 10\%$ of nodes in the mixnet.

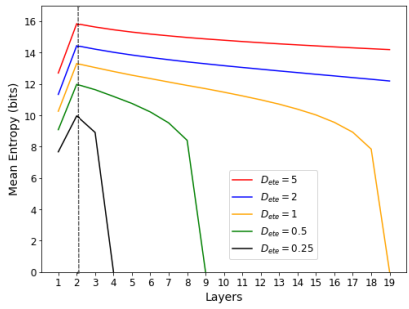
6.1.1 Global network adversary

In the case of adversaries that do not control any nodes in the mixnet ($B = 0$), the fraction of fully compromised routes $\alpha_{\mathcal{F}}$ is zero for any number of mixnet layers $L \geq 1$. Worst-case anonymity constraints are therefore satisfied for all possible values of β.

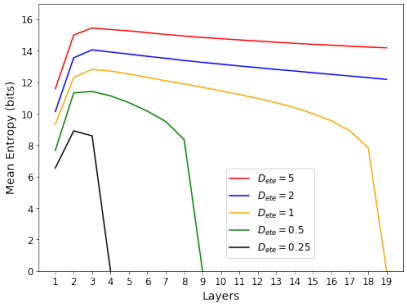
Next we turn to examining average anonymity. Figure 7a shows the mean entropy as a function of the number of layers L for different values of D_{e2e}. As we can see in the results, when $B = 0$ the optimal number of layers is $L = 2$ for all values of the end-to-end latency D_{e2e}. As expected, anonymity values are higher for higher D_{e2e} [19]. Note that for $L = 1$, messages are partitioned in W subsets with disjoint anonymity sets (similarly to how they would be in parallel cascades), and thus the anonymity of $L = 1$ is naturally inferior to $L = 2$, which aggregates all messages in one large anonymity set. This effect would be further exacerbated with higher W, as W increases the partitioning.

Thus, assuming that all mix nodes are honest, a second mixnet layer achieves the best possible mixing for any end-to-end latency. Adding more layers implies wasting more time in propagation between layers, and leaving less latency budget for delaying messages inside the nodes (and thus *mixing* them in bigger pools). Entropy drops to zero and messages are fully distinguishable when all the latency is wasted on propagation and mixes simply forward messages without adding any random latency to reorder them. Considering a propagation

[5] The currently available Nym implementation is benchmarked at 3125 Sphinx packet decryptions per second per processing core. An average node load of 500 messages per second enables mix nodes to tolerate traffic peaks of up to 6x the average load.



(a) Global network adversary, $b = 0$, $B = 0$



(b) Corruption level $b = 0.1$, $B = bLW$

Figure 7: Mean Entropy wrt number of layers L for various D_{e2e}, considering $\lambda_{\mathcal{U}} = 5000$ and mixnet width $W = 10$.

latency per link of 50ms, this happens at $L = 4$ for $D_{e2e} = 0.25$s, $L = 9$ for $D_{e2e} = 0.5$s, and $L = 19$ for $D_{e2e} = 1$s. When entropy drops to zero this means that the adversary can identify which output message corresponds to a target input and the system provides no anonymity – though recall that, as shown in Section 5.3, variations of propagation time may make message tracing more uncertain in practical scenarios.

6.1.2 Fraction of corrupted nodes

We now examine scenarios where, *in addition to* observing all links, the adversary corrupts a fraction $b = 0.1$ of the nodes.

First, from a worst-case perspective the fraction of fully compromised messages in a mixnet with L layers can be approximated as $\alpha_{\mathcal{F}} = b^L$ (Eq. (7)). Considering $b = 0.1$, a worst-case threshold $\beta = 0.001$ (one in a thousand messages is compromised by adversaries controlling 10% of the mixnet) implies that the minimum number of mixnet layers is $L = 3$. Setting a requirement of $\beta = 10^{-6}$ (just one in a million messages is compromised) raises the minimum number of layers to $L = 6$.

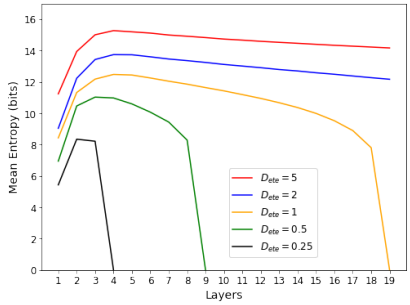
We then evaluate average anonymity for various end-to-end latencies D_{e2e} and layers L. Figure 7b shows the results, where we can observe that from an average anonymity perspective, the optimal number of layers L is dependent on the end-to-end latency D_{e2e}. For the more relaxed latency constraints $D_{e2e} \geq 1$s (yellow, blue and red lines), the optimal L is now $L = 3$; while scenarios with stricter latency constraints $D_{e2e} < 1$s (green and black lines) have their maximum at $L = 2$.

For $D_{e2e} \geq 1$s, the new optimum at $L = 3$ instead of $L = 2$ (as in the scenario without corrupted nodes) is consistent with the worst-case effect of node corruption, which brings anonymity to zero for a fraction of samples, and is mitigated by increasing L. Adding layers not only exponentially reduces the number of cases where anonymity is zero due to full route compromise, but also the number of outlier cases where anonymity is very low due to messages passing through a single honest mix. Beyond a certain point however, adding layers is more detrimental than beneficial, since the fraction of fully compromised routes is already too negligible for any further reduction to make a difference in the average, while the smaller mixing time (due to adding layers) takes a toll on anonymity.

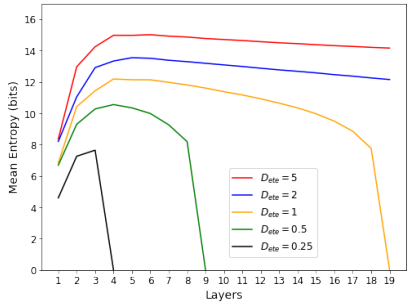
When the latency D_{e2e} is more constrained, the optimal L still maxes out at a lower $L = 2$. This is explainable because for $D_{e2e} = 0.25$s, $L = 3$ implies that 80% of the available end-to-end latency budget is wasted on propagation across four links, leaving less than 17ms for mixing at each of the three nodes in the route. The fact that little mixing takes place per node facilitates message tracking for network adversaries and makes this configuration offer worse average anonymity than $L = 2$, where only 60% of latency budget is spent on propagation leaving 50ms for mixing at each of the two nodes.

For our optimization we consider $\beta = 0.001$ for adversaries that control a fraction $b = 0.1$ of the mixnet, and thus we discard mixnet configurations with $L < 3$. We select $L = 3$ and consider this number of mixnet layers in the remaining experiments. We note that $L = 3$ is commonly used in deployed anonymity networks [21, 26] as well as default experimental setting in prior literature [22, 45]. We are however the first to show that $L = 3$ is the choice that optimizes anonymity for layered mixnets in conditions of moderate rate of compromise ($b = 10\%$ and $\beta = 0.001$) for end-to-end latency tolerances of up to five seconds.

Lowering the worst-case threshold to $\beta = 10^{-6}$ while considering $b = 0.1$ sets the minimum number of layers at $L = 6$. In networks with end-to-end latency D_{e2e} of half a second or more this sets the optimal number of layers at $L = 6$ (since this offers better average anonymity than networks with $L > 6$). For networks with $D_{e2e} = 0.25$s there is no solution that can meet both the latency ($D_{e2e} = 0.25$s) and anonymity ($\beta = 10^{-6}$) requirements. We also note that in practical terms, adding layers to a mixnet comes with significant costs, as it requires additional resources per message (servers, computation and bandwidth) as well as incurring in increased rates of message loss (since it is enough for one node in the route to fail for the message to be lost).



(a) Average entropy for $b = 0.2$



(b) Average entropy for $b = 0.3$

Figure 8: Mean Entropy wrt number of layers L for various D_{e2e}, with $W = 10$ and $\lambda_{\mathcal{U}} = 5000$.

6.1.3 Higher fraction of corrupted nodes

We examine results for selecting the optimal number of layers L with a high corruption rate b. In terms of worst-case anonymity, the same threshold $\beta = 0.001$ imposes a higher L than in the cases with lower b shown in Section 6.1: while $L \geq 1$ was enough for $b = 0$ and $L \geq 3$ for $b = 0.1$, $b = 0.2$ raises the minimum required layers to $L \geq 5$ and $b = 0.3$ to $L \geq 6$. Increasing the worst-case tolerance to $\beta = 0.01$ (on average one message out of 100 has a fully compromised route) allows for configurations where $L \geq 3$ for $b = 0.2$ and $L \geq 4$ for $b = 0.3$.

We then examine average anonymity in these scenarios, showing the results for $b = 0.2$ and $b = 0.3$ in Figure 8. As we can see in the figure, higher corruption rates b increase the optimal L for a given end-to-end latency D_{e2e}. For example, for $D_{e2e} = 5$s and $D_{e2e} = 2$s, the optimal L increases from $L = 4$ for $b = 0.2$ to $L = 5$ for $b = 0.3$; while, as shown in Figure 7, $b = 0$ had the optimum at $L = 2$ and $b = 0.1$ at $L = 3$. Increasing the number of mixnet layers beyond the optimal L makes the average anonymity, up to the point where it drops to zero because all available latency budget is spent on propagation delays. For the more constraining latency $D_{e2e} = 0.25$s, increasing the corruption rate to $b = 0.3$ makes $L = 3$ become the optimum instead of $L = 2$, which is the optimum for lower rates of corruption.

Combining worst-case anonymity constraints and average anonymity optima for the different scenarios, we conclude that for $\beta = 0.001$ the minimum layers required by the worst-case dominate, determining that the number of layers should be $L = 5$ for $b = 0.2$ and $L = 6$ for $b = 0.3$. This choice is the same for all $D_{e2e} \geq 0.5$s, while no solution exists for $D_{e2e} = 0.25$s that can satisfy

both anonymity and latency constraints. Considering a more relaxed worst-case anonymity constraint $\beta = 0.01$ for $b = 0.2$ would lead to selecting the L that maximizes average anonymity, which is $L = 3$ for $D_{e2e} \leq 0.5$s and $L = 4$ for larger D_{e2e}. In the case of $b = 0.3$, the choices would be $L = 4$ for $D_{e2e} = 0.5$s and $D_{e2e} = 1$s, and $L = 5$ for larger D_{e2e}; while again no solution exists for $D_{e2e} = 0.25$s that satisfies both worst-case anonymity and latency constraints.

6.2 Optimizing the network width W

Once we have fixed the number of layers L, we proceed to the next question: *how does the width W of the mixnet impact anonymity?* We consider a mean end-to-end latency $D_{e2e} = 1$ second, mixnets with $L = 3$ layers, and a worst-case compromise threshold $\beta = 0.001$. As before, we consider $\lambda_{\mathcal{U}} = 5000$ messages per second and a minimum width $W_{min} = 10$ nodes, each routing 500 messages per second on average. We consider three threat models: network adversaries that do not control any nodes ($b = B = 0$), adversaries that control a fraction $b = 0.1$ of nodes regardless of network size, and adversaries that control a fixed number of nodes $B = \{9, 15, 30\}$ regardless of network size.

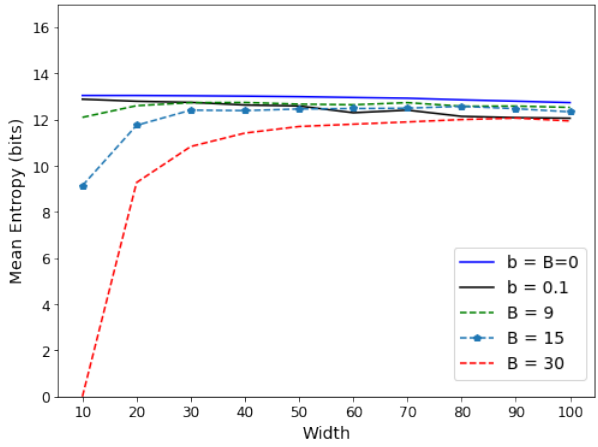


Figure 9: Mean entropy as a function of the mixnet width W for various b and B ($L = 3$, $\lambda_{\mathcal{U}} = 5000$, $D_{e2e} = 1$s).

6.2.1 Constant fraction of corrupted nodes

We first consider adversaries that control a constant fraction b of the total nodes. This means that as the network grows in width W, the number of adversarial nodes B increases proportionally to W, as $B = bLW$.

In terms of worst-case anonymity, note that the fraction $\alpha_{\mathcal{F}}$ of fully compromised routes remains constant as W grows, because the probability of selecting fully corrupted routes is given by $\alpha_{\mathcal{F}} = b^L$ and thus remains constant. For $b = 0.1$ and $L = 3$ this corresponds to $\alpha_{\mathcal{F}} = 0.001$, which matches the worst-case threshold β. For $b = 0$, all $L \geq 1$ meet any possible value for the β threshold.

Figure 9 shows average anonymity for various scenarios, with the blue and black solid lines representing the cases where $b = 0$ and $b = 0.1$, respectively. As we can see in the figure, anonymity slowly but steadily decreases as the network width W increases. For $b = 0.1$, anonymity decreases by one bit when the width is $W = 100$ compared to the minimum $W_{min} = 10$, meaning that anonymity sets are halved due to the 10-fold width increase. This (modest) decrease happens because higher width means *thinner traffic* per mix, and thus lower level of mixing at each node. This result indicates that when considering a constant fraction adversary, the optimal network width is the minimum W that is sufficient to route the required traffic volume.

6.2.2 Constant number of corrupted nodes

Next we consider adversaries that can corrupt a fixed number B of nodes. In this case, the fraction $b = \frac{B}{LW}$ of adversarial nodes diminishes when the network grows in width W. Therefore, from a worst-case perspective, increasing W can be a strategy to meet constraints on worst-case rates. For example, considering $\beta = 0.001$ and $B = 15$, a mixnet of $L = 3$ with the minimum width $W_{min} = 10$ fails to meet worst-case constraints, as $\alpha_{\mathcal{F}} = (\frac{15}{30})^3 = 0.125$ is orders of magnitude larger than the threshold β. The width W that satisfies β constraints is given by $W = \frac{B}{L\sqrt[L]{\beta}}$. Applying this analysis to $B = 9$, $B = 15$ and $B = 30$ malicious nodes, results in minimum widths of $W = 10$, $W = 50$ and $W = 100$, respectively.

We then evaluate average anonymity in the same scenarios as a function of mixnet width. The results are shown with dashed lines in Figure 9 for 3 corrupted nodes per layer ($B = 9$), 5 corrupted nodes per layer ($B = 15$) and 10 corrupted nodes per layer ($B = 30$). Note that $B = 30$ at width $W = 10$ is a corner case where the *entire* network is controlled by the adversary, and thus all messages are fully traceable and average anonymity is zero. Compared to the previous adversary defined by $b = 0.1$, anonymity levels are lower for small W because relative corruption levels are higher. Overall, in scenarios with constant levels of node

corruption we see that increasing W is initially beneficial for anonymity, as the diminishing share b of corrupted nodes dominates an anonymity improvement. After some point however, further increasing W begins to lower anonymity, as the dominant factor becomes the overall thinning of traffic (exploitable by a network adversary) rather than the fraction of compromised routes (which has already reached negligible levels). Based on these results we choose $W = 50$ for our remaining experiments.

6.2.3 Higher fraction of corrupted nodes

We now examine the effects of mixnet width W with increasing percentages b of adversarial node corruption. As mentioned in Section 6.2, a fixed b and L determine the worst-case anonymity rate as $\alpha_{\mathcal{F}} = b^L$, regardless of the mixnet width W. We thus examine average anonymity in scenarios with higher values of b, and show the results in Figure 10. As expected, an increased b lowers the average anonymity for any width W, and the decline of average anonymity caused by traffic thinning with larger W is slightly faster for higher b. Overall, the decline in average anonymity is noticeable but moderate: between one and two bits of decline when the mixnet width W is increased by an order of magnitude from $W = 10$ to $W = 100$.

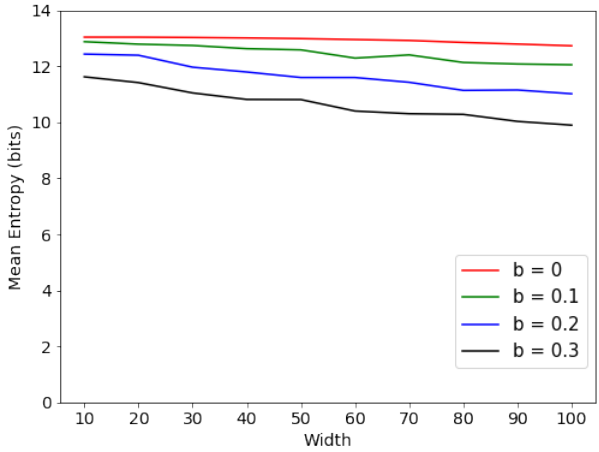


Figure 10: Mean entropy as a function of the mixnet width W for various levels of corruption b ($L = 3$, $\lambda_{\mathcal{U}} = 5000$, $D_{e2e} = 1$s).

6.3 Mix-based dummy strategies to compensate for low traffic volume

We finally turn our attention to the question of what happens when a network that has been optimized for an average traffic volume suddenly sees input traffic drop by more than an order of magnitude. A steep drop in traffic rate $\lambda_{\mathcal{U}}$, particularly in networks with a large width W, can significantly reduce average anonymity levels due to *thin traffic* effects (note that $\alpha_{\mathcal{F}}$ is independent of overall traffic volume, and thus worst-case anonymity is unaffected by traffic load fluctuations). Mixnet parameter adjustments that can counter the drop in user traffic include increasing the end-to-end latency D_{e2e} and reducing the network width W. Increasing latency however may not be possible without seriously undermining the usability of the system. As for network resizing, typically information on active mixnet nodes is updated every hour or few hours [21, 26], and thus structural changes to the network width may not be possible to effect immediately, or fast enough to follow fluctuations in the traffic volume coming from users.

Designs such as Loopix [45] and Vuvuzela [31] rely on client-based dummy traffic to ensure that traffic volumes are sustained and provide an adequate level of anonymity. Client-based dummies are a very effective solution to users going idle while staying online. If end users go offline however, all their traffic ceases, and it becomes unreasonable to expect that they will continue to generate dummy traffic.

Upon detection of low traffic volumes, mix nodes may intervene by generating an increased volume of dummy traffic to support anonymity levels. We note that various prior works propose mix-based dummies with more or less sophisticated strategies for generation and routing [17, 31, 45]. Here we consider two very simple strategies introduced in Section 2.1.4: *link-based dummies* and *partial-route dummies*.

Link-based dummies are generated by a mix and discarded by the successor. Assuming that mixes in adjacent layers communicate via a link-encrypted connection (e.g., TLS), link-based dummies need not be actual Sphinx packets that require expensive public key operations, but simply random data blocks the size of a Sphinx message that can be detected and discarded by the receiving mix node with just symmetric key operations. Link-based dummies are thus very cheap to implement for mixes, which makes them a low-cost countermeasure to use on-demand in case of decreased user traffic. On the downside, link-based dummies only protect towards network adversaries. If any of the two nodes sharing a link is compromised, the adversary can trivially filter out all the link-based dummy messages, rendering the protection ineffective for that link.

Partial-route dummies are generated by mixes in all but the last layer, routed through the mixnet and discarded by mixes in the last layer. In a network of $L = 3$ layers, compared to link-based dummies, partial-route dummies increase protection against adversarial nodes in the middle layer, who can no longer distinguish user messages from dummy messages generated and discarded by honest mixes. Note that this sort of indistinguishability towards middle-layer nodes requires that dummies are encoded as Sphinx packets, which significantly increases the cost of the dummy strategy compared to link-based dummies, as the processing of a Sphinx message requires senders, receivers and intermediaries to perform expensive public key operations.

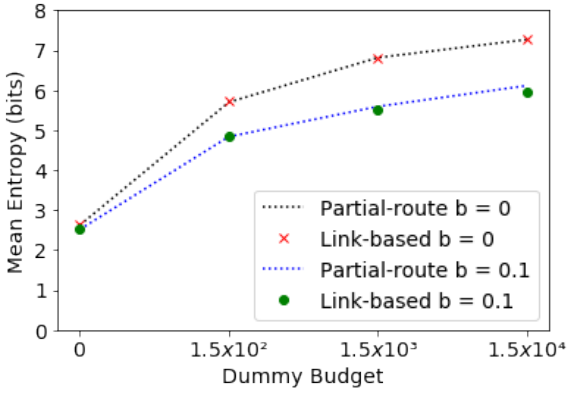


Figure 11: Average anonymity in low-traffic conditions ($\lambda_{\mathcal{U}} = 100$ m/s) with link-based and partial-route dummy strategies towards network adversaries corrupting $b = 0$ and $b = 0.1$ of a mixnet with $L = 3$, $W = 50$, and $D_{e2e} = 1$s.

Contrary to link-based dummies, partial-route dummies are not evenly distributed across layers but instead increase linearly with the layers, because the dummies generated by each layer of mixes are added to the dummies from earlier layers being forwarded, until they are all discarded by the last layer. In order to enable a fair comparison between both dummy strategies we compare scenarios that have the same overall dummy rate. Considering $L = 3$, a network where each mix generates $\lambda_{\mathcal{M}} = 1$ partial-route dummy message per second is equivalent in terms of dummy traffic volume to a network where each mix generates $\lambda_{\mathcal{M}} = 1.5$ link-dummies per second. In the former case, one third of the dummies is sent from the first to the second layer and two thirds are sent from the second to the third layer, where they are discarded. In the latter case, half the dummies are sent from the first to the second layer and the other half from the second to the third. No dummies are generated by the last layer in

either case. We compare scenarios considering the same network-wide level of dummy traffic.

