Stopping Silent Sneaks: Defending against Malicious Mixes with Topological Engineering

Xinshu Ma
University of Edinburgh
United Kingdom
x.ma@ed.ac.uk

Florentin Rochet
University of Namur
Belgium
florentin.rochet@unamur.be

Tariq Elahi
University of Edinburgh
United Kingdom
t.elahi@ed.ac.uk

ABSTRACT

Mixnets provide strong meta-data privacy and recent academic research and industrial projects have made strides in making them more secure, performant, and scalable. In this paper, we focus our work on stratified Mixnets, a popular design with real-world adoption. We identify and measure significant impacts of practical aspects such as: relay sampling and topology placement, network churn, and risks due to real-world usage patterns. We show that, due to the lack of incorporating these aspects in design decisions, Mixnets of this type are far more susceptible to user deanonymization than expected. In order to reason about and resolve these issues, we model Mixnets as a three-stage "Sample-Placement-Forward" pipeline and develop tools to analyze and evaluate design decisions. To address the identified gaps and weaknesses we propose Bow-Tie, a design that mitigates user deanonymization through a novel adaption of Tor's guard design with an engineered guard layer and client guard-logic for stratified mixnets. We show that Bow-Tie has significantly higher user anonymity in the dynamic setting, where the Mixnet is used over a period of time, and is no worse in the static setting, where the user only sends a single message. We show the necessity of both the guard layer and client guard-logic in tandem as well as their individual effect when incorporated into other reference designs. We develop and implement two tools, 1) a mixnet topology generator (Mixnet-Topology-Generator (MTG)) and 2) a path simulator and security evaluator (routesim) that takes into account temporal dynamics and user behavior, to assist our analysis and empirical data collection. These tools are designed to help Mixnet designers assess the security and performance impact of their design decisions.

CCS CONCEPTS

• **Security and privacy** → *Network security*; **Pseudonymity, anonymity and untraceability**.

KEYWORDS

Anonymous communication network, mixnets, network construction

ACSAC '22, December 5–9, 2022, Austin, TX, USA

ACM ISBN 978-1-4503-9759-9/22/12…$15.00
https://doi.org/10.1145/3564625.3567996

ACM Reference Format:
Xinshu Ma, Florentin Rochet, and Tariq Elahi. 2022. Stopping Silent Sneaks: Defending against Malicious Mixes with Topological Engineering. In *Annual Computer Security Applications Conference (ACSAC '22), December 5–9, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3564625.3567996

1 INTRODUCTION

Since the "Five Eyes" mass surveillance disclosures by Snowden high-lighted real-world adversaries' pervasive and global nature, we observe a greater community focus on strong meta-data privacy to protect and improve communication protocols on the Internet [26]. Tor, with ≈ 8 million daily users [43], provides limited protection against a global adversary and traffic analysis attacks [35, 47, 55]. Thus, there is a resurgent interest in mix networks (Mixnets) [8]—once considered impractical to deploy—with many recent proposals from academia [7, 36, 38, 39, 51, 68, 70] and industry [18] that have strong security guarantees and improved performance at scale.

The security of many known designs, such as Vuvuzela [70], Karaoke [39], Loopix [51], and Nym [17] rely on the *anytrust* assumption where at least one server in the user's path must be honest. In other words, security comes from distributing trust across many relay operators. Practical real-world designs distribute trust *and* provision network resources by drawing from third-parties, such as volunteers or for-profit participants, on which the network applies light (e.g., Tor's path selection IP restrictions) or no constraints. These third-parties can be malicious and it is critical that the mixnet design resist their influence. In Mixnet literature, the security analysis typically considers active attacks like traffic analysis [1], (n-1) [59], and Denial-of-Service (DoS) [5]. In addition, users can also be deanonymised by passive adversaries whenever a message traverses a path composed entirely of adversarial relays.

However, the literature typically takes for granted real-world issues such as *network configuration and routing, network churn, and risk due to real-world usage patterns.* In this paper, we consider the impact of these practical concerns and investigate designs that strengthen the anytrust assumption while minimizing performance degradation. We present the first thorough analysis of continuous-time stratified Mixnet designs, and the implications on the security of typical users against realistic resource-bounded strategic adversaries in the network. Our *temporal* analysis, where we model an adversary cumulatively deanonymizing users over time, shows that in the state-of-the-art reference designs close to 100% users are expected to use a fully malicious route in about one week of email activity over the Mixnet. Overall, the adversary is able to deanonymize a significant portion of network traffic running a realistic amount of bandwidth and quantity of nodes. This implies

that the anytrust assumption is not easy to maintain in real Mixnet deployments.

Contributions.

(1) We propose Bow-Tie, a novel practical and efficient Mixnet design that mititgates the over-time client exposure to adversarial mixes and strengthen the anytrust assumption. We realize it by adapting and re-engineering the concept of guards from Tor [64] to stratified Mixnets.

(2) We present an empirical security analysis of the stratified Mixnet against reasonably realistic adversaries from the metrics of i) fully-compromised traffic fraction and ii) time-to-first compromise. We show how these results relate to a newly deployed Mix network and how it could be significantly improved.

(3) We develop the `routesim` simulator, a tool that can calculate a user's expected deanonymization probability over time, given a configured network topology and communication patterns. `routesim` may be used to shed light on the security impact of various design choices, in Bow-tie and other designs as well.

2 BACKGROUND AND MOTIVATION

Mixnets are a fundamental type of anonymous communication system, composed of a set of Mixnodes that provide sender and sender-recipient anonymity by reordering messages in addition to transforming them cryptographically, enabling message untraceability.

Unfortunately, early Mixnets have practical disadvantages such as high latency, poor scalability, and high-performance overhead that hinder their real-world deployments. Recent academic research [7, 36, 39, 51, 51, 68, 70] has made progress in designing Mixnets for anonymous communications with developed scalability and sustainable communication/computation overhead, or provable security. These developments have found their way into the industry, with the foundation of a startup company—Nym [18], whose goal is to create a sustainable anonymous communication network based on the Loopix continuous-time mix design [51] through monetary incentive schemes.

Network topology. Mixnets can be arranged in many topologies. Mesh, cascade, and stratified are some of the most common. In this paper, we focus on the stratified topology [13] due to the evidence that it is both as, or more, secure and performant as the other two [19, 22]. In a stratified topology, the network is constructed from several 'layers'. Each Mixnode is placed in a single layer, and each layer can only communicate with the previous and next ones. Generally, layers are equally sized for performance reasons, although this is not a strict requirement. At the last layer, the messages are delivered to their intended destination (or wherever the user's inbox is hosted).