Our evaluation of both dummy strategies is shown in Figure 11 for various dummy rates, considering both network adversaries ($b = 0$) and adversaries that corrupt 10% of the mixnet ($b = 0.1$). Section 6.4 includes additional results for adversaries with higher corruption rates ($b = 0.2$ and $b = 0.3$). The x axis in the figure corresponds to the overall dummy traffic in the network. Thus, $x = 1.5\text{x}10^2$ corresponds to $\lambda_{\mathcal{M}} = 1$ *partial-route dummy* generated by each mix per second. Given that $W = 50$, this implies that there are 50 dummies per second between the first and second layers, and 100 dummies per second between the second and third layers, for a total of 150. This is compared to a rate of $\lambda_{\mathcal{M}} = 1.5$ *link-based dummies* per second, with 75 dummies per second in each of the layers adding to the same 150 total. At $x = 1.5\text{x}10^4$ mixes generate $\lambda_{\mathcal{M}} = 100$ *partial-route dummies* or $\lambda_{\mathcal{M}} = 150$ *link-based dummies* per second. Considering $\lambda_{\mathcal{U}} = 100$ messages per second, dummy traffic makes up 43% of inter-mix traffic for $x = 1.5\text{x}10^2$, 88% for $x = 1.5\text{x}10^3$, and 98.6% for $x = 1.5\text{x}10^4$.

As we can see in the figure, in the absence of dummy traffic ($\lambda_{\mathcal{M}} = 0$) anonymity in this network configuration ($N = 3\text{x}50$) is very low due to the low levels of user traffic. Even where there is no corruption ($b = 0$), average anonymity is below 3 bits of entropy, meaning that effective anonymity set sizes are just a handful of messages. Anonymity levels significantly improve once mixes generate dummy traffic, with diminishing returns as the dummy rate increases and anonymity levels approach their upper bound. The best improvement is in the case of $b = 0$, where anonymity goes up by 5 bits, meaning that the anonymity set size multiplies 32-fold thanks to the dummy traffic. Compared to a mixnet that has the same three layers and minimal width ($W = 1$), we find that the average anonymity is the same for $W = 1$ (with no dummies) and for $W = 50$ with $\lambda_{\mathcal{M}} = 1.5\text{x}10^4$, meaning that a high rate of mix-generated dummy traffic succeeds in fully making up for the loss of anonymity caused by traffic thinning.

Link-based and partial-route dummies provide the same protection when there is no adversarial corruption ($b = 0$). This is to be expected since partial-route dummies offer extra protection towards intermediate corrupted nodes, but the same protection as link dummies against external network adversaries. The fact that dummies are distributed across layers $33\% - 66\%$ for partial-route dummies and $50\% - 50\%$ for link-based dummies seems to make no difference to the effectiveness of the dummy strategy. Once a fraction of nodes is compromised ($b = 0.1$), the gains obtained from dummy traffic are mitigated. A the lower levels of $\lambda_{\mathcal{M}}$ dummy traffic still significantly improves anonymity compared to not generating any dummies at all, and both link-based and partial-route

strategies provide similar protection. As the dummy rate $\lambda_{\mathcal{M}}$ increases, partial-route dummies provide slightly better anonymity than link-based dummies.

This effect becomes more pronounced with higher corruption rates.

6.4 Effectiveness of dummy strategies

Finally, we examine the effectiveness of partial-route and link-based dummy strategies when there is very low traffic from users ($\lambda_{\mathcal{U}} = 100$m/s) in a mixnet dimensioned for higher traffic loads (width $W = 50$), in the presence of adversaries that corrupt $b = 0.2$ and $b = 0.3$ of the mixnet. Our results, shown in Figure 12, illustrate that high levels of mixnet corruption diminish the effect of dummies (compared to the results for lower b shown in Figure 11) and in particular of link-based dummies — many of which are now identified and discarded by the large number of adversarial nodes. When $b = 0.3$, even high levels of link-based dummies result in anonymity below 4 bits; while for $b = 0.2$ the anonymity set triples (1.4 bit increase), reaching an average entropy of 5 bits.

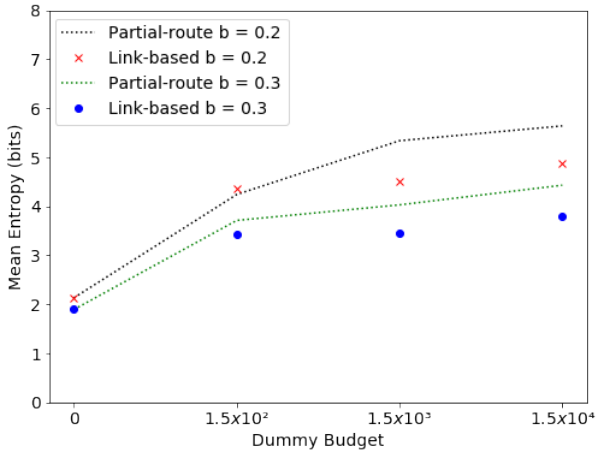


Figure 12: Average anonymity in low-traffic conditions $\lambda_{\mathcal{U}} = 100$ m/s with link-based and partial-route dummy strategies towards adversaries corrupting $b = 0.2$ and $b = 0.3$ of a mixnet with $L = 3$, $W = 50$, and $D_{e2e} = 1$s.

Partial-link dummies fare moderately better in this challenging adversarial scenario with low traffic and high level of compromise. When $b = 0.2$, partial-route dummies can increase anonymity up to 6 bits, a 16-fold increase in anonymity set size compared to the 2 bits obtained when not using dummies.

In the case of $b = 0.3$ however, even high levels of partial-route dummies result in a mean entropy of 4.5 bits, corresponding to a perfect indistinguishability set of about twenty other users, which may be too small to provide meaningful protection.

Given the huge difference in cost of the two considered dummy strategies and their comparable impact, we conclude that link-based dummies are a simple and low-cost, yet effective option for mixes to support anonymity levels when there are sudden dips in user traffic.

7 Related work

Since Chaum's seminal work on untraceable email in 1981 [7], there has been a great amount of research related to mixnets, both in design [8, 9, 24, 31, 32, 35, 36, 37, 43, 45] as in evaluation and optimization [11, 13, 23, 39, 41, 44, 47, 51, 52]. We highlight in this section the most relevant prior work in terms of mixnet parameter optimization.

Rebollo-Monedero et al. [47] provide a method for optimizing the threshold and pool parameters of individual batch-based mixes. Their optimization problem is similar to ours: given a traffic volume and latency constraint, what are the optimal parameters that maximize entropy-based anonymity? Their anonymity system model is however vastly simpler than ours: where we consider full mixnets that may be partially compromised, they restrict themselves to a single (trusted) mix. In terms of optimization methods, their simpler model allows for multiobjective optimization of entropy-based anonymity, while we have to resort to empirical analysis to compare configurations and find optimal parameters that maximize average (entropy-based) anonymity while meeting worst-case anonymity constraints. In another result on mixing algorithm optimization but this time concerning individual continuous-time mixes [32], Danezis [13] showed that for a given mean latency exponentially-distributed delays provide optimal anonymity, thanks to the memory-less properties of the exponential distribution. Prior results on mixnet topology optimization [22] are taken into consideration in our choice of focusing on layered networks. To the best of our knowledge, our work is the first to tackle mixnet size optimization.

Proposed systems that are reliant on dummy traffic, such as Loopix [45] and Vuvuzela [31], leave the tuning of parameters for the dummy traffic strategies as out of scope. In terms of dummy traffic optimization, Oya et al. [39] consider long-term disclosure attacks [18], which exploit persistent communication patterns to infer communication profiles over time, and propose dummy traffic strategies for networks of pool mixes. Their method, based on solving a least squares problem, optimizes the amount of dummies needed to achieve a desired

level of protection against these long-term disclosure attacks. Our model does not make assumptions about repeated user behaviour, focusing instead on the anonymity offered by the mixnet to individual messages. Given specified models for user recipient selection, the methods of Oya et al. may be applied to the mixnet configuration resulting from our methods to further mitigate long-term attacks.

Finally, a first version of the MiXiM simulator that we use in our evaluations was first presented in [3]. The contribution in [3] is the evaluation of different mixing strategies and network connectivity topologies, which shows that Poisson mixing and stratified topologies provide better anonymity than pool mixing and topologies such as XRD [35]. We build on those results by considering the strategy and topology identified as providing the best anonymity properties, and proceeding to parameter optimization within the resulting design space.

References

[1] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, WPES '07, page 11–20, New York, NY, USA, 2007. Association for Computing Machinery.

[2] Iness Ben Guirat, Devashish Gosain, and Claudia Diaz. MiXiM: A general purpose simulator for mixnet. *13th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2020)*, 2020.

[3] Iness Ben Guirat, Devashish Gosain, and Claudia Diaz. MiXiM: Mixnet Design Decisions and Empirical Evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, 2021.

[4] Rainer Bohme, George Danezis, Claudia Diaz, Stefan Kopsell, and Andreas Pfitzmann. Mix Cascades vs. Peer-to-Peer: Is One Concept Superior? In *Proceedings of Privacy Enhancing Technologies, PET 2004*, volume 3424 of *LNCS*, pages 243–255. Springer-Verlag, 2004.

[5] Dan Boneh and Philippe Golle. Almost Entirely Correct Mixing With Application to Voting. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, pages 68–77, November 2002.

[6] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? In *Proceedings of the 14th ACM conference on Computer and Communications Security Security (CCS 2017)*, pages 92–102. ACM, 2007.

[7] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[8] David Chaum, Farid Javani, Aniket Kate, Anna Krasnova, Joeri Ruiter, Alan T Sherman, and Debajyoti Das. cMix: Anonymization by high-performance scalable mixing. Technical report, 2016.

[9] Chen Chen, Daniele Enrico Asoni, Adrian Perrig, David Barrera, George Danezis, and Carmela Troncoso. TARANET: traffic-analysis resistant anonymity at the network layer. *CoRR*, abs/1802.08415, 2018.

[10] Lance Cottrell. Mixmaster and remailer attacks, 1995.

[11] George Danezis. Mix-networks with restricted routes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, pages 1–17. Springer-Verlag, LNCS 2760, March 2003.

[12] George Danezis. *Better Anonymous Communications*. PhD dissertation, University of Cambridge, Computer Laboratory Queens' College, January 2004.

[13] George Danezis. The traffic analysis of continuous-time mixes. In *International Workshop on Privacy Enhancing Technologies*, pages 35–50, Berlin, Heidelberg, 2004. Springer.

[14] George Danezis and Richard Clayton. Introducing traffic analysis. In *Digital Privacy: Theory, Technologies, and Practices (1st ed.)*, page 22. Auerbach Publications, 2007.

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

[16] George Danezis and Ian Goldberg. Sphinx: A Compact and Provably Secure Mix Format. In *2009 30th IEEE Symposium on Security and Privacy*, pages 269–282, 2009.

[17] George Danezis and Len Sassaman. Heartbeat traffic to counter (n-1) attacks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, October 2003.

[18] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *International Workshop on Information Hiding*, pages 293–308. Springer, 2004.

[19] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity Trilemma: Strong anonymity, Low Bandwidth Overhead, Low latency - Choose two. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 108–126, 2018.

[20] Claudia Diaz. *Anonymity and Privacy in Electronic Services*. PhD dissertation, KU leuven, December 2005.

[21] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym Network. `https://nymtech.net/nym-whitepaper.pdf`, February 2021.

[22] Claudia Diaz, Steven J. Murdoch, and Carmela Troncoso. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 184–201, Berlin, Heidelberg, 2010. Springer-Verlag.

[23] Claudia Diaz, Len Sassaman, and Evelyne Dewitte. Comparison between two practical mix designs. In *Proceedings of ESORICS 2004*, LNCS, September 2004.

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

[25] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards Measuring Anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 54–68, Berlin, Heidelberg, 2002. Springer-Verlag.

[26] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, page 21, USA, 2004. USENIX Association.

[27] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, pages 186–206, May 2004.

[28] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[29] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In *Proceedings of the First International Workshop on Information Hiding*, page 137–150, Berlin, Heidelberg, 1996. Springer-Verlag.

[30] Marcin Gomulkiewicz, Marek Klonowski, and Miroslaw Kutylowski. Rapid mixing and security of chaum's visual electronic voting. In *Proceedings of ESORICS 2003*, October 2003.

[31] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[32] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop- and- Go-MIXes providing probabilistic anonymity in an open system. In David Aucsmith, editor, *Information Hiding*, pages 83–98, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[33] Joe Kilian and Kazue Sako. Receipt-free MIX-type voting scheme - a practical solution to the implementation of a voting booth. In *Proceedings of EUROCRYPT 1995*. Springer-Verlag, May 1995.

[34] Katharina Kohls and Claudia Diaz. {VerLoc}: Verifiable Localization in Decentralized Systems. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2637–2654, 2022.

[35] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable Messaging System with Cryptographic Privacy. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 759–776, Santa Clara, CA, February 2020. USENIX Association.

[36] Nick Mathewson and Roger Dingledine. Mixminion: Strong anonymity for financial cryptography. In *Proceedings of Financial Cryptography (FC '04)*, pages 227–232. Springer-Verlag, LNCS 3110, February 2004.

[37] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. IETF Internet Draft, July 2003.

[38] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, 2001.

[39] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. Do dummies pay off? limits of dummy traffic protection in anonymous communications. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 204–223. Springer, 2014.

[40] Sameer Parekh. Prospects for remailers. *First Monday*, 1, August 1996.

[41] Fernando Pérez-González, Carmela Troncoso, and Simon Oya. A least squares approach to the static traffic analysis of high-latency anonymous communication systems. *IEEE Trans. Inf. Forensics Secur.*, 9(9):1341–1355, 2014.

[42] Andreas Pfitzmann and Marit Hansen. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In *Designing privacy enhancing technologies*, pages 1–9. Springer, 2001.

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

[44] Ania M. Piotrowska. An empirical study of privacy, scalability, and latency of nym mixnet, 2 2021.

[45] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix Anonymity System. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1199–1216, Vancouver, BC, August 2017. USENIX Association.

[46] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[47] David Rebollo-Monedero, Javier Parra-Arnau, Jordi Forné, and Claudia Diaz. Optimizing the design parameters of threshold pool mixes for anonymity and delay. *Computer networks*, 67:180–200, 2014.

[48] Andrei Serjantov. *On the anonymity of anonymity systems.* PhD thesis, University of Cambridge, Computer Laboratory, October 2004.

[49] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 41–53, Berlin, Heidelberg, 2002. Springer-Verlag.

[50] Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. A survey on routing in anonymous communication protocols. *ACM Computing Surveys (CSUR)*, 51(3):1–39, 2018.

[51] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of ESORICS 2006*, September 2006.

[52] Carmela Troncoso and George Danezis. The bayesian traffic analysis of mix networks. In *Proceedings of the 16th ACM conference on Computer and communications security Security (CCS 2009)*, pages 369–379, 2009.

[53] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, pages 207–225, May 2004.

Publication

Blending Different Latency Traffic With Beta Mixing

Publication Data

Ben Guirat, I., Das, D., and Diaz, C. Blending Different Latency Traffic with Beta Mixing. *Proceedings on Privacy Enhancing Technologies* (2024)

Contributions

- Equal contribution.

Blending Different Latency Traffic With Beta Mixing

Iness Ben Guirat, Debajyoti Das, and Claudia Diaz

COSIC, KU Leuven, Belgium

Abstract. We analyze the anonymity provided by continuous mixnets (e.g., Loopix) when messages with different latency requirements are sent through the same network. The anonymity provided by existing mixnets that offer bounded latency guarantees has only been studied considering that all the traffic in the network follows the same latency distribution. In this work we evaluate whether it is beneficial to aggregate different types of traffic in the same network compared to keeping them separate, when the latency distributions are exponential and the traffic arrivals are a Poisson process — as is the case in Loopix and related designs. We present a novel evaluation method to analyze the leakage to the adversary when multiple different types of traffic are sent through the same network of continuous mixes. We apply the method to empirically evaluate the end-to-end anonymity (in terms of entropy) for each type of traffic in the presence of a global passive adversary that may additionally compromise a constant fraction of mixes or may have knowledge about the type of traffic of network output messages. Finally we show via empirical evaluation using our analytical framework that it is beneficial for anonymity to blend different types of traffic in the same mixnet.

1 Introduction

Over the past few decades, a wide range of literature has emerged discussing Anonymous Communications Networks (ACNs) [1,6,7,10,17,20,21,22], paralleled by the deployment of real-world mixnet-based systems [12]. Mix networks, or mixnets, are a variant of Anonymous Communications Networks (ACN) [1, 6, 7, 10, 17, 20, 21, 22] that were designed to protect against traffic correlation by global adversaries who can observe all the traffic in the network. They reroute the traffic through multiple servers known as *mixnodes*, that delay and reorder the messages before forwarding them, in order to hide the correlation between the input and output messages of the mixnet. While there exist a variety of mixing strategies in the literature [14], such as threshold and timed mixing,

they either add an unpredictable end-to-end latency or they provide anonymity that do not take traffic levels into consideration. For instance, threshold mixing waits for a threshold number of messages before the messages are forwarded, and therefore, adds an unpredictable end-to-end latency. Designs based on timed mixing flush out messages at predetermined time intervals, even if there are only a few messages inside the mixnodes. Tuning these parameters, such as a smaller threshold or higher time intervals, according to the needs of different applications is a tradeoff between latency and anonymity as shown in [8, 9]. On the other hand, continuous mixnets [12, 22] based on a stop-and-go mixing strategy adds a random delay (typically from exponential distribution) on each hop of a message, independent of other messages, to offer a predictable end-to-end latency as well as an average anonymity that is correlated with the traffic amount.

Therefore, this stop-and-go mixing strategy allows blending different types of traffic, each having a different latency requirement in the same mixnet, simply by drawing the different delays for each traffic type from different distributions. In this work, we investigate the implications on anonymity of blending different traffic types, each having a different latency requirement, in a single mixnet. We call the strategy of blending different traffic with different latency requirements *beta-mixing*.

The impact of blending different types of traffic on anonymity guarantees of the mixnet is an open question — all the existing analyses on continuous mixnets [3, 4, 5, 11, 15, 22] consider that all the messages delayed according to the same distribution. Their techniques are not adequate to handle the scenario when the delays for different messages are drawn from different distributions. This work aims to address this problem and answer relevant questions related to continuous mixes when multiple traffic types with different latency requirements are blended together:

1. How do we quantify the anonymity provided by the mixnet when different types of traffic are blended together?
2. Are there any advantages or disadvantages to anonymity when different types of traffic with different latency requirements are blended through the same mixnet, compared to routing each type via a separate mixnet?
3. Do other factors such as the number of layers, the number of mixes per layer, the rate delays and the traffic generations rates of each traffic type have an impact on each or both traffic types when blended together?

1.1 Contributions

This work provides the first quantitative analysis on anonymity when multiple different types of traffic are blended through a mixnet. Our results demonstrate

that blending multiple traffic types does not harm the anonymity guarantees, rather improves for continuous mixnets when the latency distributions on the mixnodes are exponential and the traffic arrivals are poisson processes — as is the case in Loopix [22] and related designs [12].

As part of our analysis technique, we present novel mathematical foundations to analyze the correlation between input and output messages of an honest mixnode in the presence an adversary that observes all the traffic in the network and derives probabilistic relationships between network inputs and outputs. Even though messages from multiple different types of traffic could potentially mix in an honest mixnode, an adversary might still be able to separate the traffic based on their individual delays, which might help the adversary track a specific message. We show that there is indeed a leakage to such an adversary, and quantify the leakage as a function of the distribution parameters of the traffic types and the observed delays (3). Based on our proposed mathematical foundations, we evaluate the anonymity in terms of entropy for the entire mixnet using empirical evaluations (4), with an illustrative example of two different types of traffic. Our evaluation results provide important insights demonstrating the benefits of blending two different types of traffic over sending them through separate mixnets:

1. Blending improves anonymity when the adversary can see the type of all messages entering the mixnet, but not the types of the messages exiting the mixnet.
2. If the adversary can observe the types of all incoming and outgoing messages, blending does not offer any advantages nor disadvantages.
3. The delay parameter for one traffic type impacts not only the anonymity of messages within that specific type but also of those from the other type.
4. Even when a single message from one traffic type is blended with messages all belonging to the other type, that message achieves significant anonymity (compared to zero anonymity when it is not blended).

The insights from the results with two types of traffic are immediately translatable to the case with more than two types (4.6). Our methods and results allow protocol designers to quantitatively evaluate the anonymity guarantees of supporting multiple applications with different latency requirements through the same mixnet deployment, and shows that it is beneficial to do so. Especially for an application with very few users, blending with other traffic provides tremendous advantage compared to not blending (which would provide almost no anonymity because of the scarcity of messages).

1.2 Related Works

In an interesting work by Dingledine et al. [16], the authors propose a technique called “alpha-mixing” to mix messages with different latencies in a mixnet. Their technique proposes a hybrid mix batching strategy to integrate users with diverse anonymity and performance goals. Senders allocate a security parameter "α" to each mix in a message's route, determining its time in each mix. Users can enhance their anonymity by increasing α, and the overall anonymity of the network increases. However, their technique is restricted to traditional deterministic batching strategies and cannot provide predictable latency for the messages. Moreover, they do not provide any quantitative anonymity analysis.

Continuous mixnets [12, 22] are particularly suited for meeting latency constraints, since the sender encodes the delays for each mixnode and can thus predict the overall latency of a message. They can, therefore, support multiple traffic with different delay distributions. In fact it is already possible to use mixnet-based network Nym [12] for Telegram and crypto-currency transactions. To the best of our knowledge, all existing analyses [3, 4, 11, 22] of continuous mixnets focus on a single type of traffic following the same exponential delay distribution. Therefore these anonymity evaluation techniques are not suitable for situations involving multiple traffic types. Even though continuous mixnets are around for two decades, our work closes the above gap for the first time by providing the mathematical tools and quantitative evaluations.

2 Problem Statement And Overview

2.1 System Model

Users. We consider a user population $\mathcal{U}$ where each user generates traffic independently of other users. Message generation for each traffic type i follows a Poisson distribution with rate λ_i' messages per user second. We note that only messages generated by honest users (not controlled by the adversary) are relevant to anonymity, and thus $\mathcal{U}$ excludes malicious users. The overall network traffic generation is therefore $\mathcal{U}.\sum_{i=1}^{n_\mathcal{T}} \lambda_i$, where $n_\mathcal{T}$ is the number of traffic types blended in the same mixnet.

Source routing. We consider a source-routed continuous mixnet [12,22], where the sending user chooses the sequence of overlay mixnodes that compose the route of a message, as well as the mixing delay to be applied to the message by each intermediary mixnode on its route.

Mixing Delays. A message's per-mix delays are drawn as independent samples from an exponential distribution [18]. The mean of the used exponential distribution depends on the type of application and its latency tolerance. The

per-mix delays are encoded by the sender in the message headers. Upon decrypting a received message, a mix node retains the message in its internal memory for the specified delay, before proceeding to forward it to its next destination in the route.

Topology. We consider a network topology where mixes are arranged in L ordered layers. The layers are interconnected such that each mix in layer i receives messages from mixes in layer $i-1$ and sends messages to mixes in layer $i+1$; while the first layer receives messages from senders and the last layer forwards messages to their final recipients. The path length of message routes is determined by the number of layers L. To select a message route, each user chooses the nodes for each message uniformly at random from each layer. Prior work has found that layered network topology provides better anonymity properties than free routes [13].

2.2 Beta-mixing

We consider the scenarios where users are using different applications and send their traffic via the mixnet. This is already the case in the Nym network [12], where users are able to send their Telegram traffic as well as cryptocurrency transactions using the same network. Currently the Nym network is using the same default delay parameters for both of the traffic types. However, users have higher latency tolerance for cryptocurrency transactions reaching 10 minutes for bitcoin [1], and lower latency tolerance for instant messages. We denote each traffic type by $\mathcal{T}_i$), the delays are chosen from exponential distribution with rate parameter λ_i. Overall, the total amount of $\mathcal{T}_i$ traffic entering the network follows Poisson distribution with parameter λ_i' messages per second. We summarize the notations in Table 1.

$\mathcal{T}_i$	i-th traffic type
λ_i'	rate generation of traffic type $\mathcal{T}_i$
λ_i	parameter of the exponential distribution for $\mathcal{T}_i$
L	number of layers in the mixnet
W	width (number of mixnodes per layer) of the mixnet
k	total number of messages in a given mixnode
k_i	number of messages of type $\mathcal{T}_i$ in the mixnode
$\mathbf{E}(X)$	expectation of a random variable X

Table 1: Summary of notation for Publication 3.

[1] https://medium.com/klaytn/a-comparison-of-blockchain-network-latencies-7508509b8460

2.3 Attacker Model And Security Goals

The adversary observes all the traffic exchanged in the network links. We assume the adversary monitors the network from the first message sent. Additionally, the adversary may compromise a fraction of the nodes (e.g., 10% of all nodes are compromised, and 90% nodes are honest). The compromised nodes are *honest but curious* — i.e., they still route messages following the protocol specifications but leak to the adversary their internal state, which per message includes the amount of delay applied in the compromised node, and its immediate predecessor and successor in the message route. Finally, we also consider an adversary who knows the type of traffic of all the network output messages.

Compromised users and active attacks. For our analysis we assume that all the senders are honest. Note that compromised senders that leak to the adversary information about their messages simply achieve that their fully traceable messages do not contribute to the anonymity of honest senders, but still cannot undermine the anonymity that honest users provide to each other. Thus, the anonymity of messages from honest users in a scenario with compromised users is simply equivalent to only considering the messages sent by the subset of honest users. We do not consider any active attacks, noting that the relevant active attacks and corresponding defense strategies mentioned in Loopix [22] are applicable, independently of using a single or multiple mixing delay distributions. We thus consider active attacks to be orthogonal to the analysis presented in this work.

2.4 Anonymity Metric

We evaluate anonymity using the entropy metric [15, 23]. Although indistinguishability based metrics [2, 19] are suitable for measuring worst-case scenarios, entropy-based metrics are better suited to capture the effect of network scaling (in terms of anonymity set size) on average anonymity. An entropy of, e.g., 10 bits, indicates that a message is as anonymous as if it was perfectly indistinguishable among about a thousand ($2^{10} = 1024$) other messages, while 11 bits correspond to perfect indistinguishability among $2^{11} = 2048$ messages. Note that the scale is logarithmic, and that an increase of one bit of entropy doubles the size of the equivalent perfect indistinguishability set, while a drop of one bit halves it.

2.5 Overview of Evaluation Strategy

In order to evaluate the entropy for an input message to the mixnet, we need to derive the probabilities that correlates the input message to the output

messages. We do that in two steps: (1) first, we derive the correlation between the input and output messages of an honest mixnode based on the observations of the adversary; (2) then, based on the above mathematical derivations, we experimentally evaluate the probabilities correlating the input and output messages of a large mixnet.

When there is a single type of traffic, all the messages inside a node are equally likely to be the next message to come out next [3,4,11,22]. However, when many types of traffic are blended, the adversary might be able to partially guess the type of an outgoing message (fast traffic tend to go out earlier than slow traffic); and that would allow the adversary to correlate messages given some background knowledge about the types of the input messages. In the next section, we derive the probabilities that correlate one input message to output messages of one honest mixnode. Finally, in Section 4, we employ this probability distribution within the entropy metric to evaluate various configurations of mixnets.

3 Analysis for a single mixnode

In this section we derive the probabilities connecting the input and output messages of a single standalone honest mixnode. For the ease of explanation, we derive the probabilities in the following steps:

- first consider the most simple case when the adversary knows the types of all messages inside the mixnode;
- then we derive the probability distributions for the number of messages of each type (assuming only two types) when the adversary knows the types of all incoming messages, but does not know which of them are still in the mixnode;
- in section 3.3, we extend our analysis for a more general case (still with two types) where the adversary knows the types of incoming messages only with certain probabilities;
- finally in section 3.6, we provide the full derivation for more than two types.

While the traffic type entering the mixnet is not immediately visible, it may be possible to infer it for the first layer, for example if the message sending rate is indicative of which application may be generating the traffic. For the second and subsequent layers, that information is not available to the adversary; however, as we will see in our derivations shortly, the adversary might be able to guess the types of messages with certain probabilities based on the observed delays in the previous layer.

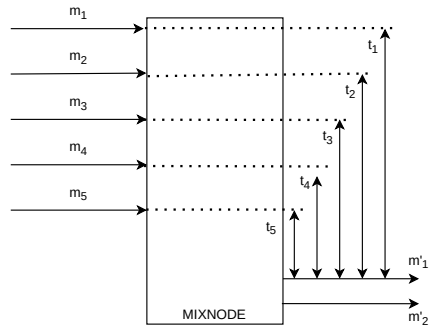


Figure 1: Adversarial observation around an honest mixnode where the mixnode receives messages m_1, m_2, m_3, m_4, m_5 at different times, and a message m'_1 goes out of the mixnode with relative time differences t_1, t_2, t_3, t_4, t_5 respectively.

3.1 Very Simple Case

As mentioned above, we first consider the most simple case where the adversary knows the types of all messages inside the mixnode. We assume that the outgoing traffic type is not directly available to the adversary, and thus we want to calculate the probability that a specific outgoing message is a target input message of a known type when the adversary does not know the types of the outgoing messages. Since the delays of the different messages are chosen from different delay distributions, based on the observed delays the adversary can partially guess which outgoing message may be of which type.

Consider the example in Figure 1: five messages m_1, m_2, m_3, m_4, m_5 are received by the mixnode at different times, and the adversary knows the types of each of them. No other messages have been received by the mixnode and no messages have left the mixnode yet. Then the first output message m'_1 goes out of the mixnode with relative time differences t_1, t_2, t_3, t_4, t_5 with each of the input arrivals, respectively. Suppose one of those five incoming messages, e.g., m_1 was sent by the target user, and the adversary wants to track that message. Let us assume that $m_1 \in \mathcal{T}_1$. We denote $m_1 = m_{\texttt{target}}$. We want to calculate the probability that $m'_1 = m_{\texttt{target}}$.

Given that m_1 did not go out before t_1, the probability that m_1 has a delay in $[t_1, t_1 + \Delta t)$ (assuming $m_1 \in \mathcal{T}_1$) can be written as follows:

$$\begin{aligned}
&\Pr\left[\mathsf{delay}(m_1) < t_1 + \Delta t \mid \mathsf{delay}(m_1) >= t_1\right] \\
=&\frac{\Pr\left[\mathsf{delay}(m_1) < t_1 + \Delta t \bigwedge \mathsf{delay}(m_1) >= t_1\right]}{\Pr\left[\mathsf{delay}(m_1) >= t_1\right]} \\
=&\frac{\int_{t_1}^{t_1+\Delta t} \lambda_1 e^{-\lambda_1 x} dx}{\int_{t_1}^{\infty} \lambda_1 e^{-\lambda_1 x} dx} \\
=&\frac{e^{-\lambda_1 t_1} - e^{-\lambda_1 (t_1+\Delta t)}}{e^{-\lambda_1 t_1}}
\end{aligned} \tag{1}$$

To generalize, let us assume that there are a total of d types of traffic and total k_j input messages of each type $\mathcal{T}_j$, $j \in \{1, 2, \ldots, d\}$. After the observations until time t_1 (denoted as $\mathcal{O}$) of the incoming messages and the first outgoing message, the probability that m_1' is a specific message m_1 can be calculated as follows:

$$
\begin{aligned}
&\Pr[m_1' = m_1 | m_1 \in \mathcal{T}_1 \wedge \mathcal{O}] \\
&= \frac{\Pr\left[\mathsf{delay}(m_1) = t_1 | m_1 \in \mathcal{T}_1 \wedge \mathsf{delay}(m_1) >= t_1\right]}{\sum_{j=1}^{d} \sum_{m_i \in \mathcal{T}_j} \Pr\left[\mathsf{delay}(m_i) = t_i | m_i \in \mathcal{T}_j \wedge \mathsf{delay}(m_i) >= t_i\right]} \\
&= \lim_{\Delta t \to 0} \frac{\Pr\left[\mathsf{delay}(m_1) < t_1 + \Delta t | m_1 \in \mathcal{T}_1 \wedge \mathsf{delay}(m_1) >= t_1\right]}{\sum_{j=1}^{d} \sum_{m_i \in \mathcal{T}_j} \Pr\left[\mathsf{delay}(m_i) < t_i + \Delta t | m_i \in \mathcal{T}_j \wedge \mathsf{delay}(m_i) >= t_i\right]} \\
&= \lim_{\Delta t \to 0} \frac{\frac{e^{-\lambda_1 t_1} - e^{-\lambda_1 (t_1 + \Delta t)}}{e^{-\lambda_1 t_1}}}{\sum_{j=1}^{d} \sum_{m_i \in \mathcal{T}_j} \frac{e^{-\lambda_j t_i} - e^{-\lambda_j (t_i + \Delta t)}}{e^{-\lambda_j t_i}}} \qquad \triangleright \textit{ using Equation (1)} \\
&= \lim_{\Delta t \to 0} \frac{\frac{\lambda_1 e^{-\lambda_1 (t_1 + \Delta t)}}{e^{-\lambda_1 t_1}}}{\sum_{j=1}^{d} \sum_{m_i \in \mathcal{T}_j} \frac{\lambda_j e^{-\lambda_j (t_i + \Delta t)}}{e^{-\lambda_j t_i}}} \qquad \triangleright \textit{ L'Hôpital's rule} \\
&= \lim_{\Delta t \to 0} \frac{\lambda_1 e^{-\lambda_1 \Delta t}}{\sum_{j=1}^{d} \sum_{m_i \in \mathcal{T}_j} \lambda_j e^{-\lambda_j \Delta t}} \\
&= \frac{\lambda_1}{\sum_{j=1}^{d} \sum_{m_i \in \mathcal{T}_j} \lambda_j} = \frac{\lambda_1}{\sum_{j=1}^{d} k_j \lambda_j} \qquad \triangleright |\mathcal{T}_j| = k_j
\end{aligned}
\tag{2}
$$

Note that the quantity $\Pr[m_1' = m_1 | m_1 \in \mathcal{T}_1 \wedge \mathcal{O}]$ is not equal to $\frac{1}{\sum_{j=1}^{d} k_j}$, and therefore, there is a bias for the message m_1' to be a specific incoming message depending on the λ_j values. Intuitively, the messages with smaller delays are more likely to be from the type with smaller average delays. We also want to compute the probability that the message m_1' is of type $\mathcal{T}_1$, and is computed as,

$$
\begin{aligned}
\Pr[m_1' \in \mathcal{T}_1 | \mathcal{O}] &= \sum_{m_i \in \mathcal{T}_1} \Pr[m_1' = m_i | m_i \in \mathcal{T}_1 \wedge \mathcal{O}] \\
&= \frac{\lambda_1}{\sum_{j=1}^{d} k_j \lambda_j} \cdot k_1 \qquad \triangleright |\mathcal{T}_1| = k_1
\end{aligned}
\tag{3}
$$

Hereafter we drop the notation for the adversarial observations $\mathcal{O}$ for brevity, and assume all the probabilities are conditional to those observations.

3.2 Second and Subsequent Output Messages

After the first message m_1' has left from the mixnode in the above example (c.f. fig. 1), we want to calculate the probability of the next message m_2' that is coming out of the mixnode is the target message $m_{\texttt{target}}$. However, now the adversary does not know the exact number of messages of each type inside the mixnode. Let us still assume that $m_1 = m_{\texttt{target}}$. There are three possible states for the mixnode after m_1' has left:

- P0: m_1' was actually m_1, which means that the mixnode does not contain m_1 anymore;
- P1: m_1' was of type $\mathcal{T}_1$ but not m_1, and now there is one less message of type $\mathcal{T}_1$ contained in the mixnode;
- P2: m_1' was of not of type $\mathcal{T}_1$ and the number of messages of type $\mathcal{T}_1$ contained in the mix remains the same.

For this subsection we assume (for simplicity) that there are only two types $\mathcal{T}_1$ and $\mathcal{T}_2$ of messages. We extend the analysis for more than two types in section 3.6. With the above assumption, the probability that the next message m_2' is the specific input message m_1 can be calculated as:

$$
\begin{aligned}
& \Pr[m_2' = m_1 | m_1 \in \mathcal{T}_1] \\
= \; & \Pr[m_2' = m_1 | P0 \wedge m_1 \in \mathcal{T}_1] \cdot \Pr[P0 | m_1 \in \mathcal{T}_1] \\
& + \Pr[m_2' = m_1 | P1 \wedge m_1 \in \mathcal{T}_1] \cdot \Pr[P1 | m_1 \in \mathcal{T}_1] \\
& + \Pr[m_2' = m_1 | P2 \wedge m_1 \in \mathcal{T}_1] \cdot \Pr[P2 | m_1 \in \mathcal{T}_1] \\
= \; & 0 \cdot \Pr[P0 | m_1 \in \mathcal{T}_1] + \frac{\lambda_1}{(k_1 - 1)\lambda_1 + k_2\lambda_2} \cdot \Pr[P1 | m_1 \in \mathcal{T}_1] \\
& + \frac{\lambda_1}{k_1\lambda_1 + (k_2 - 1)\lambda_2} \cdot \Pr[P2 | m_1 \in \mathcal{T}_1]
\end{aligned}
$$

Consequently, after i messages have left the mixnode, we need to consider all possible such combinations. Additionally, any new incoming message (after m_1' has left) would also impact the probabilities corresponding to the later messages. Most importantly, the target message could arrive anytime (possibly after m_1' has left). We need to consider all those possibilities to calculate the probabilities of the outgoing messages being the target message. So, given a specific target message $m_{\texttt{target}}$ of type $\mathcal{T}_1$ (and it can arrive anytime during the protocol run), we keep track of the state of the mixnode with the following set of random variables:

- $G(j)$ denotes the probability that there are exactly j messages of type $\mathcal{T}_1$ inside the mixnode, and the target $m_{\texttt{target}}$ has not yet arrived to the mixnode. We use $\mathcal{G}$ to denote the *event* that the target message has not arrived to the mixnode.
- $Q(j)$ denotes the probability that there are exactly j messages of type $\mathcal{T}_1$ inside the mixnode and the target message is in the mixnode. We use $\mathcal{Q}$ to denote the event that target message has arrived and is still in the mixnode.
- $R(j)$ denotes the probability that there are exactly j messages of type $\mathcal{T}_1$ inside the mixnode, and the target message has left the mixnode. We use $\mathcal{R}$ to denote the event that the target message has left the mixnode.

The quantities $Q(j)$, $G(j)$ and $R(j)$ are defined over $j \in [0, k]$ where $k = k_1 + k_2 - i$ denotes the total number of messages inside the mixnode; and i denotes the number of messages left the mixnode. Effectively, $G(j) = \Pr[\mathsf{count}(\mathcal{T}_1) = j \wedge \mathcal{G}]$, and $\Pr[\mathcal{G}] = \sum_{j=0}^{k} G(j)$. Each of these quantities are updated when a new message arrives or a message leaves the mixnode. Additionally, as we will see shortly that $R(j)$ is maintained solely for the purpose of calculating the probability of an outgoing message being of type $\mathcal{T}_1$ or $\mathcal{T}_2$.

Initialization. We initialize $G(0) = 1, Q(0) = 0$ and $R(0) = 0$ before any messages arrive, since there are exactly 0 messages of type $\mathcal{T}_1$ inside the mixnode, and the target message has not yet arrived. This is consistent with the definitions of G, Q, and R.

3.2.1 When A Message Arrives.

Until the target message arrives, $G(j)$ is updated for each $j \in [0, k+1]$ for each new incoming message m as follows:

$$G(j)^{new} = \begin{cases} G(j) & m \in \mathcal{T}_2 \\ G(j-1) & m \in \mathcal{T}_1 \wedge m \neq m_{\texttt{target}} \ , j > 0 \\ 0 & m = m_{\texttt{target}} \end{cases} \tag{4}$$

To explain briefly, whenever a message of type $\mathcal{T}_1$ arrives, the number of messages of type $\mathcal{T}_1$ in the mixnode goes from $(j-1)$ to j. If the mixnode had $(j-1)$ messages of type $\mathcal{T}_1$ with probability $G(j-1)$, now the mixnode has j messages with the same probability; and therefore, we have $G(j)^{new} = G(j-1)$. When the incoming message is of type $\mathcal{T}_2$, if the mixnode had j messages of type $\mathcal{T}_1$, the mixnode still has j messages of type $\mathcal{T}_1$; and therefore, the $G(j)$ remains unmodified. After the target message arrives, $G(j)$ becomes 0 for all $j \in [1, k]$, since the types of all incoming messages are known to the adversary.

Note that $G(j)^{new}$ (and $Q(j)^{new}$, $R(j)^{new}$ resp.) denotes the new value that will replace $G(j)$ (and $Q(j)$, $R(j)$ resp.) after the calculations are done for all the j values.

After the mixnode has received the target message, when a new message m arrives (including the target message itself), $Q(j)$ is updated for each $j \in [0, k+1]$ as follows:

$$Q(j)^{new} = \begin{cases} G(j-1) & m = m_{\texttt{target}} \\ Q(j-1) & m \in \mathcal{T}_1 \ , j > 0 \\ Q(j) & m \in \mathcal{T}_2 \end{cases} \tag{5}$$

Note that $Q(j)$ values are 0's for all for each $j \in [0, k+1]$ until the target message arrives to the mixnode. The main purpose of maintaining $G(j)$ values is to be able to correctly set the $Q(j)$ values when the target message arrives. And after the target message arrives, the update rules for $Q(j)$ are very similar to that of $G(j)$. However, very soon we are going to see that the $Q(j)$ and $G(j)$ values can be simultaneously non-zero when the adversary does not exactly know when the target message arrives (for a mixnode in the second and subsequent layers).

Analogous to $Q(j)$ values, $R(j)$ is updated for each $j \in [0, k+1]$ as follows:

$$R(j)^{new} = \begin{cases} R(j) & m \in \mathcal{T}_2 \\ R(j-1) & m \in \mathcal{T}_1, j > 0 \end{cases} \tag{6}$$

Note that, similar to $Q(j)$ values, we need to keep track of $R(j)$ values only after the target message has arrived to the mixnode. It is worth to mention here that $\sum_j Q(j)$ quantifies the probability that $m_{\texttt{target}}$ is in the mixnode, whereas, $\sum_j G(j)$ quantifies the probability that $m_{\texttt{target}}$ has not yet arrived to the mixnode. And, $\sum_j R(j)$ quantifies the probability that $m_{\texttt{target}}$ has left the mixnode.