Path selection/routing. Messages are forwarded through a Mixnet by going through a Mixnode in each layer. This multi-hop path through the network provides the sender and sender-recipient anonymity property. It is therefore critical that the route through the network is not biased or otherwise manipulated by an adversary. Most Mixnet designs route messages by 'bandwidth weight'. That is, the probability of selecting a Mixnode in layer $i + 1$ is proportional to the proportion of its bandwidth to the sum of all Mixnodes bandwidths in that layer. An alternative is to route packets by choosing uniformly at random. We experiment with both approaches in this work and show that uniform selection is inadequate for performance and can be marginally better or worse from a security perspective, depending on the adversary resource endowment.

Continuous-time mixing. Various mixing strategies have been proposed in the literature. Timed, threshold, pool, and continuous-time are the main types. We focus on continuous-time mixing in this paper since it has emerged as a good trade-off between security and performance. In continuous-time mixing, each message is independently delayed at each mix on its path. To offer some level of security against timing attacks, the delay is drawn from an exponential distribution because of its memoryless property. Indeed, Loopix, and by extension Nym, use this mixing strategy, providing real-world relevance.

Anytrust assumption. Many of these systems, Loopix included, rely on the anytrust assumption: as long as there is one honest Mixnode in a path, then the user's message cannot be fully compromised. However, we show that this assumption breaks quickly, and for every users, as soon as one considers temporal aspects in the Mixnet usage. Our work considers this problem when designing Mixnet topologies, and as a consequence, significantly strengthen how realistic this assumption is for the users.

3 THREAT MODEL

Adversary Resources. In general, the adversary has a fixed bandwidth budget to operate/corrupt Mixnodes, and is able to observe all internal states of controlled mixes and may passively observe all network traffic. In Section 6.2.3, we additionally allow the adversary the ability to corrupt honest mixes of their choice during the operational phase of the mixnet. The adversary may locally drop, inject, or delay network traffic, allowing node-targeting Denial-of-Services (DoS) capabilities. An indiscriminate sustained DoS attack on the majority (or all) of honest nodes is out of scope, as is the global active attacker. We analyze the impact of a limited DoS attack on our design in Section 7.3.

More abstractly, we assume the adversary has a certain fixed amount of network resources at their disposal. In this paper we focus on bandwidth and relays, however, financial assets, reputation, or some other scarce resource could be swapped in, since their function is the same; limiting the adversary's control to only a fraction of the paths through the network.

When deciding on its resource allocation, we allow the adversary to take advantage of and influence the network configuration and/or path selection algorithms to maximize the probability of their presence on user paths. These adversarial choices are not observable.

Adversary Goals. The adversary's *goal* is to maximize end-to-end compromised path rates by stealthily causing the mixnet configuration step to optimally place the malicious mixes into the mixnet layers in a way that maximizes their ability to passively deanonymise users.

4 BOW-TIE DESIGN

We propose a new design, Bow-Tie, a three-step pipeline (Figure 1) to configure and use the stratified Mixnet. In Bow-Tie, we discretize

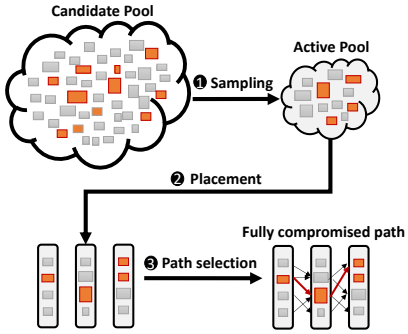


Figure 1: Three-steps basic pipeline when configuring Bow-Tie.

time into periodic epochs, where the network can be reconfigured and clients updated with the latest topology, such that the offline nodes are replaced with active ones considering network churn and the network throughput is maintained in a timely manner. Furthermore, to reflect real-world resource availability, Mixnode bandwidth are heterogeneous and the Mixnet is tunable in order to adjust the network size to suit the volume of incoming traffic. Note that bandwidth is self-announced by the mixnodes. We assume an honest-but-curious *Configuration Server* (CS) that periodically (re)configures the network, ensuring node sampling and placement is correct. It is common for anonymous communication systems to depend on a trusted party for efficiency reasons, such as Directory Authorities (DA) in Tor. Note that a malicious CS might collude with the adversary and enable malicious Mixnodes to be sampled and placed in the network. However, we believe that it is a reasonable assumption in practice because even with the collusion, the CS's deviations from honest behavior will eventually be detected by all participants over time, since the probability distribution of node selection will deviate from expectation. We aim to remove the honest-but-curious CS assumption as future work.

4.1 Bow-Tie Characteristics

Bow-Tie builds Mixnet topologies around the following important criteria.

Mitigating Client Enumeration. In general, users will fall victim to full path compromise the more (or longer) they use the system. Eventually, all users will have at least one of their messages traverse a fully compromised path (see Section 6). This problem is also referred to as client enumeration, where the adversary observes at least one message from every single user of the system. One successful strategy is to limit the exposure of clients to all nodes in the network by restricting the paths clients select. Tor realizes this strategy with its *Guard Design*, which has undergone several refinements since its initial proposal [42, 53, 56, 72]—that (1) Ensures quality of service, and (2) Limits the size of the set of guards[1] the client is exposed to. Bow-tie introduces a novel guard design that uses restricted topologies and client-side logic specifically targeting mixnet integration. The design is supported by empirical results (in Section 6).

[1]Relays in the guard layer.

Unlike Tor, where guards must be placed in the first layer, in Mixnets we have more freedom to choose in which layer to place guards. For a client building routes of length L within a Mixnet, there is a subtle Performance-Security trade-off in choosing either the first node as the guard (more performance) versus choosing one of the middle positions (more secure in specific settings). Bow-Tie adopts guards in the middle position, and the security metrics we explore in Section 6 are independent of the position choice. Appendix A covers a discussion shedding light on the subtleties.

Accommodating Network Churn. Another fundamental issue in real-world mixnet deployments is Network churn, which is a typical and natural phenomenon in volunteer-resourced networks. One of its effects is to increase the clients' exposure to potentially malicious guards [24]. Similar to Tor, the client is required to prefer using an older guard—until they go offline—before touching a new guard. Thus, the more unstable the guards are, the more guards a client will touch, which implies a higher risk of choosing a malicious guard. Therefore, putting the most stable Mixnodes into the guard layer ensures that the guard list of each client grows at a slower pace.

Good Performance & Low Cost Bow-Tie also considers the performance of generated network topologies. In the stratified Mixnet, the transmission bottleneck comes from a layer with the minimum bandwidth. To mitigate this, we model the placement of Mixnodes into a Bin-packing problem, which improves the network performance by constraining that the total bandwidth of each layer is approximately equal. The results show that Bow-Tie strikes a good balance between anonymity and performance (in Section 6.2.4) and runs in an efficient manner.

4.2 Bow-Tie Detail Description

Steps to create and maintain a Bow-Tie Mixnet are depicted in Figure 2 and detailed next.