3.2.2 When A Message Leaves.

Whenever a message m' leaves the mixnode, the probability that the message is the target message $m_{\texttt{target}}$ can be calculated as (for a total number of k

messages inside the mixnode before m' leaves),

$$\begin{aligned}
&\Pr[m' = m_{\texttt{target}}] \\
&= \sum_{1 \le j \le k} \Pr[\mathcal{Q} \wedge \mathsf{count}(\mathcal{T}_1) = j] \cdot \Pr[m' = m_{\texttt{target}} |\ j = \mathsf{count}(\mathcal{T}_1)] \\
&= \sum_{1 \le j \le k} Q(j) \cdot \frac{\lambda_1}{j\lambda_1 + (k-j)\lambda_2} \qquad \triangleright \textit{ By eq. (2)}
\end{aligned} \tag{7}$$

Explanation of Equation (7). Given that there are exactly j messages of type $\mathcal{T}_1$ and $(k-j)$ messages of type $\mathcal{T}_2$ held by the mixnode and the target message is one of those j messages, we know that the probability of the next outgoing message being the target message can be calculated as $\frac{\lambda_1}{j\lambda_1 + (k-j)\lambda_2}$ (refer to Equation (2)). The probability that the mixnode has j messages of type $\mathcal{T}_1$ and the target is inside the mixnode is given by $Q(j)$. And, we have to consider the sum over all possible j values for which $Q(j)$ is non-zero. If the mixnode does not have the target message the next outgoing message cannot be the target message, and therefore, we do not need to consider the $G(j)$ or $R(j)$ values.

Other Probabilities of an Outgoing Message. We also want to compute the probability that the message m' is of type $\mathcal{T}_1$, and can be computed as,

$$\begin{aligned}
&\Pr[m' \in \mathcal{T}_1] \\
&= \sum_{1 \le j \le k} \Pr[\mathsf{count}(\mathcal{T}_1) = j] \cdot \Pr[m' \in \mathcal{T}_1 | j = \mathsf{count}(\mathcal{T}_1)] \\
&= \sum_{1 \le j \le k} \big(G(j) + Q(j) + R(j)\big) \cdot \frac{j\lambda_1}{j \cdot \lambda_1 + (k-j)\lambda_2}
\end{aligned} \tag{8}$$

Note that when the mixnode has j messages, either the target message has not yet arrived, or it is inside the mixnode, or it has left the mixnode. Therefore, the quantity $(G(j) + Q(j) + R(j))$ represents the probability of the mixnode holding exactly j messages of type $\mathcal{T}_1$. Consequently, $\sum_j G(j) + Q(j) + R(j) = 1$.

Similarly, the probability that the message m'_1 is of type $\mathcal{T}_2$ can be computed as,

$$\begin{aligned}
&\Pr[m' \in \mathcal{T}_2] \\
&= \sum_{1 \le j \le k} \Pr[\mathsf{count}(\mathcal{T}_1) = j] \cdot \Pr[m' \in \mathcal{T}_2 | j = \mathsf{count}(\mathcal{T}_1)] \\
&= \sum_{1 \le j \le k} (G(j) + Q(j) + R(j)) \cdot \frac{(k-j)\lambda_2}{j\lambda_1 + (k-j)\lambda_2}
\end{aligned} \tag{9}$$

Update G, Q, R When A Message Leaves. After the message m' leaves the mixnode, we also need to update $Q(j)$ and $G(j)$ values, and they are updated for each $j \in [0, k-1]$ as follows:

$$\begin{aligned}
&Q(j)^{new} \\
&= Q(j) \cdot \Pr[m' \in \mathcal{T}_2 |\ j = \mathsf{count}(\mathcal{T}_1)] \\
&\ + Q(j+1) \cdot \Pr[m' \in \mathcal{T}_1 \wedge m' \neq m_{\mathtt{target}} |\ j+1 = \mathsf{count}(\mathcal{T}_1)] \\
&= Q(j) \cdot \frac{(k-j)\lambda_2}{j\lambda_1 + (k-j)\lambda_2} + Q(j+1) \cdot \frac{j\lambda_1}{(j+1)\lambda_1 + (k-j-1)\lambda_2}
\end{aligned} \tag{10}$$

For $j = k$, we update $Q(k)^{new} = 0$, since there are only $(k-1)$ messages left in the mixnode after m' has left. And,

$$\begin{aligned}
G(j)^{new} &= G(j) \cdot \Pr[m' \in \mathcal{T}_2 |\ j = \mathsf{count}(\mathcal{T}_1)] \\
&\quad + G(j+1) \cdot \Pr[m' \in \mathcal{T}_1 |\ j+1 = \mathsf{count}(\mathcal{T}_1)] \\
&= G(j) \cdot \frac{(k-j)\lambda_2}{j\lambda_1 + (k-j)\lambda_2} \\
&\quad + G(j+1) \cdot \frac{(j+1)\lambda_1}{(j+1)\lambda_1 + (k-j-1)\lambda_2}
\end{aligned} \tag{11}$$

For $j = k$, we update $G(k)^{new} = 0$. Additionally,

$$\begin{aligned} R(j)^{new} &= R(j) \cdot \Pr[m \in \mathcal{T}_2 |\ j = \mathsf{count}(\mathcal{T}_1)] \\ &\quad + R(j+1) \cdot \Pr[m \in \mathcal{T}_1 |\ j+1 = \mathsf{count}(\mathcal{T}_1)] \\ &\quad + Q(j+1) \cdot \Pr[m' = m_{\mathtt{target}} |\ j+1 = \mathsf{count}(\mathcal{T}_1)] \\ &= R(j) \cdot \frac{(k-j)\lambda_2}{j\lambda_1 + (k-j)\lambda_2} \\ &\quad + R(j+1) \cdot \frac{(j+1)\lambda_1}{(j+1)\lambda_1 + (k-j-1)\lambda_2} \\ &\quad + Q(j+1) \cdot \frac{\lambda_1}{(j+1)\lambda_1 + (k-j-1)\lambda_2} \end{aligned} \tag{12}$$

For $j = k$, we update $R(k)^{new} = 0$.

3.3 General Case With Two Traffic Types

For a mixnode on the second and consequent layers of a mixnet, the adversary might not exactly know the types of the incoming messages to the mixnode. However, based on the observed delays on the previous layers (and following the analysis in Section 3.2), the adversary can guess the type of each message with some probability. With that consideration, we want to analyze how easily the adversary can correlate the outgoing messages with the incoming messages. For example, for a mixnode on the second layer the adversary can compute the probabilities of each incoming message being type $\mathcal{T}_1$ (as shown in Section 3.2), and each of them will have a probability of being the target message (we are still assuming that the target message is a message from the traffic type $\mathcal{T}_1$). In such cases, we want to compute the probabilities of the outgoing messages of being the target message.

Similar to Section 3.2, we still assume that there are only two types of messages: $\mathcal{T}_1$ and $\mathcal{T}_2$. We keep track of the state of the mixnode using the variables $Q(j)$, $G(j)$ and $R(j)$ for $j \in [0, k]$ where k denotes the total number of message held by the mixnode. For a mixnode in the first layer, the adversary knows the exact number k_1 (resp. k_2) of messages of type $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) came to the mixnode. However, for a mixnode in the second layer, those quantities are probabilistic and dependent on the first layer. Additionally, the adversary does not know when the target message arrives to the mixnode, if at all (since there can many mixnodes in every layer).

3.3.1 Update G, Q, R When A Message Arrives

Before any messages arrive we initialize $G(0) = 1, Q(0) = 0$ and $R(0) = 0$. When a new message m arrives, $G(j)$ can be updated for each $j \in [1, k+1]$ as:

$$\begin{aligned}
& G(j)^{new} \\
=\ & \Pr[m \in \mathcal{T}_2 \wedge \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j] \\
& + \Pr[m \in \mathcal{T}_1 \wedge m \neq m_{\mathtt{target}} \wedge \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j-1] \\
=\ & \Pr[m \in \mathcal{T}_2 \mid \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j] \times G(j) \\
& + \Pr[m \in \mathcal{T}_1 \wedge m \neq m_{\mathtt{target}} \mid \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j-1] \times G(j-1) \\
=\ & \Pr[m \in \mathcal{T}_2 \mid \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j] \times G(j) \\
& + \big(\Pr[m \in \mathcal{T}_1 \mid \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j-1] \\
& - \Pr[m = m_{\mathtt{target}} | \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j-1] \big) \times G(j-1)
\end{aligned} \tag{13}$$

And for $j = 0$ we can update

$$G(0)^{new} = G(0) \times \Pr[m \in \mathcal{T}_2 \mid \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j].$$

Note that the number of messages of type $\mathcal{T}_1$ in the mixnode on the second (or subsequent) layer depends on the state of the previous layer(s). In that sense, $\Pr[m \in \mathcal{T}_2 \mid \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j]$ depends on the state of the mixnode, which in turn depends on the state of previous layers. We consider the following approximation: the probability of an incoming message being of type $\mathcal{T}_1$ (resp. type $\mathcal{T}_2$) is independent of $\mathsf{count}(\mathcal{T}_1)$ of the current mixnode; and therefore, $\Pr[m \in \mathcal{T}_1 \mid \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j] = \Pr[m \in \mathcal{T}_1 \mid \mathcal{G}] = \Pr[m \in \mathcal{T}_1]$. Similarly, $\Pr[m \in \mathcal{T}_2 \mid \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j] = \Pr[m \in \mathcal{T}_2]$. Note that the type of the incoming message m is always independent of where the target message is, however, $\Pr[m = m_{\mathtt{target}}]$ is not. So, we have the following,

$$\begin{aligned}
G(j)^{new} = \ & \Pr[m \in \mathcal{T}_2] \times G(j) + \Bigg(\Pr[m \in \mathcal{T}_1] - \\
& \frac{\Pr[m = m_{\mathtt{target}}]}{\Pr[\mathcal{G}]} \Bigg) \times G(j-1).
\end{aligned} \tag{14}$$

Note that we have used $\Pr[m = m_{\mathtt{target}} | \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j-1] = \dfrac{\Pr[m = m_{\mathtt{target}}]}{\Pr[\mathcal{G}]}$ in the final equation. That is because the target message

can be present only with one mixnode. Therefore, m could only be a target message if it has not arrived to the current mixnode before, or in other words, if $\mathcal{G}$ is true. Consequently,

$$\Pr[m = m_{\texttt{target}} \wedge \ \mathcal{G}] = \Pr[m = m_{\texttt{target}}]$$

$$\iff \Pr[m = m_{\texttt{target}} |\ \mathcal{G}] = \frac{\Pr[m = m_{\texttt{target}}]}{\Pr[\mathcal{G}]}$$

The justification for the approximation is two-fold:

- In a steady-state flow, based on the analysis from [5], it is a good approximation to consider each mixnode as an independent $M/M/\infty$ queue. That means, the internal states of the nodes can be considered independent of each other; except for the adversarial knowledge of the total number of messages contained in a node. With that approximation, the probability that an incoming message from the previous layer is of type $\mathcal{T}_1$ or $\mathcal{T}_2$ is independent of the state of current mixnode. However, the adversary has the knowledge that the target message can be held by only one mixnode at any given point. Therefore, $\Pr[m = m_{\texttt{target}}]$ for an incoming message m is not independent of the state of the current mixnode, and is bounded by $\Pr[\mathcal{G}]$.
- If we want to plug-in these probability calculations in a simulator (which we do in Section 4), maintaining the inter-dependent states for a mixnet with multiple layers and multiple mixnodes per layer explodes the computational complexity, and becomes a serious performance bottleneck.

$Q(j)$ is updated for each $j \in [1, k+1]$ as:

$$\begin{aligned} Q(j)^{new} = \ & \Pr[m \in \mathcal{T}_2 \wedge \mathcal{Q} \wedge \mathsf{count}(\mathcal{T}_1) = j] \\ & + \Pr[m \in \mathcal{T}_1 \wedge m \neq m_{\texttt{target}} \wedge \mathcal{Q} \wedge \mathsf{count}(\mathcal{T}_1) = j-1] \\ & + \Pr[m = m_{\texttt{target}} \wedge \mathcal{G} \wedge \mathsf{count}(\mathcal{T}_1) = j-1] \\ = \ & Q(j) \cdot \Pr[m \in \mathcal{T}_2] + Q(j-1) \cdot \Pr[m \in \mathcal{T}_1] \\ & + G(j-1) \cdot \frac{\Pr[m = m_{\texttt{target}}]}{\Pr[\mathcal{G}]} \end{aligned} \tag{15}$$

And for $j = 0$, we update $Q(0)^{new} = Q(0) \cdot \Pr[m \in \mathcal{T}_2]$.

Explanation for Equations (14) and (15) in Simple Words. The concept of transition between G and Q is still similar to the previous section (Section 3.2). However, now the transition is probabilistic, since the adversary does not certainly know if an incoming message m is the target message. If m has a

probability P of being the target message, overall $\Pr[\mathcal{Q}^{new}]$ should be equal to $\Pr[\mathcal{Q}] + \mathsf{P}$; in other words, $\sum Q(j)^{new} = \sum Q(j) + P$.

For each j, $G(j)^{new}$ is contributed by $G(j)$ with an amount exactly as the probability that m belongs to $\mathcal{T}_2$, and $G(j-1)$ with an amount exactly as the probability that m belongs to $\mathcal{T}_1$ but not the target. So, in total G quantities are reduced by an amount same as the probability of m being the target. And, that amount is added to the total of Q quantities by adding the amounts that we have just reduced from G values to each $Q(j)^{new}$. So, for each j, $Q(j)^{new}$ is contributed by $Q(j)$ (with an amount exactly as the probability that m belongs to $\mathcal{T}_2$), $Q(j-1)$ (with an amount exactly as the probability that m belongs to $\mathcal{T}_1$), and the amount laid off from $G(j-1)$.

Note that $G(j)^{new}$ (resp. $Q(j)^{new}$) is contributed by $G(j)$ (resp. $Q(j)$) for the amount as the probability that m belongs to $\mathcal{T}_2$ because the number of messages of $\mathcal{T}_1$ remains the same when $m \in \mathcal{T}_2$. Analogously, $G(j)^{new}$ (resp. $Q(j)^{new}$) is contributed by $G(j-1)$ because the number of messages of $\mathcal{T}_1$ increases by 1 when $m \in \mathcal{T}_1$.

Analogously, $R(j)$ is updated for each $j \in [1, k+1]$ as follows:

$$\begin{aligned} R(j)^{new} = R(j) \cdot \Pr[m \in \mathcal{T}_2] \\ + R(j-1) \cdot \Pr[m \in \mathcal{T}_1 \wedge m \neq m_{\texttt{target}}] \end{aligned} \tag{16}$$

And for $j = 0$, we update $R(0)^{new} = R(0) \cdot \Pr[m \in \mathcal{T}_2]$.

3.3.2 When A Message Leaves.

When a message leaves from the mixnode we update the quantities $G(j), Q(j), R(j)$ for $j \in [0, k]$ exactly same as in section 3.2. The probability that an outgoing message is the target message, the probability that it belongs to a specific type (e.g., $\mathcal{T}_1$) are also computed in the same way as in section 3.2.

We present the overall methodology to calculate the probabilities for a given mixnode as part of a simulator in Algorithm 2. It is worth to mention here that our method also works when there is only one type of traffic going through the mixnet, and provides the same results as the existing methods [15, 23] for a single traffic type.

Target Is From Type $\mathcal{T}_2$: When the target message is from $\mathcal{T}_2$ the derivation remain exactly the same, however, all the quantities are to be defined for type $\mathcal{T}_2$, which is equivalent to swapping the assignment of the types in the notation.

3.4 When the Adversary Does Not Know the Types of Input Messages

It is possible that the adversary does not know the types of the incoming messages to the mixnet (possibly every client is generating both types of traffic). It is still possible to extend our analysis strategy in such scenarios.

Suppose, the total number of observed packets is $\mathcal{N}$, then we have $X_1 + X_2 = \mathcal{N}$ where $X_1 \sim \mathsf{Poisson}(\lambda_1')$ and $X_2 \sim \mathsf{Poisson}(\lambda_2')$. Then we can say for an incoming message m,

$$\Pr[m \in \mathcal{T}_1] = \sum_{i=0}^{\mathcal{N}} \frac{i}{\mathcal{N}} \times \Pr[X_1 = i | X_1 + X_2 = \mathcal{N}]$$

where $(X_1 | X_1 + X_2 = \mathcal{N}) \sim \mathsf{Binom}(\mathcal{N}, \frac{\lambda_1'}{\lambda_1' + \lambda_2'})$. Therefore, the above quantity can be further reduced to:

$$\begin{aligned}\Pr[m \in \mathcal{T}_1] &= \frac{1}{\mathcal{N}} \sum_{i=0}^{\mathcal{N}} i \times \Pr[X_1 = i | X_1 + X_2 = \mathcal{N}] \\ &= \frac{1}{\mathcal{N}} \times \mathbf{E}\left[\mathsf{Binom}\left(\mathcal{N}, \frac{\lambda_1'}{\lambda_1' + \lambda_2'}\right)\right] = \frac{\lambda_1'}{\lambda_1' + \lambda_2'}\end{aligned}$$

Therefore, the value of $\dfrac{\Pr[m \in \mathcal{T}_1]}{\Pr[m \in \mathcal{T}_2]}$ does not depend on the total number of messages passing through the mixnode, or the number of messages the adversary observes.

Now the overall calculation of mapping probabilities between the incoming and outgoing messages of a mixnode is similar to the previous subsection, where the adversary knows that an incoming message i belongs to a specific type with some probability $0 \leq p_i \leq 1$. However, the adversary does not know the type of the target message. Suppose, the target message $m_{\mathtt{target}}$ is of type $\mathcal{T}_1$ with probability p and of type $\mathcal{T}_2$ with probability $1 - p$. Then the analysis for the whole mixnet needs to be done twice: once assuming $m_{\mathtt{target}} \in \mathcal{T}_1$, and then assuming $m_{\mathtt{target}} \in \mathcal{T}_2$. Suppose the probabilities of an outgoing message m_i' being the target message are $p_{i,1}$ and $p_{i,2}$ in those two analyses. Then the final probability that $m_i' = m_{\mathtt{target}}$ is calculated as $p \cdot p_{i,1} + (1 - p) \cdot p_{i,2}$.

3.5 When the Recipient Leaks the Types

If the mixnode is the last layer and the recipient leaks the type of the message it is receiving, the adversary gains additional knowledge. In such cases, the probabilities need to be adjusted to consider that factor. Suppose, the outgoing messages from the mixnode are denoted with m'_1, m'_2, ...etc. And the probabilities of them being the target message $m_{\texttt{target}}$ are $p_1, p_2, \ldots$ respectively, without using the knowledge from the recipient side. Let us consider that $m_{\texttt{target}} \in \mathcal{T}_1$. With the additional knowledge from the recipient side, we can derive the following for an outgoing message $m'_i \in \mathcal{T}_1$,

$$\begin{aligned}
&\Pr[m'_i = m_{\texttt{target}} \mid m'_i \in \mathcal{T}_1 \wedge m_{\texttt{target}} \in \mathcal{T}_1] \\
&= \frac{\Pr[m'_i = m_{\texttt{target}} \wedge m'_i \in \mathcal{T}_1 \mid m_{\texttt{target}} \in \mathcal{T}_1]}{\Pr[m'_i \in \mathcal{T}_1 \mid m_{\texttt{target}} \in \mathcal{T}_1]} \\
&= \frac{\Pr[m'_i = m_{\texttt{target}} \wedge m'_i \in \mathcal{T}_1 \mid m_{\texttt{target}} \in \mathcal{T}_1]}{\Pr[m'_i \in \mathcal{T}_1]} \;=\; \frac{p_i}{D_1};
\end{aligned}$$

where $D_1 = \sum_{i:m'_i \in \mathcal{T}_1} p_i$. Similarly, for an outgoing message $m'_i \in \mathcal{T}_2$ we can derive,

$$\begin{aligned}
&\Pr[m'_i = m_{\texttt{target}} \mid m'_i \in \mathcal{T}_2 \wedge m_{\texttt{target}} \in \mathcal{T}_1] \\
&= \frac{\Pr[m'_i = m_{\texttt{target}} \wedge m'_i \in \mathcal{T}_2 \mid m_{\texttt{target}} \in \mathcal{T}_1]}{\Pr[m'_i \in \mathcal{T}_2 \mid m_{\texttt{target}} \in \mathcal{T}_1]} \;=\; 0.
\end{aligned}$$

Therefore, we can adjust the probabilities for the leakage on the recipient side by normalizing them for all the outgoing messages of type $\mathcal{T}_1$ (for the target message $m_{\texttt{target}} \in \mathcal{T}_1$).

3.6 More Than Two Types of Traffic

The methodology presented until now can be extended to analyze anonymity when there are more than two types of traffic. However, an additional set of quantities would be required to keep track of the number of messages for each type inside the mixnode, except the type of the target.[2] However, we do not need to modify G, Q, R calculations, because they only concern about if an incoming or outgoing message is of the same type as the target or not. The new quantities corresponding to every type are very similar to e.g., G with a

[2]With only two types, the number of messages of type $\mathcal{T}_2$ can easily be calculated if the number of messages of $\mathcal{T}_1$ and the total number of messages are known. However, that is not the case when there are many types.

slight difference — they are not conditional on the arrival (or departure) of the target message. How they are updated when a message arrives or leaves are also similar.

Assuming that the target is from type $\mathcal{T}_1$, let $H_w(j)$ denote the probability that there are j messages inside the mixnode of type $\mathcal{T}_w$ except $w = 1$; and,

$$H_w(j|j<x) = \begin{cases} \dfrac{H_w(j)}{\sum_{a=0}^{x} H_w(a)} & j < x \\ 0 & \text{otherwise} \end{cases}$$

denote the probability that there are j messages inside the mixnode of type $\mathcal{T}_t$ but conditioned on $j < x$. Note that $H_1(j) = G(j) + Q(j) + R(j)$. Once the H_w quantities are in place, the probabilities of an outgoing message being a target message, or of a specific type can be computed similar to Section 3.3. For example, the probability of an outgoing message being the target message can be calculated as (assuming a total of d types),

$$\begin{aligned} &\Pr[m' = m_{\texttt{target}}] \\ &= \sum_{1\le k_d\le k} \cdots \sum_{1\le k_1\le k} H_d(k_d)\cdots H_2(k_2) \times Q(k_1) \\ &\qquad \times \Pr[m' = m_{\texttt{target}} \mid k_t = \mathsf{count}(\mathcal{T}_t)\ \forall 1 \le t \le k_d] \\ &= \sum_{k_d=0}^{k} \cdots \sum_{k_1=1}^{k} H_d(k_d|k_d \le k - k_{d-1} - \cdots - k_1)\cdots \\ &\qquad \times H_2(k_2|k_2 \le k - k_1) \times Q(k_1) \times \frac{\lambda_1}{\sum_{a=1}^{d} k_a\lambda_a} \end{aligned} \tag{17}$$

However, the H_w quantities need to be updated whenever a message arrives or leaves. Whenever a new message arrives, $H_t(j)$ values can be updated as follows,

$$H_w(j)^{new} = H_w(j)\cdot\Pr[m \in \mathcal{T}_w] + H_w(j-1)\cdot\Pr[m \notin \mathcal{T}_w];$$

where $\Pr[m \in \mathcal{T}_w]$ and $\Pr[m \notin \mathcal{T}_w]$ are calculated based on the previous layer. And when a message leaves,

$$\begin{aligned} H_w(j)^{new} = {} & H_w(j)\cdot\Pr[m \notin \mathcal{T}_w|\ j = \mathsf{count}(\mathcal{T}_w)] \\ & + H_w(j+1)\cdot\Pr[m \in \mathcal{T}_w|\ j+1 = \mathsf{count}(\mathcal{T}_w)] \end{aligned} \tag{18}$$

For $w = 1$ we can say,

$$\begin{aligned}
&\Pr[m \in \mathcal{T}_w |\ j = \mathsf{count}(\mathcal{T}_w)] \\
&= \sum_{k_d=0}^{k-j} \cdots \sum_{k_2=1}^{k-j} H_d(k_d | k_d \leq k - k_{d-1} - \cdots - k_2 - j) \cdots \\
&\quad \times H_2(k_2 | k_2 \leq k - j) \times \frac{j\lambda_w}{\sum_{a=2}^{d} k_a \lambda_a + j\lambda_w}
\end{aligned} \tag{19}$$

and $\Pr[m \notin \mathcal{T}_w |\ j = \mathsf{count}(\mathcal{T}_w)] = 1 - \Pr[m \in \mathcal{T}_w |\ j = \mathsf{count}(\mathcal{T}_w)]$. The evaluation is exactly the same for any other w, except for the switched indices of the variables.

4 End-to-end Anonymity Analysis for Mixnets with Beta-mixing

In order to demonstrate the effect of blending, we provide empirical analysis for end-to-end mixnets with the simple case of two types of traffic, and discuss in section 4.6 how to extend the insights from these analysis when there are more types.

4.1 Methodology

Our methodology to evaluate the impact of blending different traffic types on top of the same mixnet is as follows: First, we presented our analytical method in the previous section which shows how to calculate the probability of an output message being one input message. We then modified the open-source simulator used in [4] by implementing our analytical method. The modifications made to the simulator are summarized in Algorithm 2. We executed the updated simulator across various mixnet configurations (number of nodes, number of layers, rates of the different traffic generations etc). Finally, as a measure of anonymity, we evaluate the entropy of the probability distribution linking the mixnet's input and output messages [15, 23]. For our evaluations, we assume that the adversary knows the types of the messages coming to the mixnet. Note that, while the traffic type is not immediately visible, it may be possible to infer it, for example if the message sending rate is indicative of which application may be generating the traffic.

Algorithm 2: Probability computation on a single node with two types of traffic.

Result: Updated $\Pr[m_i = m_t | m_t \in \mathcal{T}_1]$.
Initialize:
G, Q, R : arbitrarily expandable Lists ;
$G.append(1)$; $Q.append(0)$;
$k = 0$; $i = 0$;
if *event(receive(m_i))* **then**
  $k++$;
  for $j \leftarrow 0$ **to** k **do**

$$Q[j] = Q[j-1] \cdot \Pr[m_i \in \mathcal{T}_1] + Q[j] \cdot \Pr[m_i \in \mathcal{T}_2] + G[j-1] \cdot \frac{\Pr[m_i = m_t]}{\mathsf{sum}(G)};$$

$$G[j] = G[j] \cdot \Pr[m_i \in \mathcal{T}_2] + G[j-1] \cdot \Big(\Pr[m_i \in \mathcal{T}_1] - \frac{\Pr[m_i = m_t]}{\mathsf{sum}(G)} \Big);$$

$$R[j] = R[j-1] \cdot \Pr[m_i \in \mathcal{T}_1] + R[j] \cdot \Pr[m_i \in \mathcal{T}_2];$$

  end
end
if *event(send(m_i))* **then**
  define $\mathsf{denom}(j) = j \cdot \lambda_1 + (k - i - j) \cdot \lambda_2$;

$$\Pr[m_i = m_t] = \sum_{j=0}^{k-1} \frac{\lambda_1}{\mathsf{denom}(j)} \cdot Q[j];$$

$$\Pr[m_i \in \mathcal{T}_1] = \sum_{j=0}^{k-1} \frac{j \cdot \lambda_1}{\mathsf{denom(j)}} \cdot (Q[j] + R[j] + G[j]);$$

$$\Pr[m_i \in \mathcal{T}_2] = \sum_{j=0}^{k-1} \frac{(k - i - j) \cdot \lambda_2}{\mathsf{denom}(j)} \cdot \big(Q[j] + R[j] + G[j]\big);$$

  for $j \leftarrow 0$ **to** k **do**

$$R[j] = \frac{\lambda_1}{\mathsf{denom}(j+1)} \cdot Q[j+1] + \frac{(j+1) \cdot \lambda_1}{\mathsf{denom}(j+1)} \cdot R[j+1] + \frac{(k - i - j) \cdot \lambda_2}{\mathsf{denom}(j)} \cdot R[j];$$

$$Q[j] = \frac{j \cdot \lambda_1}{\mathsf{denom}(j+1)} \cdot Q[j+1] + \frac{(k - i - j) \cdot \lambda_2}{\mathsf{denom}(j)} \cdot Q[j];$$

$$G[j] = \frac{(j+1) \cdot \lambda_1}{\mathsf{denom}(j+1)} \cdot G[j+1] + \frac{(k - i - j) \cdot \lambda_2}{\mathsf{denom}(j)} \cdot G[j];$$

  end
  Forward Message (m_i);
  $i++$;
end

4.2 Experimental Setup

The simulation starts with users generating messages, selecting a route for each message, and sending them through the network to their respective recipients. Each user generates messages from two traffic types, $\mathcal{T}_1$ and $\mathcal{T}_2$, following a Poisson distribution with parameters λ'_1 and λ'_2. The delays parameters of each message belonging to either $\mathcal{T}_1$ or $\mathcal{T}_2$ also follow Poisson Distribution with parameters λ_1 and λ_2, respectively. We consider a user population of $\mathcal{U} = 100$ users, each user generating a total of 5 messages per second for the two types of traffic combined. The total traffic generation rate is 500 messages per second with a total of 100000 messages. For each simulation run, each run representing one data point in the graphs, once the network has been initialized and is in a steady state we choose 50 input target messages[3]. In order to evaluate the impact of blending two types of traffic on the anonymity of the system for each of these types of traffic, we choose target messages from each type of traffic. $m_{\texttt{target}}$ is the message that the adversary follows. At the end of the simulations, all the 100000 received messages, in one simulation run, have a probability of being $m_{\texttt{target}}$. Finally we plug this probability distribution in the entropy metric in order to evaluate the anonymity provided by the mixnet.

We vary for different experiments the generation rates of each traffic such that the sum of the two rates of traffic generations $(\lambda'_1 + \lambda'_2)$ is equal to 5. Our goal is to evaluate whether there are advantages of blending two different traffic types with different latency requirements in the same mixnet over maintaining separate mixnet infrastructures as well as determining the significance of the traffic generation ratios (λ'_1:λ'_2). We consider $\mathcal{T}_1$ as `fast` traffic, and $\mathcal{T}_2$ as `slow` traffic, meaning all messages belonging to $\mathcal{T}_1$ have an average delay $\frac{1}{\lambda_1} = d_1 = 1$, and messages from $\mathcal{T}_2$ have higher delays. For each experiment, we plot the entropy values, for two cases: (i) for target messages belonging to $\mathcal{T}_1$ and (ii) for messages belonging to $\mathcal{T}_2$. To evaluate the impact of blending traffic on the anonymity provided by the mixnet for messages from:

- $\mathcal{T}_1$ (target messages are from type $\mathcal{T}_1$): The first 4 data points represent the entropy values for : ($\lambda'_1 = 5$, $\lambda'_2 = 0$), ($\lambda'_1 = 4$,$\lambda'_2 = 1$), ($\lambda'_1 = 2.5$, $\lambda'_2 = 2.5$), and ($\lambda'_1 = 1$, $\lambda'_2 = 4$) The 5th data point of this graph represent only one target message from $\mathcal{T}_1$ and the rest of the network traffic is from $\mathcal{T}_2$ with $\lambda'_2 = 5$.
- $\mathcal{T}_2$ (target messages are from type $\mathcal{T}_2$): The first 4 data points represent the entropy values for: ($\lambda'_1 = 0$, $\lambda'_2 = 5$), ($\lambda'_1 = 1$,$\lambda'_2 = 4$), ($\lambda'_1 = 2.5$, $\lambda'_2 = 2.5$), and ($\lambda'_1 = 4$, $\lambda'_2 = 1$). The 5th data point of this graph represent only one

[3]The simulation times vary for experiments due to the extensive probability computation for each output message. We will include the open-source code, the data, as well as other details related to the simulator as an artefact.

target message from $\mathcal{T}_2$ and the rest of the network traffic is from $\mathcal{T}_1$ with $\lambda'_1 = 5$.

In order to compare anonymity of the messages of $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) where traffic is blended to the anonymity where there's a dedicated infrastructure for each traffic type, we also plot the entropy values for the same values of λ'_1 (resp. λ'_2) but all values of λ'_2 (resp. λ'_1) are equal to 0. We call the scenario a `Solo` case, meaning that the network only has messages from the the type traffic $\mathcal{T}_1$ ($\mathcal{T}_2$). Finally, in the scenario of one single message from either $\mathcal{T}_1$ or $\mathcal{T}_2$, we want to evaluate the anonymity provided by the network when there's only one message from that type of traffic and the rest of the 500 messages per second are from the opposite traffic. Such scenarios can manifest in real-world situations. For instance, in the case of Nym [12], a practical scenario might involve one user initiating a cryptocurrency transaction, while the rest of the network during a rather large period of time, are sending Telegram messages. We consider the following mixnet settings for our evaluations:

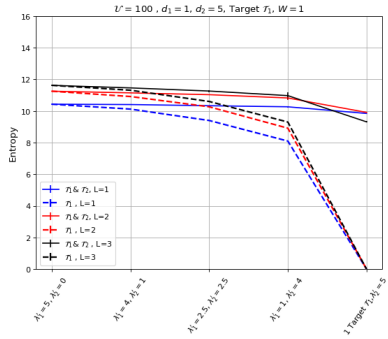
1. `Cascade:` One mixnode per layer for $L = 1$, $L = 2$ and $L = 3$ (4.3);
2. `3x10:` A mixnet consisting of 3 layers with 10 mixnodes per layer (4.4):
   - against a Global Passive Adversary (GPA);
   - the GPA additionally compromises 10% mixnodes;
   - the GPA can see the types of all output messages;
3. d_1`:`d_2`:` Different ratios of per-mix delays; $\mathcal{T}_1$ with an average delay d_1=1 and $\mathcal{T}_2$ with average delays $d_2 = 5$, $d_2 = 10$, $d_2 = 15$ and $d_2 = 20$ (4.5);

4.3 Evaluation: One Mixnode Per Layer

First, we evaluate the entropy for a small mixnet: one mixnode per layer for $L = 1$ (one single standalone mixnode), $L = 2$ and for $L = 3$. We plot the entropy values for the different λ'_1 and λ'_2 ratios evaluating the anonymity provided by the mixnet for messages from $\mathcal{T}_1$ in Figure 2a and from $\mathcal{T}_2$ in Figure 2b.

In these figures, the blue lines denote the entropy values for 1 single mixnode, the red ones denote the entropy for $L = 2$ and the black is for $L = 3$. The solid lines for each of these configuration denote the entropy values of the two types of traffic blended together and the dashed lines are for the `Solo` cases, meaning that all λ' values from the opposite traffic are equal to 0.

As we can see in Figure 2a, as the ratio $\lambda'_1 : \lambda'_2$ declines when blending traffic (solid lines), the entropy only slightly decreases. The `Solo` cases, represented by the dashed lines, show the entropy values for single traffic $\mathcal{T}_1$ in the mixnet. The decreasing entropy values of the `Solo` cases are to be expected since there is a decrease in the traffic generation rate λ'_1 in 2a. We use the `Solo` cases as



(a) Entropy for messages of type $\mathcal{T}_1$ for $W = 1$, $L = 1$, $L = 2$ and $L = 3$.

(b) Entropy for messages of type $\mathcal{T}_2$ for $W = 1$, $L = 1$, $L = 2$ and $L = 3$.

Figure 2: Evaluation of anonymity in terms of entropy for continuous mixnets with width $W = 1$ (number of mixnodes per layer), average delay $d_1 = \frac{1}{\lambda_1} = 1$ for traffic type $\mathcal{T}_1$, average delay $d_2 = \frac{1}{\lambda_2} = 5$ for traffic type $\mathcal{T}_2$.

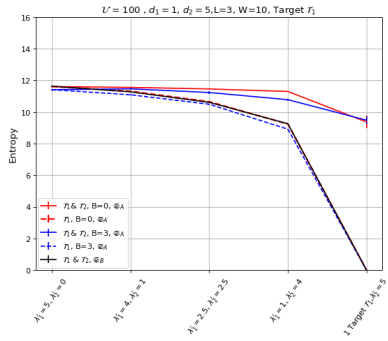
reference in order to compare the impact of blending on anonymity to dedicating a different mixnet per traffic type. We conclude from this graph that even though there's a slight decrease, the overall anonymity is much better when blending traffic. This is due to the fact that messages from $\mathcal{T}_2$ do make up for the reduced number of messages of $\mathcal{T}_1$.

Figure 2b provides similar observations. Additionally, it shows that the slow traffic ($\mathcal{T}_2$) has much better entropy compared to the fast traffic ($\mathcal{T}_1$) for both `Solo` and with blending. This is due to the higher per-mix delay ($d_2 = 5$) for messages from $\mathcal{T}_2$ compared to messages from $\mathcal{T}_1$ ($d_1 = 1$). However we do notice a slight increase in anonymity for traffic type $\mathcal{T}_2$ when the ratios $\lambda'_2 : \lambda'_1$ declines. This is due to the fact that messages $\mathcal{T}_2$ are able to meet with many more messages of fast traffic ($\mathcal{T}_1$) when they are inside the mix because of their higher delays. In other words, for a message with a large delay, having other messages with small delays decrease the probability of that message being one input message and hence this provide higher value of entropy.

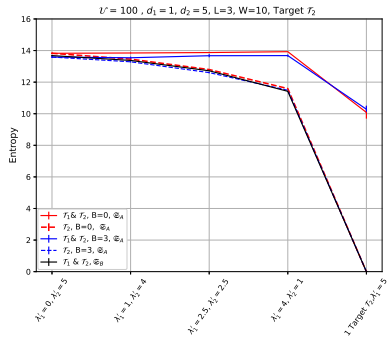
When comparing the entropy values for a single mixnode to the case of $L = 2$ and $L = 3$, we notice that adding layers increase anonymity, a result that is consistent with previous works [3, 4, 22] analysing only a single type of traffic. For example, when having $L = 1$ and $W = 1$ (a single standalone mixnode represented by the blue lines), we have entropy values of around 10.3 for type $\mathcal{T}_1$ and $\lambda'_1 = 1, \lambda'_2 = 4$; and entropy values of around 11.5 for the same λ' when $L = 3$ and $W = 1$ (black lines).

However, when there's only one single unique target message, we notice that adding layers has the opposite impacts, especially for the slow traffic. This is due to the fact that the adversary has more advantages in following this unique target message that has a different delay compared to all ther other messages in the network. We emphasize however the importance of blending, because otherwise this *unlucky* message would be 100% de-anonymized as shown by the dashed lines representing the `solo` cases.

We emphasize the validity of our analysis by comparing our simulation results in `Solo` cases with those from [4]. This comparison is justified as the authors in [4] exclusively evaluated one type of traffic, and both methods yield identical entropy values when utilizing the same network size, user population, and traffic generation parameters.



(a) Entropy for messages of type $\mathcal{T}_1$ for a network with $L = 3$, and $W = 10$ and two adversarial setups.



(b) Entropy for messages of type $\mathcal{T}_2$ for a network with $L = 3$, and $W = 10$ and two adversarial setups.

Figure 3: Evaluation of anonymity in terms of entropy for continuous mixnets with width $W = 10$ (number of mixnodes per layer), number of layers $L = 3$, average delay $d_1 = \frac{1}{\lambda_1} = 1$ for traffic type $\mathcal{T}_1$, average delay $d_2 = \frac{1}{\lambda_2} = 5$ for traffic type $\mathcal{T}_2$. B is the number of compromised mixnodes in the network. Scenario $\mathcal{A}$: Adversary only knows the input types of traffic. Scenario $\mathcal{B}$: Adversary knows the type of traffic of input and output messages

4.4 Evaluation: 3 Layers, 10 Mixnodes Per Layer

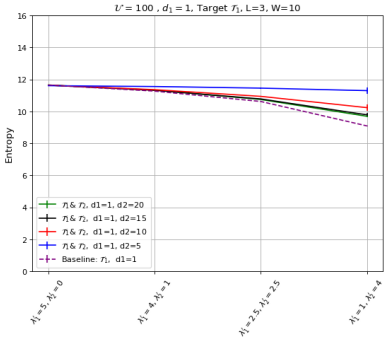
In this section, we evaluate the impact of blending the two types of traffic in a network with $L = 3$ and $W = 10$. We maintain the same experimental setups, in terms of number of clients and number of messages per second, as in the previous experiments. Additionally, we evaluate the impact of having

an adversary who is corrupting 10% of the network ($B = 3$) as well as the impact of an adversary that additionally knows the traffic type of each output message. We symbolise by $\mathcal{A}$ the adversary who only knows the types of input messages and by $\mathcal{B}$ the adversary who knows the the types of traffic of all input and output messages. The type of messages when received by the recipient can be leaked when the adversary can compromise the recipient, or controls the ISP of the recipient. In order to quantify this knowledge of the adversary in our empirical analysis, we normalize the probabilities of all messages being the target as explained in Section 3.5. We report the entropy values in Figure 3.

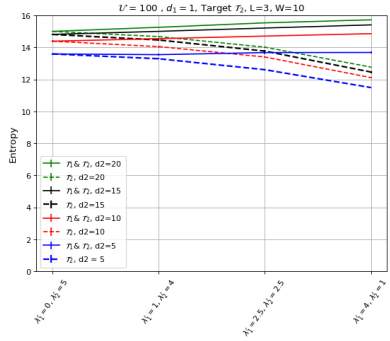
In Figure 3, the solid lines represent the entropy values when the two types of traffic are blended and the dashed line represent the `Solo cases`. The red lines represent the entropy values for 0 corruption, the blue lines represent the adversary who corrupts 10% of all the nodes ($B = 3$) and the black lines represent the scenario $\mathcal{B}$ where the adversary knows the types of traffic of all the input and output messages. Similar to previous experiments, we observe that the entropy decreases as λ_1' (resp. λ_2') decreases when traffic types are not blended; however, when we blend messages from different traffic types, the messages from the second traffic type compensates for the lack of messages from $\mathcal{T}_1$ (resp. $\mathcal{T}_2$). Similar to previous evaluation, the slow traffic benefits slightly more from blending than the fast traffic.

When we consider 10% of the mixnodes are compromised (we choose them by choosing one corrupt mixnode per layer), we observe slight decrease in entropy values for each type of traffic, and in both blended and non-blended scenario (c.f. Figure 3). The black lines, representing adversary $\mathcal{B}$, shows that the entropy is lower when the adversary knows the types of output messages than when the adversary does not know — which was expected, since the adversary has the additional information of the output traffic types. In fact, the entropy values closely match the dashed red lines depicting entropy of single-type traffic. This indicates that, even in the worst-case scenario where the adversary knows the types of traffic of all input and output messages, blending traffic is as effective as dedicating the entire network to each traffic type. We can summarize the insights as follows:

1. Blending traffic is slightly more advantageous for the slow traffic than for the fast traffic.
2. However even for the fast traffic, blending two different types of traffic into the same mixnet infrastructure is more beneficial in terms of anonymity compared to sending them through separate mixnets.
3. On the other hand, anonymity for the slow traffic improves as the difference between the delay parameters increases.



(a) Entropy for messages of type $\mathcal{T}_1$ for $L = 3$, $W = 10$, $B = 0$.

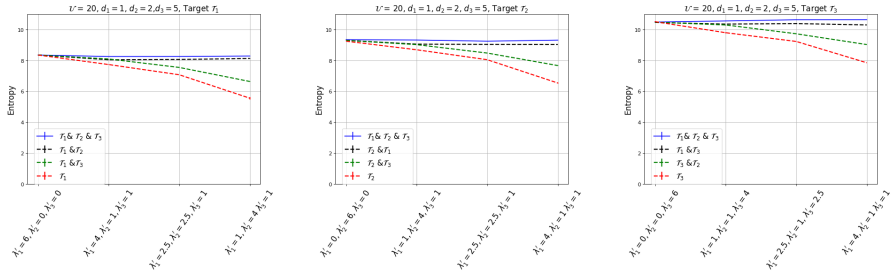


(b) Entropy for messages of type $\mathcal{T}_2$ for $L = 3$, $W = 10$, $B = 0$.