4.2.1 Mixnet Initialization. The bandwidth of the candidate pool, P_{bw}, is the sum of all the available relays' bandwidths. A predetermined sampling fraction, h, of P_{bw} is the total bandwidth of the active pool from which the generated Mixnet is populated. Each layer accounts for $\frac{1}{l} \times h$ of P_{bw} in a l-layer Mixnet. We consider the case $l = 3$.

(1) *Initialize the Guard Layer.* The CS initializes the guard layer in the first epoch, $i = 0$, by sampling a total of $\frac{1}{3}h \times P_{bw}$ weighted by bandwidth from the candidate pool. The rational is to ensure that the guard layer has $\frac{1}{3}$ of the overall active network bandwidth with the remaining to be distributed evenly across the remaining 2, i.e. $l - 1$, layers. The guard layer is initialized before the other layers to ensure a high likelihood of fast and fewer mixnodes to be the guards.

(2) *Initialize the Guard Set.* The guard set G consists of three subsets: *Active Guard (AG)*, *Backup Guard (BG)*, and *Down Guard (DG)*. All nodes in the initialized guard layer are elements of *AG*. The CS then samples an additional tolerance fraction τ[2] from the candidate pool by bandwidth as *BG*. *DG* is empty at this stage. The rational behind these sets is to minimize client exposure by remembering which nodes were used as guards, even if they go

[2]The value of τ is defined as $\tau = c \times$ churn rate. In this paper, we set $c = 1$.

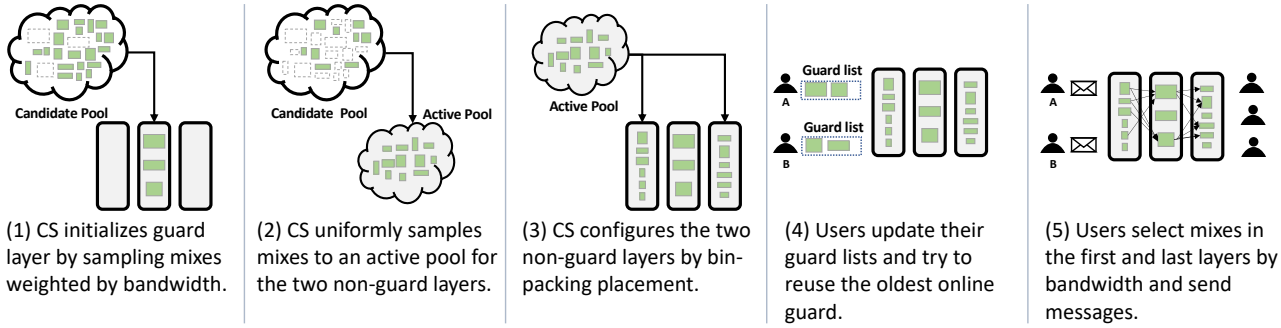


Figure 2: High-level overview of Bow-Tie for Mixnet initialization and message routing. Green rectangles indicate that malicious mixes are indistinguishable.

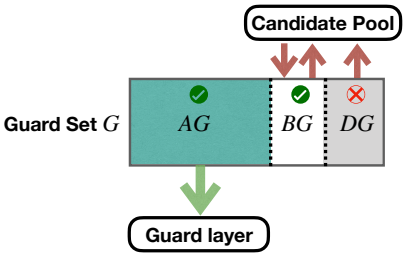


Figure 3: Guard Set composition and interactions.

offline for a period. That is, clients can revert to a previously used but offline guard whenever it appears online again.

(3) *Initialize non-Guard Layers.* Next, the CS uniformly samples a total of $\frac{2}{3}h \times P_{bw}$ Mixnodes from the candidate pool and places them using a bin-packing approach with the constraint that each layer has a similar amount of bandwidth. We convert the placement problem to *one-dimensional bin packing problem (1BPP)* [57], where the objective is to pack all items into a minimum number of bins while the total size of any bin is not larger than the given capacity c. Thus the capacity is set to be slightly larger (ϵ in Algorithm 1) than the $\frac{1}{3}h \times P_{bw}$ bandwidth for each layer as it is difficult to aggregate several indivisible entities to a precise cumulated bandwidth value.

4.2.2 Mixnet Maintenance. Clients need to learn about new Mixnodes, and offline Mixnodes need to be be removed from the active pool. Moreover, specific maintenance for the guard layer is required at each epoch based on the bandwidth and stability of mixnodes. Finally, to ensure all mixnodes in the candidate pool may contribute, at the end of each epoch, a new placement step is executed for non-guards layers.

(1) *Stability Tracking.* To track the stability of each Mixnode, we use the metrics Weighted Mean Time Between Failure (WMTBF) as also used by Tor [65]. Briefly, online/offline states are represented by $1/-1$ respectively in a discretized time interval. The weights of these values are adjusted in proportion to their age from the current epoch. The rational is to discount epochs' values in proportion to their age such that very old epoch values would not significantly influence the WMTBF result.

(2) *Guard Set Maintenance.* For subsequent epochs $i > 0$, the CS checks online/offline status of all nodes in G and updates their stability information. Offline nodes within AG and BG are moved to DG the rest remain where they were.

In addition, CS checks if the bandwidth of $AG \cup BG$ is within the minimum threshold T_{low} and maximum threshold T_{high}. In the case it is greater than T_{high}, nodes in BG that have never have been selected into the guard layer are dropped according to the ascending order of bandwidth×stability[3]. This continues until the bandwidth is $\leq T_{high}$ or all eligible nodes in BG have been evicted. In the case of bandwidth being lower than T_{low}, the CS introduces fresh nodes from the remaining candidate pool by bandwidth×stability to BG. Note that the CS tracks the online/offline status of each node and obtains their stability values; this scheme is detailed later. By introducing new guard nodes or dropping unstable and slow ones, the CS maintains the whole guard set with high stability and sufficient capacity (Figure 3). Please refer to Algorithm 2 in Appendix E.

(3) *Guard Layer Maintenance.* Once the G set is updated, the new AG set is generated by inheriting online guards from the old AG and guards back online from the previous epoch's DG set. To minimize the number of guards users are exposed to, the CS records the number of epochs of active operation as t_{AG}, for all Mixnodes in G, and selects the most stable ones based on their WMTBF value. If AG still does not meet the minimum bandwidth threshold, the CS samples some nodes from BG by bandwidth×stability to AG. In the end, all nodes in AG are placed into the guard layer.

(4) *Non-guard layers Maintenance.* The non-guard layers are refreshed through the same procedure as initialization. Please refer to Section 4.2.1.

4.2.3 Mixnet Routing. Once the network has been constructed, the Mixnet is ready for use.

(1) *Client-side guard logic.* When a user first uses the Mixnet they sample a defined number of Mixnodes, proportional to their bandwidth, belonging to the guard layer and adds these to their guard list. The user's guard list will grow over time. To limit its growth and reduce the user's exposure to malicious Mixnodes in the guard layer, the user's client only adds a new guard if all existing guards in the list are offline. Whenever a new path is required,

[3]The stability values of nodes are evaluated by WMTBF and normalized to $0-1$ scale.

the client tries to reuse guards from oldest to most recent ((4) in Figure 2).