Figure 4: Evaluation of anonymity in terms of entropy for continuous mixnets with $L = 3$, $W = 10$, average delay $d_1 = \frac{1}{\lambda_1} = 1$ for traffic type $\mathcal{T}_1$, and different average delays for traffic type $\mathcal{T}_2$: $d_2 = \frac{1}{\lambda_2} = 5$, $d_2 = 10$, $d_2 = 15$, $d_2 = 20$.

4.5 Evaluation: Different Ratios of Delay Parameters

In the previous sections, we consider two types of traffic which we called *fast* with an average delay $d_1 = 1$ and a second type of traffic $\mathcal{T}_2$ which we called *slow* with an average delay $d_2 = 5$. In this section we want to investigate the impact of these per mix delays ratios on the anonymity of both of the traffic. In figure 4a, we keep the same rate for per-mix delay $d_1 = 1$ for traffic type $\mathcal{T}_1$, and we change the rate delays of traffic $\mathcal{T}_2$ to $d_2 = 10$ (red), $d_2 = 15$ (black) and $d_2 = 20$ (green). We also plot the entropy values for `Solo` case of traffic $\mathcal{T}_1$ in purple dashed line. When the ratios λ_1':λ_2' is large, meaning that the majority of the messages in the network are from type $\mathcal{T}_1$, the entropy values are almost the same for all values of d_2. However when this ratio λ_1':λ_2' declines, meaning that the majority of the message generation in the network are coming from $\mathcal{T}_2$ traffic we see that the best entropy values for target messages from $\mathcal{T}_1$ are when $d_2 = 5$ (blue lines). Having the slow traffic with large per-mix delays compared to the fast traffic does not provide the best anonymity for the fast traffic, however blending these two traffic together does still provide better anonymity than the `Solo` case (purple dashed line). As for the traffic $\mathcal{T}_2$ in 4b, the entropy values are to be expected: when we increase the per-mix delay d_2 we increase the anonymity for the messages from that traffic: the best entropy values are provided when $d_2 = 20$ (the green solid lines). We can summarize the insights as follows:



(a) Entropy for target messages of type $\mathcal{T}_1$ (b) Entropy for target messages of type $\mathcal{T}_2$ (c) Entropy for target messages of type $\mathcal{T}_3$.

Figure 5: Evaluation of anonymity in terms of entropy for continuous mixnets with $L = 1$, $W = 1$, average delay $d_1 = \frac{1}{\lambda_1} = 1$ for traffic type $\mathcal{T}_1$, average delay $d_2 = \frac{1}{\lambda_2} = 2$ for traffic type $\mathcal{T}_2$, and average delay $d_3 = \frac{1}{\lambda_3} = 5$ for traffic type $\mathcal{T}_3$.

1. When the amount of messages from the *fast* traffic is the majority in the network, the delay parameter of the *slow* traffic has less impact on the anonymity of the *fast* traffic.
2. However, when the amount of the fast traffic is not the majority, the delay parameter of the slow traffic has a negative impact the anonymity of the fast traffic. As the difference between the delay parameters increases, anonymity slowly deteriorates.
3. The rate delays of one traffic type impact not only the anonymity of messages within that specific type but also those from other types of traffic.

Remark We consider in our analysis that the adversary observes the mixnet since the first message being sent, and therefore can count how many messages came to the mixnet, how many has left, and how many are still remaining. We argue, however, that changing the starting time of observation will not radically change our results since an adversary observing the network for a long time can infer these information with high confidence.

4.6 More Than Two Types of Traffic

To validate our insights obtained with two types of traffic, we evaluate the impact of blending three types of traffic with a set of small-scale experiments. We vary for different experiments the generation rates of each traffic such that the sum of the three rates of traffic generations $(\lambda'_1 + \lambda'_2 + \lambda'_3)$ is equal to 6

with $\mathcal{U} = 20$, meaning that we generate 120 messages per second, and the total number of messages is equal to 12000. Note that the number of clients as well as the number of messages does not change our insights regarding the blending strategy. In order to compute the probability of an output message being the target input message, given that the adversary knows the type of traffic of the target message when there are three types of traffic, we need to consider all possible values of k_2, which is the number of messages of type $\mathcal{T}_2$ that entered the node. We update Algorithm 2 to compute the probability distribution over all possible k_2 values based on the derivations in section 3.6, and present it in Algorithm 3.

We evaluate the impact of blending three types of traffic in a network with one mixnode: traffic type $\mathcal{T}_1$ with an average delay $d_1 = 1$, traffic type $\mathcal{T}_2$ with an average delay $d_2 = 2$, and traffic type $\mathcal{T}_3$ with an average delay $d_3 = 5$. In Figure 5a, we present entropy values for target messages from traffic type $\mathcal{T}_1$ under different scenarios: when there are messages only from $\mathcal{T}_1$ (red), when there are messages from both $\mathcal{T}_1$ and $\mathcal{T}_2$ (black), when there are messages from $\mathcal{T}_1$ and $\mathcal{T}_3$ (green), and when there are messages from all three types of traffic (blue). Similarly, we repeat this analysis for target messages from $\mathcal{T}_2$ in Figure 5b and for $\mathcal{T}_3$ in Figure 5c.

In all three figures, there is a noticeable increase in entropy values when blending different types of traffic. Specifically, the highest entropy values are observed when messages from all three traffic types are combined.

5 Discussion And Conclusion

5.1 Broader Impact and Future Work

We have showed, through a diverse set of experiments in multiple settings, that the blending traffic types improves anonymity. The degree of this improvement, however, depends on the different average delays of the traffics and their respective generation ratios. In this paper, we assume that the end-to-end latency is set by the application using the mixnet-based system, leaving users with limited control over this aspect. However, envisioning a more user-centric mixnet-based system can consider the possibility of allowing users to choose delays to improve anonymity, as proposed by the authors in [16]. In this paper, the authors let senders specify for each message whether they prefer security or speed and hence end-to-end delays is chosen by the users. In such a scenario, a systematic study of the effects of different generation ratios along with the different average delays on anonymity is needed. Furthermore, we assumed in this paper that the adversary knows the type of traffic of input

Algorithm 3: Probability computation on a single node with three types of traffic.

Result: Updated $\Pr[m_i = m_t | m_t \in \mathcal{T}_1]$.

Initialize:

G, Q, R : arbitrarily expandable Lists ;

$G.append(1)$; $Q.append(0)$;

$k = 0$; $i = 0$;

if *event(receive(m_i))* **then**

$k++$;

for $j \leftarrow 0$ **to** k **do**

for $k2 \leftarrow 0$ **to** k **do**

$$Q[j,k2] = Q[j-1,k2] \cdot \Pr[m_i \in \mathcal{T}_1] + Q[j,k2-1] \cdot \Pr[m_i \in \mathcal{T}_2] + Q[j,k2] \cdot \Pr[m_i \in \mathcal{T}_3] + G[j-1] \cdot \Pr[m_i = m_t];$$

$$G[j,k2] = G[j,k2-1] \cdot \Pr[m_i \in \mathcal{T}_2] + G[j-1,k2] \cdot \Big(\Pr[m_i \in \mathcal{T}_1] - \Pr[m_i = m_t] \Big) + G[j,k2] \cdot \Pr[m_i \in \mathcal{T}_3];$$

$$R[j,k2] = R[j-1,k2] \cdot \Pr[m_i \in \mathcal{T}_1] + R[j,k2-1] \cdot \Pr[m_i \in \mathcal{T}_2] + R[j,k2] \cdot \Pr[m_i \in \mathcal{T}_3];$$

end

end

end

if *event(send(m_i))* **then**

define $\mathsf{denom}(j,k2) = j \cdot \lambda_1 + k2 \cdot \lambda_2 + (k - i - k2j) \cdot \lambda_3$;

$$\Pr[m_i = m_t] = \sum_{j=0}^{k-1} \sum_{k2=0}^{k-1} \frac{\lambda_1}{\mathsf{denom}(j,k2)} \cdot Q[j,k2];$$

for $j \leftarrow 0$ **to** k **do**

for $k2 \leftarrow 0$ **to** k **do**

$$R[j,k2] = \frac{\lambda_1}{\mathsf{denom}(j+1,k2)} \cdot Q[j+1,k2] + \frac{(j+1) \cdot \lambda_1}{\mathsf{denom}(j+1,k2)} \cdot R[j+1,k2] + \frac{(j,k2+1) \cdot \lambda_2}{\mathsf{denom}(j,k2+1)} \cdot R[j,k2+1] + \frac{(k-i-k2-j) \cdot \lambda_3}{\mathsf{denom}(j,k2)} \cdot R[j];$$

$$Q[j,k2] = \frac{j \cdot \lambda_1}{\mathsf{denom}(j+1,k2)} \cdot Q[j+1] + \frac{(k2+1) \cdot \lambda_2}{\mathsf{denom}(j,k2+1)} \cdot Q[j,k2+1] + \frac{(k-i-k2-j) \cdot \lambda_3}{\mathsf{denom}(j,k2)} \cdot Q[j,k2];$$

$$G[j,k2] = \frac{(j+1) \cdot \lambda_1}{\mathsf{denom}(j+1,k2)} \cdot G[j+1,2] + \frac{(k2+1) \cdot \lambda_2}{\mathsf{denom}(j,k2+1)} \cdot G[j,k2+1] + \frac{(k-i-k2-j) \cdot \lambda_3}{\mathsf{denom}(j,k2)} \cdot G[j,k2];$$

end

end

Forward Message (m_i);

$i++$;

end

messages. However, it is not always straightforward for an adversary to deduce the types of messages, particularly when this information is not overtly leaked from client behaviors. An in-depth exploration of different applications that enable adversaries to infer message types is deferred for future research. In addition, future research should also consider more real-world scenarios: while our analysis and insights remain the same irrespective of message volume or the geo-location of the different mixes, exploring the impact of real data in practical settings may yield valuable additional insights. Such analysis will not reverse the relevance of our observations but may reveal nuanced insights that can be crucial in real-world implementations. Finally, due to limitations in simulation constraints, we regrettably had to work with a relatively small number of messages per second. Exploring the impact of blending different traffic types on anonymity in scenarios with a substantial volume of traffic presents an intriguing avenue for future research. Investigating by how much does the anonymity increases with blending traffic in high-volume situations could provide valuable insights for a more comprehensive understanding of the blending traffics.

5.2 Conclusion

We have provided the first quantitative analysis of the anonymity offered by continuous mixnets when multiple different traffic types are blended together. To that end, we provided (i) a novel analytical framework to compute the probabilities connecting the input and output messages of a mixnode when different traffic types with different latency requirements are blended together; (ii) a simulation-based evaluation of anonymity based on the proposed analytical framework considering varying proportions of traffic types, different average delays per traffic type, and diverse network settings. Our evaluations reveal that blending different traffic types through a mixnet enhances anonymity compared to dedicating a different mixnet to each traffic type.

Acknowledgments

This research is partially supported by the Research Council KU Leuven under the grant C24/18/049, by CyberSecurity Research Flanders with reference number VR20192203, and by DARPA FA8750-19-C-0502. Any opinions, findings and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of any of the funding agencies.

References

[1] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. MCMix: Anonymous Messaging via Secure Multiparty Computation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1217–1234, Vancouver, BC, August 2017. USENIX Association.

[2] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. Anoa: A framework for analyzing anonymous communication protocols. In *2013 IEEE 26th Computer Security Foundations Symposium*, pages 163–178. IEEE, 2013.

[3] Iness Ben Guirat and Claudia Diaz. Mixnet optimization methods. *Proceedings on Privacy Enhancing Technologies*, page 456–477, 2022.

[4] Iness Ben Guirat, Devashish Gosain, and Claudia Diaz. MiXiM: Mixnet Design Decisions and Empirical Evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, 2021.

[5] George Danezis. The traffic analysis of continuous-time mixes. In *International Workshop on Privacy Enhancing Technologies*, pages 35–50, Berlin, Heidelberg, 2004. Springer.

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

[7] Debajyoti Das, Easwar Mangipudi, and Aniket Kate. Organ: Organizational anonymity with low latency. *Proceedings on Privacy Enhancing Technologies*, 2022:582–605, 07 2022.

[8] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity Trilemma: Strong anonymity, Low Bandwidth Overhead, Low latency - Choose two. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 108–126, 2018.

[9] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Comprehensive anonymity trilemma: User coordination is not enough. *Proceedings on Privacy Enhancing Technologies*, 2020(3):356–383, 2020.

[10] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Divide and funnel: a scaling technique for mix-networks. Cryptology ePrint Archive, Paper 2021/1685, 2021. `https://eprint.iacr.org/2021/1685`.

[11] Claudia Diaz. *Anonymity and Privacy in Electronic Services*. PhD dissertation, KU leuven, December 2005.

[12] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym Network. `https://nymtech.net/nym-whitepaper.pdf`, February 2021.

[13] Claudia Diaz, Steven J. Murdoch, and Carmela Troncoso. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 184–201, Berlin, Heidelberg, 2010. Springer-Verlag.

[14] Claudia Diaz and Bart Preneel. Taxonomy of mixes and dummy traffic. In *Information Security Management, Education and Privacy*, pages 217–232. Springer, 2004.

[15] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards Measuring Anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 54–68, Berlin, Heidelberg, 2002. Springer-Verlag.

[16] Roger Dingledine, Andrei Serjantov, and Paul Syverson. Blending different latency traffic with alpha-mixing. In *Privacy Enhancing Technologies: 6th International Workshop, PET 2006, Cambridge, UK, June 28-30, 2006, Revised Selected Papers 6*, pages 245–257. Springer, 2006.

[17] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[18] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop- and- Go-MIXes providing probabilistic anonymity in an open system. In David Aucsmith, editor, *Information Hiding*, pages 83–98, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[19] Christiane Kuhn, Martin Beck, Stefan Schiffner, Eduard Jorswieck, and Thorsten Strufe. On privacy notions in anonymous communication. *Proceedings on Privacy Enhancing Technologies*, 2019(2):105–125, 2019.

[20] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle. *Proceedings on Privacy Enhancing Technologies*, 2016(2):115–134, 2016.

[21] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable Messaging System with Cryptographic Privacy. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 759–776, Santa Clara, CA, February 2020. USENIX Association.

[22] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix Anonymity System. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1199–1216, Vancouver, BC, August 2017. USENIX Association.

[23] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 41–53, Berlin, Heidelberg, 2002. Springer-Verlag.

Publication

Traffic Analysis by Adversaries with Partial Visibility

Publication Data

Ben Guirat, I., Diaz, C., Eldefrawy, K., and Zeilberger, H. Traffic Analysis by Adversaries with Partial Visibility. In *28th European Symposium on Research in Computer Security (ESORICS)* (2023), Springer

Contributions

- Principal author.

Traffic Analysis by Adversaries with Partial Visibility

Iness Ben Guirat[1], Claudia Diaz[1], Karim Eldefrawy[2], and Hadas Zeilberger[3]

[1] COSIC, KU Leuven, Belgium
[2] SRI, USA
[3] YALE, USA

Abstract. Mixnets are a fundamental privacy enhancing technology in the context of anonymous communication that have been extensively studied in terms of a Global passive Adversary (GPA). However a more realistic adversary that captures only a portion of the network have not yet been studied. We call these adversaries "Adversaries with Partial Visibility (APV)". In this paper we provide a framework that models the mixnet-based system and captures different types of Adversaries with Partial Visibility. Each adversary is modeled based on their goals, prior knowledge, and capabilities. We then use this model to perform traffic analysis using the Metropolis-Hastings algorithm in conjunction with a Bayesian inference engine. To the best of our knowledge this is the first time such an adversary is explicitly defined and studied, despite this adversary model being championed as more realistic than global passive adversaries in prior work. We highlight that our framework is flexible and able to encompass a broad range of different adversaries. Naturally, our model also captures the classical global passive adversary (GPA) as a special case.

1 Introduction

The notion of a mixnet was first invented by Chaum [2]. Mixnets were designed specifically to resist an adversary with a global view who can observe all inputs and outputs in the network. Resisting such adversaries is achieved via the mixing strategy where each mix does not output messages immediately upon receiving them. Instead, a mix stores messages for some amount of time before sending them to either another mix or to a recipient. This technique hides correlations between the inputs and outputs which therefore makes mixnets resilient against a Global Passive Adversary (GPA). Currently, a substantial body of literature exists using simulation-based evaluation in order to quantify anonymity under such a powerful adversary for generic mixnets [1, 14].

However, adversaries with such full visibility are of questionable practical utility, since in practice even very powerful adversaries may have blind spots in their network coverage [17]. The literature currently lacks methods to model and analyze anonymity properties with respect to an adversary who has partial visibility of the network, such an adversary (such as Iran or Russia surveillance operations) that can only control mixes within their jurisdiction. One challenge in evaluating anonymity under such an adversary is that their advantages in inferring information about messages in a network vary greatly depending on which portions of the network they have visibility over. Deriving, analytically, the probability of an event when considering such an adversary requires conditional probabilities based on the different assumptions of the portions of the network that they can or cannot see. Even a simple network results in an explosion in complexity from the derivation of such probabilities.

To address this problem, we first provide a general and flexible matrix-based mixnet model that captures different mixnet configurations, then we model adversaries based on their goals, priors, and capabilities. By performing this, we retrieve two sets of data: an *Observation* $\mathcal{O}$ and a *Hidden State* $\mathcal{HS}$. The $\mathcal{O}$ is the part of the trace the adversary has visibility over and the $\mathcal{HS}$ is the part of the trace the adversary lacks visibility over. Finally, we use this modeling to perform mixnet traffic analysis under an APV. The main strength of this framework is that it is able to capture a broad range of adversaries. We develop and implement a Bayesian inference engine which takes an APV and a network configuration as input. The inference engine then uses the Metropolis Hastings to estimate the distribution over possible traces.

In summary, our main contributions are as follows:

- A matrix-based mixnet model that is able to capture different mixnet configuration such as different topologies and mixing strategies.
- A model for an *Adversary with Partial Visibility (APV)* that can be captured by the matrix-based mixnet model. This model can be used in conjunction with Bayesian inference techniques to perform traffic analysis attacks. To the best of our knowledge such a general adversary model has not been studied in mixnets before.
- A traffic analysis framework that uses the Metropolis-Hastings algorithm to evaluate anonymity under different APVs.

2 Motivation and Related Work

In this section we present our motivation and related work. First, we discuss Adversaries with Partial Visibility in the context of mixnets. Second, we discuss

our decision to choose the Metropolis–Hastings algorithm for traffic analysis.

Adversaries with Partial Visibility: Since Chaum's seminal work on untraceable email [2], there has been considerable research exploring and understanding the design space of mixnets [5, 10, 13, 15]. However, most of this literature assumes a very strong notion of the adversary, called a *Global Passive Adversary* (GPA), which observes the entire network. In [17], Syverson argues that a GPA is unrealistic and does not reflect the real anonymity of a system. Starting from adversary models that are not reasonable in practice has contributed to the lack of wide-spread adoption of anonymous communication systems. Syverson argues that in security analysis, most adversaries are subject to some constraints on their capabilities such as having a partial view of a network [17]. An adversary's ability to execute a successful attack is dependent on the adversary's available resources and their effectiveness. In other words, the measure of an adversary's ability to succeed must include the resources it will take them to learn enough information to link relations.

In [8], Gallagher et al. have argued for the need for new adversary ontologies that could better model real-world adversaries by outlining the limitations of the adversaries that are commonly used in academic literature, such as Global Passive Adversary. A GPA is unrealistic in real world scenarios where the mixnet system spans multiple administrative domains (e.g., different autonomous systems) in several countries. In fact, considering such an unnecessarily strong adversary may hinder exploring and curtail the development of mixnet designs that can scale better and provide adequate and meaningful privacy in the real world [14].

Traffic Analysis in Mixnets: Multiple variants of traffic analysis attacks in mixnets do exist in the literature [3, 11, 12, 16]. However the focus has been on the edges of mixnets where messages are sent and received and therefore the adversary does traffic analysis on the patterns of these messages in order to correlate for example a sender to a receiver. These attacks do not capture an adversary that is also able to corrupt or monitor a portion of the network in order to de-anonymize the sender and receiver. Traffic analysis by an Adversary with Partial Visibility (APV) over the network is a hard task as it requires consideration of prior knowledge of the adversary as well as the different assumptions for the parts of the network that the adversary does not have visibility over. In [6], the authors argue that at the heart of traffic analysis lies an inference problem. Therefore applying Bayesian techniques provides a sound framework on which to build attacks and algorithms to estimate different quantities. Their main contribution is introducing the application of Bayesian inference to traffic analysis. The Metropolis–Hastings algorithm is a well-known

method for obtaining a sequence of random samples where direct sampling is difficult [9]. In order to estimate a function of the mapping between input and output messages of the network, they build an inference engine that samples from the distribution of network states that a set of given observations allow. However, this model only models global passive adversaries [6].

One of the key contributions of our work is to extend this bayesian-based framework in conjuction with Metropolis Hastings in order to analyse Adversaries with Partial Visibility. This encompasses a much wider set of adversaries than the Global Passive Adversary considered in prior work [6, 18].

3 Adversary Model

In this paper we provide a method that solves the problem of traffic analysis under an Adversary with Partial Visibility (APV). In a nutshell the problem is presented as follows: The adversary APV has a prior knowledge about the mixnet-based system. An APV observes the network over some time interval t and produces an observation $\mathcal{O}$. Given the prior knowledge on the system model as well as the observation $\mathcal{O}$, the adversary tries to infer the probability of certain events that happened either partially or entirely outside of her visibility during the time t. We call the parts of the network trace that the adversary does not have visibility over a Hidden State $\mathcal{HS}$. Table 1 provides a handy reference for all the notations used in this paper. One of our objectives for solving this problem is to make the adversary model as general as possible in order to accommodate different types of adversaries. We therefore start by modeling the three major aspects of any adversary: (i) Goal, (ii) Prior and (iii) Capability:

Goal: The adversary chooses a goal. For example, an adversary's goal can be to infer the probability of a sender s_i communicating with receiver r_j ($Pr[s_i \rightarrow r_j]$), or correlating a specific input i_i to an output o_j ($Pr[i_i \rightarrow o_j]$). Note that in the first example, the adversary needs to track all messages from s_i to r_j, however, in the second example the adversary is only interested in correlating i_i with o_j.