(2) *Path selection.* With an online guard chosen, the user selects Mixnodes in the first and last layer weighted by bandwidth and sends the message through this fresh route ((5) in Figure 2). Bow-Tie uses Bandwidth-weighted path selection since it has better performance and security than the alternative, random path selection, which does not help in protecting clients and incurs a significant performance cost (for details refer to Appendix C).

4.3 Bandwidth Discussion

In our design, we use bandwidth as the sampling criterion since it is well established in the literature and in real-world deployments [65]. However, other attributes can also be used [30, 31, 34]. As for how the bandwidths of the nodes are determined, the simplest way would be for the nodes to advertise their bandwidth, however in such case, adversarial nodes could advertise false information. There exist previously proposed bandwidth measurement systems [2, 32, 33, 62, 67] that have improved the security, accuracy, and efficiency of estimating capacity in Tor network. A similar approach may be adopted here, however the exact solution to this problem is out of the scope for this present paper.

5 METHODOLOGY

We now describe the security metrics, reference algorithms, and adversary model used in our evaluation. Since we consider a realistically constrained and strategic adversary, we identify the optimal adversarial resource allocation strategy that will be employed in our evaluations.

5.1 Security Metrics

We wish to evaluate how well a mixnet design is able to resist compromised mixes (as defined in Section 3). We use the following metrics:

(1) *Time to first compromise*: The expected time it takes until a user has their first message traverse a fully compromised path. This is a dynamic metric since it is affected by usage patterns and is useful to reason about user behavior.

(2) *Compromised fraction of paths*: The expected fraction of total paths in the network topology that are fully compromised (i.e. composed entirely of the adversarial relays). This is a static metric since it is not affected by usage patterns.

(3) *Guessing entropy*: We also consider an *active external* adversary for the scenario where she targets a specific message sent from a particular user. We wonder how many Mixnodes on average she needs to strategically compromise until she can fully observe the complete route of this message, and this metric is called guessing entropy [45, 54]. In particular, this metric can be interpreted as a worst-case adversarial resource endowment to guarantee deanonymizing a given single-message target.

These measures are not only helpful for the mixnet designer or operator but also meaningful for users wishing to know "How secure am I if I use the system?"

5.2 Reference Algorithms

We will empirically evaluate our Mixnet construction algorithm on a statistically significant number of generated topologies, and compare Bow-Tie to three *reference* construction methods: *BwRand*, *RandRand*, *RandBP*, described as follows.

(1) *BwRand*: the CS samples Mixnodes from the active pool with the probability proportional to their bandwidth and places these Mixnodes into random layers with uniform probability. This is a good proxy for the Nym Mixnet design. Indeed, Nym expects to sample nodes based on their stake value, which is expected to correlate with bandwidth in their reward system (i.e., staking to nodes proportional to their true bandwidth maximizes the profit).

(2) *RandRand*: CS samples the active pool uniformly at random and places the Mixnodes into a layer uniformly at random.

(3) *RandBP*: CS samples the active pool uniformly at random and assigns Mixnodes into each layer with the Bin-packing placement algorithm.

5.3 Adversary Modeling

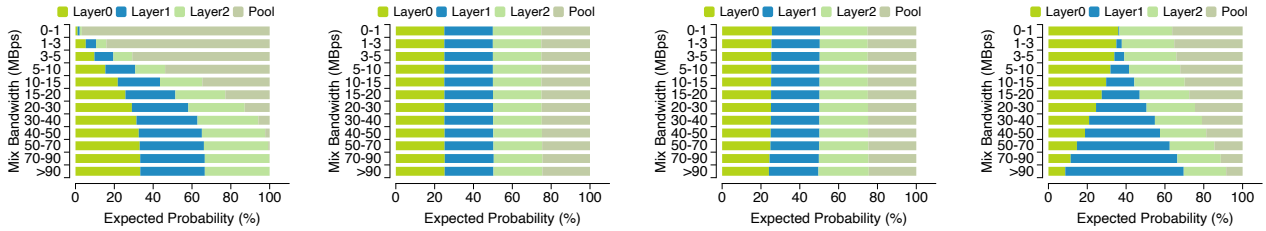
In our simulation, we consider one adversary who wants to deanonymize messages by optimizing the use of Mixnodes and bandwidth resources. We model the adversary bounded by two elements: the number of Mixnodes available to the adversary m, and the bandwidth available for each node b_i^m, $i \in [1, m]$. An adversary is assumed to control a certain fraction α of the total bandwidth resources, with a resource budget B_m such that any combination of m and b_i^m that meets the constraint $B_m \geq \sum_{i=1}^{m} b_i^m$ can be applied. In our simulation, a candidate mix pool consists of 1000 benign nodes and m malicious nodes. The total bandwidth of candidate pool is $P_{bw} \approx 11400$ MBps including 2280 MBps malicious bandwidth ($\alpha = 20\%$) such that honest Mixnodes are the majority.

5.4 Adversary Resource Allocation

The adversary must determine how best to allocate his bandwidth to maximize the compromised fraction of paths. Since the same Mixnode cannot be chosen twice, he must run at least 3 Mixnodes for a 3-layer Mixnet. The crucial insight here is that the adversary has knowledge of what algorithm is being run to establish topologies and can distribute its total budget as a particular number of nodes and bandwidth that would maximize its chances. That is, having a similar presence in each layer would maximize deanonymization, and the adversary needs to determine its resource endowment to achieve it.

To answer this adversarial question and thoroughly model the adversary, we ran numerous experiments using the `mixnets topology generator` (Section 6.1.1) to empirically investigate how topological construction algorithms shape the network. We statistically derive, over 200 K runs with $h = 0.75$, the probability of Mixnodes' placement into different layers. Each Mixnode will either go to one layer of the Mixnet or remain in the candidate pool.

The results are displayed in Figure 4, from which we can infer appropriate allocation strategies for the adversary. We can see (Figure 4a) that BwRand's clear preference for bandwidth is in favour of big Mixnodes (especially with bandwidth no less than 70 MBps), which will be assigned into three layers evenly. Figure 4c show that there is a 25% chance that each Mixnode will be placed



(a) BwRand (b) RandRand (c) RandBP (d) Bow-Tie

Figure 4: Expected probability to fall in Layer i depending on the Mix capacity, with sampling threshold $h = 0.75$. Pool is the expected probability to stay within the candidate pool and not participate.

into one of four positions with RandRand or RandBP. Bow-Tie (Figure 4d) shows a different position distribution, where Mixnodes with bandwidth in between $20 - 30$ MBps has the same chance of being placed into any layers. Thus allocating malicious bandwidth resources evenly across a number of nodes makes sense for the adversary.

When generating Mixnet topologies according to different construction designs, we consider an adversary who has individual resource allocation strategy respectively. An adversary will generate the following malicious Mixnodes: nodes with 71.25 MBps against BwRand, nodes with 11.75 MBps against RandRand and RandBP, and nodes with 20.72 MBps against Bow-Tie. Note that we also test the compromised fraction of paths with varied capacity (ranging from 1 MBps to 150 MBps[4]) of equal-size malicious nodes and the results supports our choices of best allocation strategy (please refer to the Appendix D).

6 EMPIRICAL ANALYSIS OF BOW-TIE

We now evaluate the security of the Mixnets with respect to the metrics and adversaries in Section 5. To do so, we develop two tools[5]: `mixnets topology generator (MTG)` to produce the reference and Bow-Tie topologies and `routesim` to evaluate the topologies on their expected security metrics of typical dynamic email-like usage. We conclude by investigating the necessity of Bow-Tie's guard layer and client-side guard-logic.

6.1 Tools

6.1.1 MTG. We implemented a scalable Mixnet topology generator incorporating the four mixnet construction algorithms in Python. We use Gurobi optimizer [29] to solve the linear bin-packing optimization problem [57]. The bandwidths of Mixnodes are generated by fitting to the bandwidth distribution of Tor relays from its historical data [52]. We use an R package [16] to fit the bandwidth data captured from Tor consensus documents and server descriptors from January 2021 to March 2021. Among three common right-skewed distributions [12] we choose the *gamma distribution* as the best-fitted via maximum likelihood estimation (MLE) method.

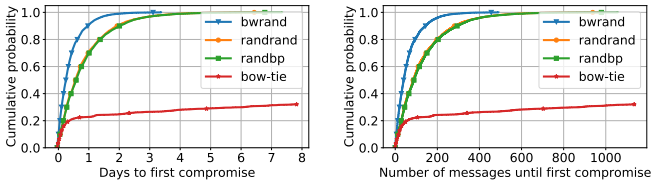
6.1.2 Routesim. To enable our evaluation of time to first compromise metric based on realistic Mixnets usage, we implement `routesim` to support the dynamic, multi-message user scenario, aiming at estimating user's resilience against client enumeration. `routesim` applies a Monte Carlo method to sample a user's usage distribution and simulate the user's expected anonymity impact. For each sample simulation, it takes the message timings and sizes following a communication pattern provided by the user, the Mixnet's topology generated by `MTG` for each epoch, and two families of mixing protocol interactions (recipient-anonymous and non recipient-anonymous, described in Section 7.1) as the input, and outputs the trace of all messages that are produced and transmitted through the network. `routesim` is written in *Rust*, scales with the number logical processors, has a low-memory footprint and is designed to be easily extensible for new client models and probabilistic events to capture. It can simulate statistically relevant durations (e.g. months) of a given client behavior in a few minutes on a regular laptop, i.e. it is usable on low budgets.

6.2 Analysis

6.2.1 Time to First Compromise. We assume a simple client that sends one message through the Mixnet every 5 to 15 minutes at random within this interval and we model 10,000 such clients. We use `routesim` to conduct simulations and obtain the distribution of time to first compromised message. The network churn rate between each epoch is 3% for each simulation. The epoch value is set to 1 hour; i.e., at each epoch, the network topology is refreshed according to the topology sampling and placement algorithms introduced in Section 4. The choice of one hour copies Tor's consensus document renewal.

Results. Figure 5 shows the CDF for the event that a user's message first traverses over a fully compromised path. For the reference designs, the client is expected to use a fully compromised route extremely fast since each message has the potential to go over *any* of the potential routes and the users will expose themselves to many Mixnodes, including adversarial ones. We can see (Figure 5a) that with all three reference designs there is more than 80% chance of deanonymization of at least one message within 2 days by an adversarial Mixnode and the median time to full compromise is less than 0.7 days. By looking at the distribution of messages sent (Figure 5b) for the reference designs, the median number of messages sent (for the "simple" client model) before compromise is 100. In

[4] Allocating too much bandwidth to one node is not realistic due to the CPU cost of the public-key encryption within each Mixnet packet, so we set the upper bound of malicious mix to 150 MBps.
[5] Both are open sourced at https://github.com/sus0pid/BowTie-Artifacts



(a) Time to first compromised message.

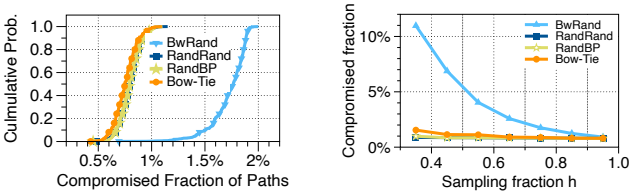
(b) Number of messages until first compromised message.

Figure 5: Empirical distribution of how much time/how many messages before a user's message traverses over a fully compromised path since first usage. We model a user sending one message every 5 to 15 minutes at random.

contrast, Bow-Tie enjoys a significantly longer time and higher number of messages sent until first compromise.

6.2.2 Compromised Fraction of Paths. We now evaluate how many network paths the adversary may control by considering the fraction of compromised paths metric. Recall that a path or route within Mixnets is compromised if the entire route is composed by malicious Mixnodes. Thus, we set the compromised fraction of paths F_b in a stratified l-layer Mixnet using bandwidth-weighted message forwarding as

$$F_b = \prod_{i=1}^{l} \frac{\text{Amount of Malicious Bandwidth in Layer } i}{\text{Amount of Bandwidth in Layer } i}. \quad (1)$$



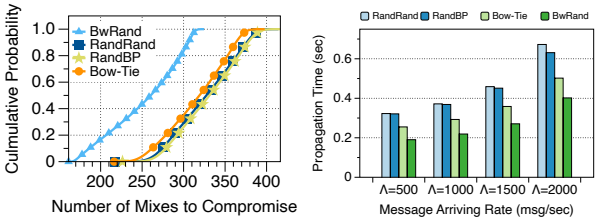
(a) Probability distribution on compromised fraction of paths with $h = 0.75$.

(b) Average compromised fraction of paths for h between 0.35 to 0.95.

Figure 6: Empirical distribution on compromised fraction of paths and empirical average compromised fraction.

We empirically evaluate this metric, statistically derived over 1000 runs, with simulations that construct Mixnets using Bow-Tie and reference algorithms. Adversarial bandwidth is set to 20% of the total network bandwidth.

Results. Figure 6a shows that, when $h = 0.75$, there is more than a 99% chance of compromising less than 1% paths using Bow-Tie, and the Rand- reference algorithms. In contrast, in BwRand the adversary can compromise upto 2% paths. This is because selecting all nodes by bandwidth in BwRand gives the adversary that intelligently allocates bandwidth an advantage. This is not so effective against Bow-tie since the non-guard layers use random placement.