Prior: Depending on the system model, different adversaries can have different sets of prior knowledge. One could imagine a system that have different paths lengths for each message or different constraints on users in choosing the route (for example users in certain geographical locations are more likely to route their messages through nodes in adjacent areas). We use $\mathcal{C}$ for the system constraints. This type of information is encoded in the prior knowledge of the adversary.
To be more formal we assume that an abstract system consists of an Observation $\mathcal{O}$ and a Hidden State $\mathcal{HS}$. We therefore assign a joint probability given the

Notation	Interpretation
Adversary	
$\mathcal{A}$	Adversary
APV	Adversary with Partial Visibility
$\mathcal{O}$	Observation
$\mathcal{HS}$	Hidden State
$\mathcal{C}$	Constraints
Messages	
μ	mix
M_t	Set of messages that are inside μ during t
$\mathcal{M}_I$	Incoming messages to a mix
$\mathcal{M}_O$	Outgoing messages of mix
Matrices	
P_m	Path of the message m
$\mathcal{V}$	Set of Vertices in a path
$\mathcal{E}$	Set of edges in a path
$\mathcal{V}_I$	Set of mix input vertices
$\mathcal{V}_O$	Set of mix output edges
$\mathcal{E}_\mu$	Set of edges in the subgraph associated with the inner-mix mappings of mix μ
$\mathcal{E}_{I,O}$	Set of edges in subgraph associated with inter-entity connections between $\mathcal{ET}_O$ and $\mathcal{ET}_I$
$v_{i,I}(\mu_j)$	ith input edge of jth mix
$v_{i,O}(\mu_j))$	ith output edge of jth mix
$\mathcal{GM}$	Set of ghost messages
Metropolis–Hastings	
$\mathcal{TR}$	Trace
$\mathcal{TR}_p$	Proposed Trace
$\mathcal{TR}_c$	Current Trace
$\mathcal{TR}_G$	Trace of the Ground Truth
Q	Transitioning function

Table 1: Summary of notation for Publication 4.

constraints $\mathcal{C}$, $Pr[\mathcal{HS}, \mathcal{O} \mid \mathcal{C}]$. We further elaborate how to incorporate the joint probability in our framework in section 5.

Capability: In order to achieve its goals, the adversary does traffic analysis by making use of its *Compromising* and/or *Monitoring* capabilities. Monitoring the network means that the adversary is monitoring the connections between different physical entities of the network either throughout some portion of the

network or throughout the entire network in the case of a GPA. A physical entity in the network can be either a mix μ or a user u. We call this an *inter-entity connection*. An ISP is one example of an adversary that can see inter-entity connections and use this data for traffic analysis. The adversary is also able to *compromise*. Compromising a mix means this adversary has the capability of seeing inside a mix. One real-world example of an adversary that can see inner-mix mappings is one that can add mixes to a volunteer-based network [7].

4 A Matrix-based Model of a Mix Network

In this section, we describe our model for a matrix-based mix network. As explained in the previous section, the APV is trying to estimate the probability distribution of the Hidden State $\mathcal{HS}$ given their Observation $\mathcal{O}$ and the prior knowledge about the system constraints $\mathcal{C}$. We therefore need to provide a model for the mixnet system that encompasses the adversary's constraints $\mathcal{C}$ and partitions the message trace (defined next) into an Observation $\mathcal{O}$ and Hidden State $\mathcal{HS}$.

4.1 Trace Graph: A graph derived from the message trace

A trace, which is the full set of messages traversing the network during a time t, is represented by $\mathcal{HS}$ and$\mathcal{O}$. We now describe how to derive a graph $(\mathcal{V}, \mathcal{E})$ from a trace, where $\mathcal{V}$ is the set of vertices in the graph and $\mathcal{E}$ is the set of edges. The purpose of this graph is to split each message path into segments so that we can differentiate between the parts of each path that an adversary can see and the parts that they cannot see. We start by defining a *path*, and then show how to model each path as a graph.

Definition 4.1 (Path)**.** A **path**, $P = (s_i, \mu_1, .., \mu_n, r_j) = \{e_1, ..e_{n+2}\}$, is an ordered set of entities, users and mixes, that describes the path of the message from the sender s_i to the receiver r_j (n: length of the message path).

If P_i is the ith path, then as it goes through mix μ_j, we will denote its input vertex in this mix as $v_{i,I}(\mu_j)$ and its output vertex as $v_{i,O}(\mu_j)$. We will denote the vertex associated with the path's sender as $(v_i(s))$ and the vertex associated with the path's receiver as $(v_i(r))$.

Definition 4.2 (Path Graph)**.** Let $P_i = (s, \mu_1, .., \mu_n, r)$ be a path. A path graph is a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. A vertex represents (i) the sender of the message $(v_i(s))$ of the corresponding path, (ii) the input/output of the message to/from the different mixes, $v_{i,I}(\mu_j)$, $v_{i,O}(\mu_j)$ and (iii) the recipient of the message $(v_i(r))$. An edge is the connection between two vertices of the

network, which can be between two entities (either a mix or a user) or between an input vertices and an output vertices of each mix.

Note that a vertex in the network is not just a physical point of entry of the message. It is rather a representation of time in the graph. For example, if a message m arrives to mix μ at $t_1 = 1$ and a message m_2 arrives to μ at time $t_2 = 2$, we assign different vertices in the mix to the different messages. Figure 1b shows a simple example of modeling a portion of the path and its relationship to the mix by vertices and edges.

Definition 4.3 (Trace)**.** Let M be a set of messages sent through the mixnet. For message $m \in M$, let P_m be the path of m. A **trace** $\mathcal{TR} = \{P_m : m \in M\}$ is the set of paths of all messages sent through the network during time interval t, in other words, a trace is the full set of *paths* that occur in a network during a time interval t.

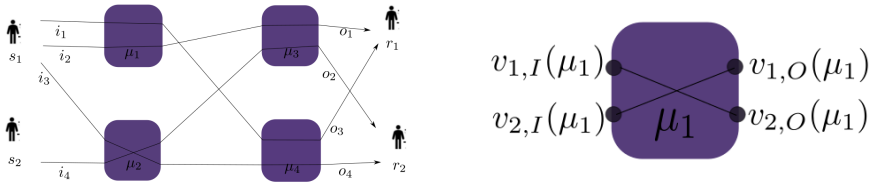
Definition 4.4 (Trace Graph)**.** Let M be the set of messages sent across the network during time interval t and let $\mathcal{TR} = \{P_m : m \in M\}$ be a trace of a network in that time interval. Let G_P be the Path graph of path P. The Trace Graph, $G_{TR} = \{G_P : P \in \mathcal{TR}\}$ is the set of Path Graphs associated with paths in $\mathcal{TR}$.

We model the *Trace Graph* as a set of adjacency matrices.

Definition 4.5 (Adjacency Matrix)**.** Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Then an adjacency matrix Mx is a matrix with $|\mathcal{V}|$ columns and $|\mathcal{V}|$ rows, where row i and column i are both labeled by vertex $v_i \in \mathcal{V}$, for $i \in [1, |\mathcal{V}|]$. Element (i, j) of Mx is 1 if the edge $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise.

Each adjacency matrix will be associated with a subgraph of the Trace Graph, G_{TR}. We identify two types of subgraphs in the Trace Graph. The first type is an inner-mix subgraph, defined by the vertices and edges associated with each mix. We call the adjacency matrix of this subgraph an *inner-mix matrix*. This matrix represents the relationship of the input messages to a mix to the output messages of the same mix.

Definition 4.6 (Inner-mix matrix)**.** Let TG be a trace graph, let μ be a mix, and let $\mathcal{V}_I$ be the incoming vertices of messages associated with μ and $\mathcal{V}_O$ be the outgoing vertices of messages associated with μ, as described in Definition 4.2. For each message going through μ, there is an edge (v, w) between a vertex in $v \in \mathcal{V}_I$ and a vertex $w \in \mathcal{V}_O$, as defined by the path of that message. Let $\mathcal{E}_\mu$ be the set of such edges. Then an inner-mix matrix is the adjacency matrix of the graph $(\mathcal{V}_I \cup \mathcal{V}_O, \mathcal{E}_\mu)$.



(a) An example of a network trace

(b) Mix μ_1 modeled by vertices and edges.

Figure 1: An example of network trace $\mathcal{TR}$ and the representation of mix μ_1 by vertices and edges.

We also model *inter-entity connections* as a set of adjacency matrices. An inter-entity matrix is a matrix that represents the connections between entities that have messages going between them.

Definition 4.7 (Inter-entity matrix)**.** Let $\mathcal{V}_O$ be the set of outgoing vertices of a subset of the network entities that are able to send messages to subset of entities with vertices $\mathcal{V}_I$. Let $\mathcal{E}_{I,O}$ be the collection of all edges that connect the vertices of $\mathcal{V}_O$ with the vertices $\mathcal{V}_I$. Then an inter-entity matrix is the adjacency matrix of the graph $(V_O \cup V_I, \mathcal{E}_{I,O})$.

The number of inter-entity matrices in our model vary depending on the network configuration. For example, if we have a free-route network where each mix can send a message to any mix in the network then we can have one inter-entity matrix. However, if the system's topology is stratified then mixes are arranged in a fixed number of layers and can only communicate with mixes in the subsequent layer. In this case we have an inter-entity matrix that connects every two layers (eg $\mathcal{E}_O$ is associated with layer i and $\mathcal{E}_I$ is associated with layer $i+1$).

4.2 Matrices in two sets

As given in Section 4.1, we model the entire Trace Graph by a list of matrices. There exists two types of matrices (i) inner-mix matrices that represent the "inside" of the mix and (ii) the inter-entities matrices that represent the connections between two sets of entities (either a mix or a user). The motivation behind this modeling is two-fold:

- Using this model we are able to trace any message by simply tracing the elements with value 1 that connect the edges in the different matrices. Figure 2 shows the matrices that are the representation of the trace depicted in Figure 1a. We can see in Figure 2 how by simply following

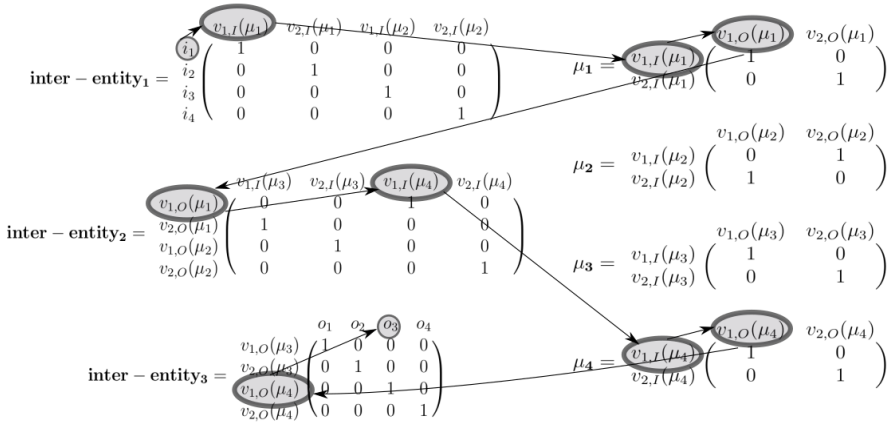


Figure 2: Modeling the trace in Figure 1a by matrices

$$\mathcal{O} = \begin{array}{c} \\ v_{1,O}(\mu_1) \\ v_{2,O}(\mu_1) \end{array} \begin{array}{c} \begin{array}{cc} v_{1,I}(\mu_3) & v_{2,I}(\mu_3) \end{array} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \end{array} \quad \mathcal{HS} = \begin{array}{c} \\ v_{1,O}(\mu_2) \\ v_{2,O}(\mu_2) \end{array} \begin{array}{c} \begin{array}{cc} v_{1,I}(\mu_4) & v_{2,I}(\mu_4) \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$$

Figure 3: Inter-entity matrix split into $\mathcal{O}$ and $\mathcal{HS}$.

the connected edges in the matrices we are able to correlate the input i_1 to the output o_3.

- Accommodating the adversary in our model. The set of matrices represents the full trace of the network, so in order to include the adversary in this model, we need to split the matrices into two sets: matrices that are part of the observation $\mathcal{O}$ and matrices that are part of the Hidden State $\mathcal{HS}$. For example, if an adversary APV is only corrupting a mix μ_1, we put the corresponding matrix in the observation and the rest of the matrices in the hidden state. However, for the inter-entity matrices, depending on which portions of the network the adversary monitors, the same matrix has to be split to $\mathcal{O}$ and $\mathcal{HS}$. For example, if APV controls the connections between μ_1 and μ_3 the *inter-entity*$_2$ in Figure 2 is split in two as shown in Figure 3.

We call the *Ground Truth* the trace of messages that actually happens in reality, i.e. the trace that the adversary wants to learn information about. The adversary can see some part of the Ground Truth in its Observation $\mathcal{O}$ and

proceeds to conduct the traffic analysis by traversing through the space of the different possible traces in the Hidden State $\mathcal{HS}$.

5 Traffic Analysis of Mixnets

The goal of the APV, having an observation $\mathcal{O}$ and a prior knowledge about the network $\mathcal{C}$, is to estimate the probability distribution over the possible hidden states in order to achieve its goal, for example linking senders to receivers or input messages to output messages. In other words, the APV tries to estimate the target distribution $Pr[\mathcal{HS} \mid \mathcal{O}, \mathcal{C}]$ using the Metropolis–Hastings method.

5.1 Markov Chain Monte Carlo (MCMC)

The Metropolis–Hastings method works by sampling states by conducting a random walk across a state space. In our context, a state is the Hidden State, $\mathcal{HS}$, of the trace. The Metropolis-Hastings algorithm requires that we define a transition function that allows us to propose a new Hidden State ($\mathcal{HS}_p$) for every Hidden State ($\mathcal{HS}_c$) that we encounter in our random walk. Next we compute the following ratio, α_{next_move} in order to decide whether or not to accept the $\mathcal{HS}_p$.

$$\alpha_{next\_move} = \frac{Pr[\mathcal{HS}_c|\mathcal{O}, \mathcal{C}] * Q(\mathcal{HS}_p|\mathcal{HS}_c)}{Pr[\mathcal{HS}_p \mid \mathcal{O}, \mathcal{C}] * Q[\mathcal{HS}_c|\mathcal{HS}_p]} \tag{1}$$

Note that $\mathcal{Q}$ is the transition function that provides the probability of moving from a current state $\mathcal{HS}_c$ to a proposed state $\mathcal{HS}_p$. Further details on how to compute $\mathcal{Q}$ are provided in Algorithm 4. We recall from Section 4 that in the set of matrices $\mathcal{MX}$ associated with a trace $\mathcal{TR}$, a matrix in $\mathcal{MX}$ is either an *inter-entities matrix* or an *inner-mix matrix*. Recall also that we divide this set of matrices into two sets; an Observation $\mathcal{O}$ and a Hidden State $\mathcal{HS}$.

The transition function we use to move from current Hidden State $\mathcal{HS}_c$ to a proposed Hidden State, $\mathcal{HS}_p$ works as follows. We start by choosing a matrix $Mx \in \mathcal{HS}$ at random, and then two columns (which corresponds to vertices in the network) in Mx at random to swap. For inner-mix matrices, this corresponds to the following: let $\mathcal{M}_t$ be a set of messages in a mix with *inner-mix mapping* $F_{\mu_j} : \mathcal{M}_I \to \mathcal{M}_O$ during time t. Let $v_{1,I}, v_{2,I}(\mu_j) \in \mathcal{M}_I$, and let $v_{1,O}, v_{2,O}(\mu_j) \in \mathcal{M}_O$ such that $F_{\mu_j}(v_{1,I}) = v_{1,O}, F_{\mu_j}(v_{2,I}) = v_{2,O}$. Then swapping two columns in Mx assigns to M_t a new *inner-mix mapping* F'_μ, where $F'_\mu(v_{1,I}(\mu_j)) = v_{2,O}(\mu_j)$ and $F'_\mu(v_{2,I}(\mu_j)) = v_{1,O}(\mu_j)$ and $F'_\mu(x) = F_\mu(x)$ for all x such that $x \notin \{v_{1,I}, v_{2,I}\}$.

For inter-entity connections, the transition function corresponds to the following: Let $\mathcal{ET}_O$ and $\mathcal{ET}_I$ be two sets of entities and let $\mathcal{M}_t$ be a set of messages that are sent from entities in $\mathcal{ET}_O$ to entities in $\mathcal{ET}_I$ during time t. Let $F_e : \mathcal{ET}_O \rightarrow \mathcal{ET}_I$ be the inter-entity connection. Let $\mathcal{V}_O$ be the vertices associated with outgoing packets from $\mathcal{ET}_O$ and let $\mathcal{V}_I$ be the vertices associated with the incoming packets of $\mathcal{ET}_I$. Let $v_{1,O}, v_{2,O} \in \mathcal{V}_O$, and let $v_{1,I}, v_{2,I} \in \mathcal{V}_I$ such that $F_e(v_{1,O}) = v_{1,I}$ and $F_e(v_{2,O}) = v_{2,I}$. Then swapping two columns in Mx assigns to $\mathcal{M}_t$ a new *inter-entity connection* F'_e, where $F'_e(v_{1,O}) = v_{2,I}$, $F'_e(v_{2,O}) = v_{1,I}$ and $F'_e(x) = F_e(x)$ for all x such that $x \notin \{v_{1,O}, v_{2,O}\}$.

5.2 Computing the Probability of a State based on Priors

Depending on the system, the adversary may have prior knowledge on the system, which will be accounted for in the estimation of the target probability. We model the system by a list of constraints $\mathcal{C}$. Any system can have a variety of constraints, such as paths lengths [18], or user constraints in choosing nodes based on geo-locations, or having a biased distribution for routing (as in the example of Nym [7]).

To consider user constraints, the adversary, having a proposed trace $\mathcal{TR}$, computes $Pr[\mathcal{TR} \mid \mathcal{C}]$ which is the probability of the trace given its prior knowledge on the system model $\mathcal{C}$. As described in Section 4.2, a trace $\mathcal{TR}$ is composed of an Observation $\mathcal{O}$ and $\mathcal{HS}$, therefore:

$$Pr[\mathcal{TR} \mid \mathcal{C}] = Pr[\mathcal{HS}, \mathcal{O} \mid \mathcal{C}] \tag{2}$$

We note however that the adversary is trying to estimate $Pr[\mathcal{HS} \mid \mathcal{O}, \mathcal{C}]$ (Equation 1). We therefore apply Bayes rule:

$$Pr[\mathcal{HS} \mid \mathcal{O}, \mathcal{C}] = \frac{Pr[\mathcal{HS}, \mathcal{O} \mid \mathcal{C}]}{Pr[\mathcal{O} \mid \mathcal{C}]} \tag{3}$$

Thus, we can rewrite $ratio_{next\_move}$ as follows:

$$\begin{aligned}
\alpha_{next_move} &= \frac{Pr[\mathcal{HS}_c \mid \mathcal{O}, \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p \mid \mathcal{O}, \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} \\
&= \frac{Pr[\mathcal{HS}_c, \mathcal{O} \mid \mathcal{C}] / Pr[\mathcal{O} \mid \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p, \mathcal{O} \mid \mathcal{C}] / Pr[\mathcal{O} \mid \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} \\
&= \frac{Pr[\mathcal{HS}_c, \mathcal{O} \mid \mathcal{C}] * Q(\mathcal{HS}_p \mid \mathcal{HS}_c)}{Pr[\mathcal{HS}_p, \mathcal{O} \mid \mathcal{C}] * Q[\mathcal{HS}_c \mid \mathcal{HS}_p]} \\
&= \frac{Pr[\mathcal{TR}_c \mid \mathcal{C}] * Q(\mathcal{TR}_p \mid \mathcal{TR}_c)}{Pr[\mathcal{TR}_p \mid \mathcal{C}] * Q[\mathcal{TR}_c \mid \mathcal{TR}_p]}
\end{aligned} \tag{4}$$

Computing the α_{next_move} is therefore reduced to computing the probabilities of the different traces $\mathcal{TR}$ given the constraints $\mathcal{C}$. Assuming path selection is independent (we further elaborate on this point in Section 6). Because a trace is a set of message paths, the adversary computes:

$$Pr[\mathcal{TR}|\mathcal{C}] = \prod_{P \in Tr} Pr[P|\mathcal{C}] \tag{5}$$

Based on this derivation, we execute the Metropolis-Hastings algorithm by generating a sequence of different samples (traces) starting from a random trace.

5.3 Metropolis-Hastings Algorithm

We now put everything together to present the concrete algorithm used to gain the target distribution. We describe the algorithm here, and its pseudo-code is included in Algorithm 4.

$Q(\mathcal{TR}_p|\mathcal{TR}_c)$ is the transition function, that is the probability of moving from one state $\mathcal{TR}_c$ to another state $\mathcal{TR}_p$. In our model, this translates to choosing a matrix of $Mx \in \mathcal{MX}$ and two of its columns (c_1, c_2) to permute which yields $\mathcal{TR}_p$. Therefore the transition function is symmetric and equal to $\frac{1}{|\mathcal{MX}|*|\mathcal{V}|*|\mathcal{V}|-1}$, where $|\mathcal{V}|$ is the number of columns in Mx_c. As noted in [4], the values of $Q(\mathcal{TR}_p|\mathcal{TR}_c)$ are not key to the correctness of the sampling, however different transition functions may speed the convergence to a the stationary distribution.