(a) Guessing Entropy

(b) Expected queuing delay

Figure 7: (a) Guessing Entropy: expected number of Mixnodes the adversary had to compromise to trace a target message. (b) Expected queuing delay, with message arriving rate Λ for the Mixnet.

Figure 6b shows the worst-case expected compromise rates, where all malicious relays are selected for use under all the values of h considered. We see that Bow-Tie, RandRand, and RandBP have generally low compromise rates across all sampling fractions h, with Bow-Tie slightly higher (less than 0.05%) when $h < 0.6$. This is due to the fact that the guard layer is bandwidth weighted, however, the non-guard layers minimize an intelligent adversary's optimal allocation strategy. In contrast, as h decreases BwRand's compromise rates increase, with 10.9% of paths compromised when $h = 0.35$. The compromise rates are generally converging towards a lower value (around 0.08%) as h increases for all algorithms, which is expected since more honest nodes will enter the active pool and the fraction of adversarial relays will decrease.

This raises an interesting question about how to derive and adjust the parameter of sampling fraction h. The appropriate h should be able to handle all of the incoming traffic without overloading the majority of Mixnodes, and should limit the number of paths in the network to avoid very thin traffic from the perspective of entropy [28]. Thus, h should be set as a minimum value that satisfies the throughput requirement, based on historical data or reasonable predictions. We leave as future work the case when the volume of incoming traffic changes suddenly within an epoch.

6.2.3 Guessing Entropy. We model the deanonymization of a given message as a guess and let $\mathcal{G}$ represent the total number of guesses for success (i.e. deanonymizing the target message). $\mathbb{E}(\mathcal{G})$ is computed by selecting the nodes in descending order of the marginal probability p_i that the adversary can deanonymize the targeted message when cumulatively compromising the i_{th} node. Thus, the guessing entropy can be calculated by:

$$\mathbb{E}(\mathcal{G}) = \sum_{i \in |Pool_{active}|} i \cdot p_i, \quad (2)$$

where $|Pool_{active}|$ represents the number of Mixnodes in the *active pool*.

Results. Figure 7a shows the cumulative guessing entropy value obtained from 1000 trials of each topological construction algorithm, for a network containing $\approx$ 1000 nodes. We can see that the median number of Mixnodes required to compromise by an adversary for BwRand is around 250, while for other three algorithms, the median is increased to less than 320. While Bow-Tie is edged out by RandRand, and RandBP, it is significantly more secure in

the dynamic setting (above) and with better performance, as we shall see next.

6.2.4 Performance Evaluation. We measure the *expected queuing delay* (i.e., expected message queuing time) based on the topologies generated by the MTG with $h = 0.75, \alpha = 0.2$. The expected queuing delay is calculated by using a $M/D/1$ queue model [48]. The input messages of the whole mix network can be treated as a Poisson process with rate Λ. The message queuing time for each node is inversely proportional to its capacity, e.g., for a Mixnode with b_i bandwidth, the average processing time for it is $u_i = 1/b_i$. [6]

In bandwidth-weighted path selection, using U to represent the total bandwidth of the current layer, the expected queuing time of this layer is:

$$T_b = \sum_{i=1}^{k} \frac{2 - \frac{\Lambda}{U}}{2(U - \Lambda)} = k \frac{2 - \frac{\Lambda}{U}}{2(U - \Lambda)}. \quad (3)$$

Results. Figure 7b shows the expected delay due to queuing for a message going through the Mixnet. Indeed, algorithms that sample using bandwidth (i.e., BwRand and Bow-Tie) achieve relatively low processing delay and outperform random sampling schemes. Compared to BwRand, Bow-Tie sacrifices less than 0.05 seconds of queuing delay for a comparatively higher security level (see Figures 5 and 6a).

Note that Bow-Tie topologies are also fast to generate: a sub-second cost to both generate the Guard layer and to apply the bin packing optimization to the other layers.

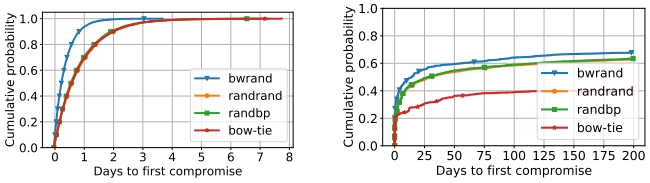
6.2.5 Recap. Our empirical results in this section confirm that the construction and routing of a Mixnet is characterised by a security and performance trade-off. Taken together, the results for these metrics show that Bow-Tie provides a high level of protection for users' anonymity in a dynamic and realistic setting with a relatively small sacrifice in performance.

6.3 Necessity of Both Client Guard Logic and Guard Layers

6.3.1 Turn off Client Guard-logic for Bow-Tie? A natural question is does the guard layer by itself (i.e. where the client does *not* maintain a guard list) provide a high level of protection. To answer this question we turn off the clients' guard list maintenance logic while they use the network, but keep Bow-Tie's other aspects the same (i.e. Bow-Tie still produces a guard layer).

As we observe in Figure 8a, a guard layer by itself has reduced security at a comparable level to those schemes without guard layers (i.e. RandRand, RandBP, and BwRand), although Bow-Tie is still slightly better. Nevertheless, the client is expected to use a fully compromised route extremely fast since each message is sent at random (bandwidth-weighted here), and the users will expose themselves to many Mixnodes. This implies that users should not explore all potential routes, which is the exact effect of the client guard-logic.

[6] We focus on bandwidth-weighted path selection since it performs an order of magnitude better than random path selection. The interested reader can refer to Appendix C for the random path selection performance results.



(a) Without client-side guard logic. **(b) Enabling client-side guard logic.**

Figure 8: Comparison between reference algorithms and Bow-Tie without/with client side guard logic.

6.3.2 Turn on Client Guard-logic for Reference Methods? We now turn on the client guard-list logic for *all* designs, Bow-Tie and the references, since the client component is fundamentally independent of the layer construction algorithm. For the reference designs the client will select initial and replacement relays from the middle layer using the client guard-list logic. This will allow us to gauge the effect of the client guard-list on designs without an engineered guard layer. Figure 8b shows the results of this comparison. Note that the results we provide here are independent of the Guard's position in users' routes. We see that all the reference designs improve with client guard-logic enabled. However, it is clear that Bow-Tie enjoys a significantly higher time to first compromise metric than the reference designs with client guard-list logic enabled. This means that the guard layer provides an added benefit that the client guard-list by itself does not provide, providing at least a 30% improvement over the most similar reference design RandBP.

This confirms the necessity of both Bow-Tie's guard layer and client guard-logic that combined reduce clients' *guard exposure* more effectively than they each could alone.

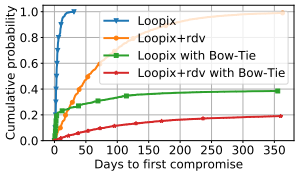
7 INFLUENCE OF PROTOCOLS AND USER BEHAVIOR

In general, user anonymity is significantly impacted by aspects that we organize into three broad and independent families: topological design choices, Mixnet protocol designs, and user behavior.

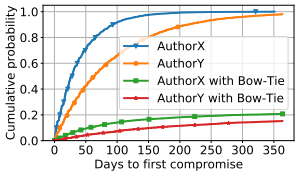
Our discussions and analysis so far concerned topological design choices, which refer to engineering aspects of the network itself (such as our guard design) to maximize users' expected anonymity. Other designs such as Atom [36] or XRD [38] add strong topological constrains making the anytrust assumption realistic and trustworthy[7], but at the price of severe performance impact limiting potential network use-cases and wide adoption. It is up to the user, and or application designer what trade-off is appropriate for their use-case.

So far we have not considered the impact of protocol integration or client usage, which, if done carelessly, may nullify the benefits of Bow-Tie's topological design choices. For example, BiTorrent [44] exchanges IP information with a tracker required by its application-level protocol. For this reason, tunneling BiTorrent inside an anonymous network does not provide anonymity protection, yet this user

[7] Due to their fundamentally different designs (i.e. round-based and dependance on heavy cryptographic primitives) it is not apt to compare their anonymity or performance to Bow-Tie.



(a) Benefit of rendezvous-based protocols over no-recipient anonymity.



(b) Comparing the email behavior of two authors with/without Bow-Tie designs.

Figure 9: Influences of protocol interaction and individual behaviours.

activity is observed in Tor [21]. In the same vein, for Mixnets we cannot tunnel many existing protocols as-is because it may similarly also nullify the Mixnet's protection. For example, in email, SMTP and IMAP servers contain many pieces of meta-information that can link users to their activity, and even the plaintext if the user does not manually set-up end-to-end encryption (which requires advanced understanding of threats, email, and technology). It is also the case for secure messaging applications, such as Signal, which leverage a central server to enable confidential communications (while exposing the users' social graph to the central server). To mitigate these threats, the Mixnet protocol suite has to offer the means [69] to perform asynchronous messaging, which applications could then use to build secure and private protocols.

Similarly, user behavior also has a significant impact. For users sending a single message in the network, we can evaluate the user's anonymity via entropic considerations. Different entropy measures may capture different criteria [48, 63] and lead to different interpretation of the user's anonymity.

We now bring these aspects into our investigation to round out our evaluation of Bow-Tie.

7.1 Influence of Protocol Designs

We now consider the impact of a recipient anonymity property over the Mixnet protocol design. Recipient anonymity may be needed to improve users anonymity in some context, and unnecessary in others. For example, uploading a file to a public-facing server would not be recipient-anonymous. Exchanging messages asynchronously with a peer at a private address would be recipient-anonymous. To obtain recipient-anonymity, we assume the existence of a private and secure Naming scheme and rendezvous protocol (also called "dialing") defined by the Mix network [39, 64, 68, 70].

There is a rich history of anonymous networks that claim strong anonymity [4, 6, 8, 11, 14, 18, 27, 36, 37, 39, 40, 51, 61, 68, 70, 71]. Thoroughly studying and comparing the influence of those various designs is out of scope of this paper. However, we can explore how simple design choices can lead to significant recipient anonymity improvements for continuous-time Mix networks.

Loopix [51] and the Nym Network [18] based on the Loopix design do not offer recipient anonymity for asynchronous messages between clients. Other designs such as Tor [64], Vuvuzela [70], Stadium [68] and Karaoke [39] do offer it thorough a rendezvous protocol (also called "dialing") to asynchronously connect peers both seeking to communicate together anonymously. `routesim` can model the two approaches and evaluate the benefit of rendezvous-based protocols, with respect to users' activity and the path length. The user model is based on a real-world Email sending patterns built from a dataset of University staff members. The dataset was built from meta-info contained within the `sendmail` logs from the university SMTP server over a period of two months with the sending habits of *hundreds* IT staff members from the authors' faculty. [8] The network churn rate between each epoch is 3% and the number of hops to destination or rendezvous is 3. Figure 9a shows an evaluation of a typical Email sending pattern derived from the dataset. We see that designs with rendez-vous protocol have better security and in combination with Bow-Tie are significantly more secure. Given those results, existing deployments, such as Nym, may find valuable to incorporate recipient anonymity. However, there is a cost to obtaining recipient anonymity this way; it doubles the bandwidth consumption for asynchronous messaging, and requires the establishment of an out-of-bound solution to propagate addressing information. Many different approaches have been detailed in the literature [40, 66, 70]. Privately accessing and retrieving the mailbox contents [25] is also a potential approach to gain recipient anonymity, yet would limit the size of the anonymity set to the number of mailboxes stored on a given mixnode and be significantly more CPU costly.

7.2 Evaluating Individual Risks

Our earlier analysis considered a simple client. We now consider and evaluate complex personal usage behaviors. We built several datasets containing typical weekly behavior from years of our own email communication patterns and fed them into `routesim`. Knowing how we behave in a typical period of one week, `routesim` plays a sequence of events (i.e. sending emails)—that statistically matches our recorded behavior—indefinitely through time. Note that `routesim` could also simulate other usage patterns, provided a dataset is available.

In `routesim`, many configuration options are possible. For this experiment, we assume each user has a set of ten contacts, use the Bow-Tie topology with a 3% Mixnode churn rate and an epoch of 1 day, set the route length of the Mix network is three, and assume that the Mix network exposes a protocol suite for asynchronous messaging offering anonymity for both communicants (i.e. a naming scheme and a rendezvous protocol). The Mix network carries the same quantity of data as was typically contained in the authors' sending email patterns, rounded up to a product of the Mixnet message payload length (2048 bytes). Essentially, a sender sends the (end-to-end encrypted) message to the recipient's Mailbox located within one of the Mixnodes. The recipient anonymously retrieves the Mailbox contents on demand. The protocol to check and retrieve the encrypted content is assumed to be derived from a PIR protocol [9] to avoid leaking which Mailbox is queried to the Mixnode. In `routesim`, we assume (it is configurable) that a user's Mailbox changes its location at each epoch (i.e., handled by a different Mixnode selected at random in the first $N-1$ layers). We advise Mixnet developers to never store any encrypted content

[8] See Appendix B for ethics details.

on a Mixnode that can exit to the clearnet, hence to never store Mailboxes on a Mixnode that can be placed in the Nth layer.

Figure 9b shows the time to compromise the first pair[9] of communicants with email-like communication patterns. The simulated users `AuthorX` and `AuthorY` have a different email-like communication pattern in terms of frequency leading to a significant difference in the time to first compromise. In this simulation, users change their mailbox location every day, meaning that all emails sent the same day to the same contact are all guaranteed to be exposed to the same Mailbox. Therefore, with this design choice, the more the user's emails are sparsely sent in time, the more likely they are exposed to different contact's Mailboxes. Different design choices would lead to different results. For example, users could decide to change their Mailbox location not on a day-by-day basis, but rather dependent on the number of email messages which they fetched. Eventually such design choice needs to be enforced by the Mixnet developers, and can be evaluated with `routesim`.