5.4 Constraints in sampling

Depending on the network configuration as well as the adversary capabilities, some sample states are not possible therefore are thrown away in the Metropolis Hastings algorithm. In order for our model to be accurate we incorporate two categories of these impossible states:

Ghost Messages: In prior work using the Metropolis-Hasting algorithm [4], the authors acknowledge that one of the limitation of their model is assuming that the adversary starts observing the network when the all mixes are empty. We argue however that this is not realistic. Instead the adversary starts observing the network when there are already messages inside the mixes and those message contribute in *hiding* an item of interest. We model this scenario by *ghost messages*. Not to be confused with dummy or cover traffic, ghost messages gm are an output message whose inputs are outside the observation of the adversary and therefore are not items of interests but they do play a

Algorithm 4: Metropolis–Hastings Algorithm

Result: Computing $Pr[s_i \rightarrow r_j]$
Input: $\mathcal{TR} \leftarrow \{\mathcal{O}, \mathcal{HS}\}$, APV, **burn in,** n_{MH}
Initialize:
$s_i \leftarrow pick\_sender$;
$r_j \leftarrow pick\_receiver$;
$reality \leftarrow \text{check}(s_i \rightarrow r_j)$;
$P_t \leftarrow \text{compute}(Pr[TR_c])$;
$I \leftarrow 0$;
$n_{samples} \leftarrow 0$;
while $n_{samples} \leq n_{MH}$ **do**
  $MX_c \leftarrow pick\_matrice(HS_c)$;
  $MX_p \leftarrow permute(MX_c)$;
  $HS_p \leftarrow \mathcal{MX}_p = \mathcal{MX} \setminus \{Mx_c\} \cup \{Mx_p\}$;
  $TR_p \leftarrow \mathcal{O}, HS_p$;
  violations $\leftarrow$ check_constraints(TR_p) ;
  if *Not violations* **then**
    $P_{t+1} \leftarrow \text{compute}(Pr[TR_p])$;
    $\alpha \leftarrow min\left[1, \frac{P_{t+1}}{P_t}\right]$;
    $r \leftarrow \mathcal{U}(0,1)$;
    if $\alpha \geq r$ **then**
      accept(TR_p);
      $n_{samples}++$;
      if *check*$(s_i \rightarrow r_j)$ **then**
        $I++$;
end
$Pr[s_i \rightarrow r_j] = \frac{I}{n_{samples}}$;
return: $\{Pr[s_i \rightarrow r_j], reality\}$

role in hiding an item of interest. We therefore add vertices to the subgraph associated with the matrix.

Routing: When an adversary is monitoring or compromising a mix, not only do the have knowledge about the input and output messages of that mix but also the destination and source mixes of those messages. We model this by a list of rules. Any sampled state that is the result of a violations of these rules will be thrown away.

6 System Model

In this section, we showcase modeling a small example of a mixnet-based system of 9 mixes in order to perform traffic analysis. We choose a stratified topology where the 9 mixes are arranged in 3 layers as shown in Figure 6. Each mix

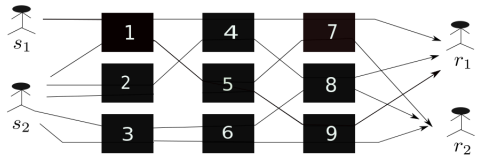


Figure 4: Mixnet-based system Model.

is assigned a layer, and messages are sent from layer l_i to layer l_{i+1}. In our experiments, we use a *threshold mix* [2] that buffers messages in the queue until the threshold parameter T is reached. At that point the mix permutes the T messages, and then outputs all messages in a random order. Finally all messages are source-routed. We emphasize that our model is not limited to this type or size of network . We argue that the matrix-based model is able to accommodate any type of mixes and/or network. However, depending on the type of the network, Equation 5 should be adapted accordingly. Next we show how to compute the probability of a trace in a network of certain constraints $\mathcal{C}$.

6.1 Probability of a Trace

Recall from equation 5, that a trace $\mathcal{TR}$ is a collection of all its paths and therefore the probability of a specific trace is the product of the probabilities of different paths.

$$Pr[\mathcal{TR}|\mathcal{C}] = \prod_{P_i \in \mathcal{TR}} Pr[P_i] \tag{6}$$

In addition, as explained in Section 4, a message path P_i is an ordered set of entities that describe the path of the message from a sender s_i to a receiver r_i. The routing of the message through the different mixes, as well as choosing a receiver for each message is probabilistic. Let's denote $Pr[\mu]$ the probability of the mix μ being chosen in the path P_i and $Pr[r_i]$ the probability of the receiver r_i is part of P_i, meaning sender s_i chose receiver r_i to send a message to. The probability of the path P_i is then:

$$Pr[P_i|\mathcal{C}] = \left[\prod_{\mu \in P_i} Pr[\mu])\right] Pr[r_i] \tag{7}$$

The above equation holds because we consider source routing, meaning that each user chooses the path of each message among the set of mixes and receivers. We leave other types of routing outside the scope of this work. Putting Equation 6 and Equation 7 together, the probability of a trace is therefore:

$$Pr[\mathcal{TR}|\mathcal{C}] = \prod_{P_i \in \mathcal{TR}} (\prod_{\mu \in P} Pr[\mu]) Pr[r_i] \tag{8}$$

Considering our stratified topology system model of l layers and $N(l_i)$ mixes in each layer l_i, we now proceed to compute the probability of each path. We abuse notation slightly and also consider μ_{j,l_i} as the event that the sender s_i chooses mix $\mu_{j,i}$ as the jth mix in layer l_i, as well as r_i as the event the sender s_i in path i chooses the receiver r_i as the receiver. We first consider that each sender knows all the public parameters of the network such as all the mixes in the network as well as the number of mixes in each layer. The sender then chooses each mix as well as the receiver of each message uniformly.

$$\begin{aligned} Pr[P_i] &= \left[\prod_{i=1}^{l} Pr[\mu_{j,l_i}]\right] Pr[r_i] \\ &= \left[\prod_{i=1}^{l} \frac{1}{N(l_i)}\right] Pr[r_i] \end{aligned} \tag{9}$$

Next we show how we compute the probability of a trace of a network when mixes are not chosen uniformly.

6.1.1 Probability of a trace: Mixes are not chosen Uniformly

Depending in the systems requirements, mixes can be chosen according to a certain weights assigned to them.We now consider the case of mixes not chosen uniformly, but according to an assigned weight. Mix weights can reflect geographic approximation, bandwidth, trust etc. Let's denote by $w(\mu_{j,l_i})$ the probability of the mix μ_{j,l_i} being chosen among the set of mixes in layer l_i. The probability of each path becomes:

$$Pr[P_i] = \left[\prod_{i=1}^{l} w(\mu_{j_i,i})\right] Pr[r_i] \tag{10}$$

We can also model the case when each sender s_i only knows a subset of the mixes in each layer l_j ($\mathcal{S}(s_i, l_j)$).

$$Pr[P_i|\mathcal{S}(s_i, l_j)] = \left[\prod_{j=1}^{l} Pr[\mu_{j,l_j}|\mathcal{S}(s_i, l_j))]\right] Pr[r_i] \tag{11}$$

To compute $\texttt{Pr}[\mu_{j_i,i}|\mathcal{S}(s_i,l_j)]$, we need to divide the weight of mix $\mu_{j_i,i}$ by the sum of the weights of the all the mixes in layer l_i *that sender* s_i *knows*.

$$Pr[\mu_{j,l_j}|\mathcal{S}(s_i,l_j)] = \frac{w(\mu_{j_i,i})}{w(\mathcal{S}(s_i,l_i))} \tag{12}$$

Therefore:

$$Pr[P_i|\mathcal{S}(s_i,l_j)] = \left[\prod_{j=1}^{l} \frac{w(\mu_{j_i,i})}{w(\mathcal{S}(s_i,l_i))}\right] Pr[r_i] \tag{13}$$

Recipient probability The recipient probability $Pr[r_i]$ captures any prior knowledge of the adversary about the relationship between a specific sender and a specific receiver, for example if the adversary has a prior knowledge about *friendship* graphs of certain senders (i.e sender s_1 is twice as likely to send a message to receiver r_1 than to a receiver r_2).

7 Empirical Evaluation

In this section we first explain how we assess the effectiveness of our model and then show the results of inferring probabilities of trace-related events under two adversaries of different capabilities.

7.1 Determining the Accuracy of our model

Our framework is effective if it reaches the target distribution. To evaluate the effectiveness of our model of an APV we need to assess the closeness of the samples given by our model to a target distribution based on the ground truth across many rounds of messages in a mixnet.

Recall from Section 5, that we first start by generating the ground truth trace $\mathcal{TR}_{\mathcal{GT}}$ and the adversary specifies the trace related event that they want to infer probability of. We refer to the success or failure of this event in the ground truth as a *goal* and we model it as a predicate function $P : \{\mathcal{TR}_{GT}\} \rightarrow \{0,1\}$, which takes the ground truth trace$\mathcal{TR}_{GT}$ as input and outputs 1 if the occurrence happens in $\mathcal{TR}_{GT}$ and 0 otherwise. For example, if the goal of the adversary is to infer the probability of sender s_1 sending a message to receiver r_1 during its window of observation, then we associate to this goal a predicate P, where $P(TR) = 1$ if s_1 did send a message to r_1 in the trace TR and 0 otherwise.

Then we translate this ground truth trace into a set of matrices. Depending on what portions of the network the adversary observes, we divide this set of matrices into two sets: Hidden State $\mathcal{HS}$ and $\mathcal{O}$. In order to avoid starting

from a high probability state, we do a randomized uniformly chosen number of permutations on the different matrices that are part of $\mathcal{HS}$ and finally we give these two sets of matrices to the adversary as well as the list of constraints.

The adversary then execute the Metropolis–Hastings algorithm on the trace. At the end of one run of the Metropolis–Hastings algorithm, the adversary assign a probability p_{tr} to the event of interest. We refer to the probability, p_{tr}, as a goal and the output of $P(\mathcal{TR}_{GT})$ as an occurrence. We call an occurrence positive when $P(\mathcal{TR}_G) = 1$ and we call an occurrence negative when $P(\mathcal{TR}_G) = 0$.

In order to assess the accuracy of our model we need to generalize beyond a single trace. To do this, we run the Metropolis Hastings algorithm on N_{tr} number of traces for the same adversary each time using a different ground truth trace. After each run of the algorithm, we record the probability p_{tr_i} that the adversary assigns a given event to its goal, and we record the occurrence $P(TR_{GT})$. Once we have our full data set $(p_{tr_i}, occurrences)$ tuples, we compare the average inferred probability to the confidence score on the occurrences. We use the Wilson score interval [20], as it has been shown to be the most accurate [19] of binomial confidence intervals. The Wilson Score takes number of successes and number of failures as inputs, and outputs the probability of success along with the 95% confidence interval. We consider our model to be effective if the sampled probability lies within the confidence interval. Therefore, the average ability of the adversary to correctly infer the probability of the specified event across many different events can be taken into account.

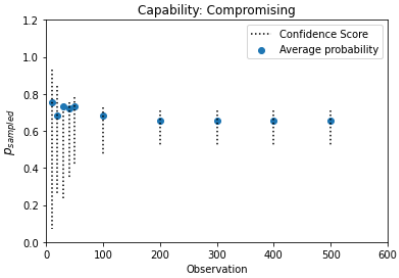
7.2 Capability: Compromising

We start by analysing an adversary who has compromising capability. The goal of this adversary is to infer *whether at-least one of the messages of a sender s_i arrived to a receiver r_j.* This adversary tries to link the receiver to senders that route a message through one of the compromised mixes. Therefore we choose an adversary who is compromising one entry mix and one exit mix. We evaluate this adversary for up to 500 observations.

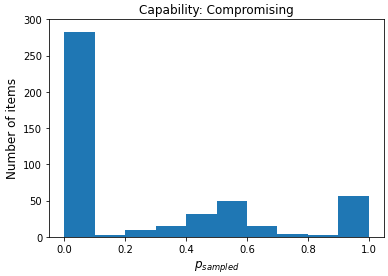
We can see in Figure 5a that when there are only a few observations, the confidence interval is high, however, we show that our model eventually converges into the the target distribution where the confidence interval is within tight bounds as the model is guessing with 95% confidence the correct probability. Note however, that we are showing the average sampled probability of the different events. The probability of each trace-related event may differ per different observation. In Figure 5b we show the histogram of the number of experiments per bin. The idea behind this binning is to analyse beyond the average probability inferred by the adversary for the different observations. After collecting all the sampled probabilities where each probability corresponds

to one trace, we store them in different bins depending on their values. Each bin corresponds to an interval of probabilities and contains the number of times the adversary inferred that probability.

We can see that for this adversary, the bin that corresponds to a probability between $[0, 0.1]$ contains the largest number of items, meaning that for most observations the adversary guessed between $[0, 0.1]$ due to lack of information. The second largest bin is the one that corresponds to the events having a probability between $[0.9, 1]$. Those events are likely to be the messages that went through both the compromised mixes (and so the APV had more information), and finally around 25% of the total events the adversary infers a probability between 0.2 and 0.7. This would intuitively match to the case where messages went through a single compromised mix that the APV could watch, but not both.



(a) Accuracy



(b) Bins

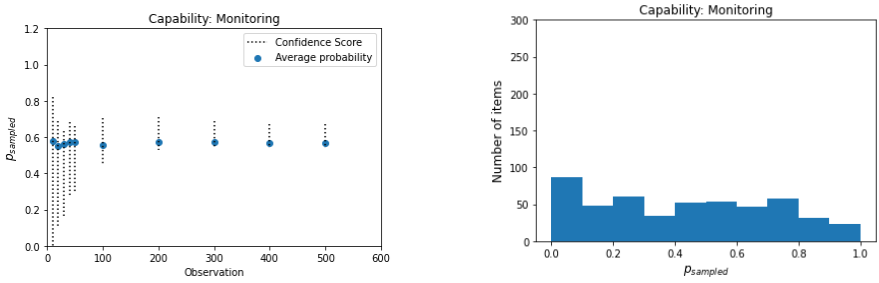
Figure 5: Capability of APV_1: Compromising 2 mixes.

7.3 Capability: Monitoring

Similar to the previous experiment, the goal of this adversary is to infer *whether at-least one of the messages of a sender s_i arrived to a receiver r_j*. The adversary is monitoring the inter-entity matrices, and so the adversary is what is usually called a Global Passive Adversary in mixnet literature. The adversary in this case is not compromising the mixes, and therefore all permutations of inner-mix mapping are possible.

We can see in Figure 6a that the adversary captured by our model is able to infer probabilities correctly with enough number of observations. As opposed to the APV, the GPA can connect s_i to a receiver r_i with a higher confidence. Figure 6b shows the histogram of the different probabilities estimated by this adversary. Unlike the previous adversary, we can see that the number of items

inside the different bins is more balanced between probabilities, meaning that this adversary does not sample states with a very high probability compared to other states with low probability. This confirms our understanding with the modeled GPA. In the first adversary, we can see that there are traces with high probabilities (a less balanced histogram) that correspond to messages being routed via at least one compromised node. We note however that the average probability inferred about the different events is similar in this scenario even though the first adversary is more realistic as it compromises only 2 mixes as opposed to the second adversary that monitors all mixes in the network.



(a) Accuracy (b) Bins

Figure 6: Capability of APV_2: Monitoring all inter-entity matrices (GPA).

We emphasize that the main goal of the paper is not to present the specific results of de-anonymization, but rather to demonstrate the possibility of going beyond the traditional Global Passive Adversary (GPA) model in mixnets. In addition, we show that our model is also able to capture this GPA as a special case.

However, this flexible framework comes with one main drawback which is the Metropolis Hastings algorithm is computationally expensive, requiring a large number of state samples to be drawn in order to reach from the target distribution. Additionally, if the adversary is relatively weak, it may be more efficient to rely on the adversary's prior knowledge and analytically infer trace-related events. Nevertheless, for adversaries with a reasonable level of visibility into the network, we consider the computational cost of the algorithm to be acceptable, especially given the lack of alternative methods for studying such adversaries.

7.4 Performance evaluation

Our Metropolis-Hastings sampler is composed by 2363 LOC of Python and run on an Intel(R) Core(TM) i9-9920X with 3.50GHz CPU and 132 GB RAM. As explained in 5.1, in order to compute α_{next_move} we need to compute the probability of every state. Depending on the adversary's knowledge, the network size and the number of messages per observation, the computation time might increase. For our simulated scenarios, it took about 6 hours for the full analysis. While this may seem expensive, our implementation is not optimised for size, memory usage or running time and we argue that more optimized implementations will outperform it.

8 Conclusion

To the best of our knowledge, this is the first work considering a model of an adversary whose capabilities can be modeled by the partial view of the network. In this paper we present and analyze traffic analysis by Adversaries with Partial Visibility (APV), who have a visibility over some portion of the network. We put forth a mathematical model of a mix network and use this model with the conjunction of Metropolis-Hastings to build a Bayesian inference engine that models how an adversary infer probabilities about trace-specific events. Finally, we analyzed the effectiveness of our model under different simulated adversaries.

The critiques made of previous work on mixnets by Syverson [17] are to a large extent correct, as even powerful nation-state adversaries may have only partial visibility. However, this does not mean that mixnets cannot be analyzed in terms of such adversaries, as our work shows. Our work would allow the more realistic practical engineering of privacy-enhancing technologies, especially as real-world mixnets like Nym are deployed in countries with partial adversaries such as Iran [7].

Nonetheless, the work is limited. While we studied compromised mixes, we assumed the mixnet adversaries that were compromised were honestly following the protocol. Thus, an aspect that needs to be studied is actively malicious mixes as well as users. More importantly further work should systematically study different adversaries, each having different visibility over multiples mix networks with different parameters and design decisions.

Acknowledgement

We thank the anonymous reviewers for their comments and insightful feedback. This research is partially supported by the Research Council KU Leuven under the grant C24/18/049, by CyberSecurity Research Flanders with reference number VR20192203, and by DARPA FA8750-19-C-0502. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the funding agencies.

References

[1] Iness Ben Guirat, Devashish Gosain, and Claudia Diaz. MiXiM: Mixnet Design Decisions and Empirical Evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, 2021.

[2] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[3] George Danezis. Statistical disclosure attacks. In *IFIP International Information Security Conference*, pages 421–426. Springer, 2003.

[4] George Danezis. The traffic analysis of continuous-time mixes. In *International Workshop on Privacy Enhancing Technologies*, pages 35–50, Berlin, Heidelberg, 2004. Springer.

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

[6] George Danezis and Carmela Troncoso. Vida: How to use Bayesian inference to de-anonymize persistent communications. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 56–72. Springer, 2009.

[7] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The nym network, 2021.

[8] Kevin Gallagher, Diogo Barradas, and Nuno Santos. Rethinking realistic adversaries for anonymous communication systems. In *Free and Open Communications on the Internet, FOCI'23*, pages 81–87, 2023.

[9] W Keith Hastings. *Monte Carlo sampling methods using Markov chains and their applications*. Oxford University Press, 1970.

[10] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[11] Dogan Kesdogan, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Breaking anonymity by learning a unique minimum hitting set. In *International Computer Science Symposium in Russia*, pages 299–309. Springer, 2009.

[12] Dogan Kesdogan and Lexi Pimenidis. The hitting set attack on anonymity protocols. In *International Workshop on Information Hiding*, pages 326–339. Springer, 2004.

[13] Albert Kwon, David Lu, and Srinivas Devadas. XRD: Scalable Messaging System with Cryptographic Privacy. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 759–776, Santa Clara, CA, February 2020. USENIX Association.

[14] 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*, pages 39–44, 2021.

[15] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix Anonymity System. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1199–1216, Vancouver, BC, August 2017. USENIX Association.

[16] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Designing Privacy Enhancing Technologies*, pages 10–29. Springer, 2001.

[17] Paul Syverson. Why i'm not an entropist. In *International Workshop on Security Protocols*, pages 213–230. Springer, 2009.

[18] Carmela Troncoso and George Danezis. The bayesian traffic analysis of mix networks. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, page 369–379, New York, NY, USA, 2009. Association for Computing Machinery.

[19] Sean Wallis. Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods. *Journal of Quantitative Linguistics*, 20(3):178–208, 2013.

[20] Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Bibliography

[1] Ben Guirat, I., Das, D., and Diaz, C. Blending Different Latency Traffic with Beta Mixing. *Proceedings on Privacy Enhancing Technologies* (2024).

[2] Ben Guirat, I., and Diaz, C. Mixnet optimization methods. *Proceedings on Privacy Enhancing Technologies* (2022), 456–477.

[3] Ben Guirat, I., Diaz, C., Eldefrawy, K., and Zeilberger, H. Traffic Analysis by Adversaries with Partial Visibility. In *28th European Symposium on Research in Computer Security (ESORICS)* (2023), Springer.

[4] Ben Guirat, I., Gosain, D., and Diaz, C. MiXiM: A general purpose simulator for mixnet. *13th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2020)* (2020).

[5] Ben Guirat, I., Gosain, D., and Diaz, C. MiXiM: Mixnet Design Decisions and Empirical Evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society* (2021).

Curriculum

Iness graduated with an engineering degree in Networks and Telecommunications from INSAT in Tunisia. She worked as a research engineer at INRIA de Paris in the area of Formal Verification. In September 2019, she started her PhD at COSIC, KU Leuven.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
COSIC
Kasteelpark Arenberg 10, bus 2452
B-3001 Leuven
iness.benguirat@esat.kuleuven.be
http://www.esat.kuleuven.be/cosic