In the same vein, given an established design such as Bow-Tie, end users such as journalists or whistleblowers can use `routesim` to evaluate their chance of being deanonymized assuming a realistic mix adversary. Results obtained may help them evaluating whether the risks are worth their information.

7.3 DoS Attack Discussion

We discuss the potential effects of targeted DoS attacks on our Bow-Tie design. Broadly, it is reasonable to assume the mixnode/CS hosting provider will try to mitigate (sustained) DoS attacks against its customers, e.g., Cloudflare [10] provides consumer-grade DoS attacks resistance. Furthermore, mixes are assumed to be operated by non-colluding and geographically diverse parties, which raises the bar on the attack. That said, we acknowledge that a user can be forced to pick a new (potentially malicious) guard if their current honest guard is forced offline, but will revert back when the honest guard comes back online.

The best theoretical adversarial strategy against Bow-Tie to optimize the likelihood to deanonymize users depends mostly on the target behaviour. If the target is ephemeral (e.g., would only send a few messages), then the adversary requires to DoS evenly on all layers to optimize the fraction of compromised paths. If the target is not ephemeral, then controlling a large fraction of the guards, applying DoS attacks on honest guards and keeping them offline would force new potentially adversarial guards on clients using them. This strategy would increase usage of the adversarial guards, and usage of adversarial resources in the guard set. This strategy is not optimal in a static setting (i.e., at a given time), but should increase the likelyhood to observe all non-ephemeral users quickly through time. A subtlety of Bow-Tie is that the adversary has to maintain the DoS on all previous forcefully-evicted guards even though new adversarial nodes get inserted in the guard set, or users will revert back to their older honest guard.

The key takeaway is that in continuous-time mixnets (independent of Bow-Tie) the adversary's strength grows with the size of the DoS attack and under large attacks there cannot be an expectation for high anonymity. As a general mitigation network maintainers need to be vigilant in detecting DoS attacks. Bow-Tie 's client guard logic provides a measure of protection by making these attacks costly.

[9]The authors are always the message senders in these pairs.

8 RELATED WORK

The literature is rich of Mixnets proposals [3, 4, 6, 8, 11, 14, 18, 25, 27, 36–38, 40, 49, 51, 61, 68, 70, 71]. Some works investigate the detection and mitigation of active malicious mixes and combine it with the Mixnet construction design. Dingledine and Syverson [23] discuss how to build a mix cascade network through a reputation system that decrements the reputation score of all nodes in a failed cascade, and increments the reputation of nodes in a successful cascade. It improves the reliability of mixnet and reduce the chance that an adversary controls an entire cascade. However, some pitfalls are introduced by the reputation system and the actual deployment is still a complex problem. Leibowitz et al. propose Miranda [41], another reputation-based design that detects and isolates active malicious mixes. They also discuss how to construct the cascade mixnet based on their faulty mixes detection scheme and a set of cascades are selected randomly for the upcoming epoch. This design relies on a fixed set of mixes and it is still challenging to deploy in the real world.

Nym [17] stratified network is periodically constructed from a large number of available mixes run by profit-motivated mix operators, who are compensated for their investment with payment in Nym's cryptocurrency tokens. Nym's design, is sketched out in a whitepaper [18], presenting their solution to construct a Mixnet by randomly selecting mixes weighted by mixes' stake and randomly placing them into layers. Nym uses a verifiable random function (VRF) [46] to facilitate the features of decentralization in their blockchain-based ecosystem.

Guirat and Diaz [28] investigate how to optimize the Mixnet parameters for a continuous-time mix network, and focus on the number of layers and the width of the network (i.e., the number of nodes in each layer). They theoretically analyze the fully compromised rate for a continuous-time mix network in a designated shape and they mainly concentrate on optimizing the Mixnet parameters using the Shannon entropy [20, 58, 60] as the guiding metric.

9 CONCLUSION

In this paper, we address the question of "how to shape the Mixnet to strengthen the anytrust assumption?" and study the design of Mixnet configuration and routing that limits the adversary's power to deanonymize traffic. We proposed Bow-Tie, an efficient novel design for mix network engineering; we present the first thorough security analysis of stratified Mixnet against reasonably realistic adversaries; we develop the *routesim* simulator that can easily calculate users' expected deanonymized probability. In the future, we will further explore the case with untrusted configuration server.

ACKNOWLEDGMENTS

We thank our shepherd, reviewers, and the artifact evaluation committee for their helpful comments in improving this paper and the accompanying artifacts. Florentin Rochet and Tariq Elahi were supported by REPHRAIN: The National Research Centre on Privacy, Harm Reduction and Adversarial Influence Online, under UKRI grant: EP/V011189/1.

REFERENCES

[1] Dakshi Agrawal and Dogan Kesdogan. 2003. Measuring anonymity: The disclosure attack. *IEEE Security & privacy* 1, 6 (2003), 27–34.

[2] Greubel Andre, Dmitrienko Alexandra, and Kounev Samuel. 2018. Smartor: Smarter tor with smart contracts: Improving resilience of topology distribution in the tor network. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 677–691.

[3] Sebastian Angel and Srinath Setty. 2016. Unobservable Communication over Fully Untrusted Infrastructure. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, Savannah, GA, 551–569. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/angel

[4] Yawning Angel, George Danezis, Claudia Diaz, Ania Piotrowska, and David Stainton. 2017. Katzenpost Mix Network Specification. https://github.com/Katzenpost/docs/blob/master/specs/mixnet.rst.

[5] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. 2007. Denial of service or denial of security?. In *Proceedings of the 14th ACM conference on Computer and communications security*. 92–102.

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

[7] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri De Ruiter, and Alan T Sherman. 2017. cMix: Mixing with minimal real-time asymmetric cryptographic operations. In *International conference on applied cryptography and network security*. Springer, 557–578.

[8] David L Chaum. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (1981), 84–90.

[9] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. 1995. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 41–50.

[10] Cloudflare. 2022. Cloudflare Comprehensive DDoS Protection. https://www.cloudflare.com/en-gb/ddos/. Accessed: September 2022.

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

[12] Alison C Cullen, H Christopher Frey, and Christopher H Frey. 1999. *Probabilistic techniques in exposure assessment: a handbook for dealing with variability and uncertainty in models and inputs*. Springer Science & Business Media.

[13] George Danezis. 2003. Mix-networks with restricted routes. In *International Workshop on Privacy Enhancing Technologies*. Springer, 1–17.

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

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

[16] Marie Laure Delignette-Muller, Christophe Dutang, et al. 2015. fitdistrplus: An R package for fitting distributions. *Journal of statistical software* 64, 4 (2015), 1–34.

[17] Claudia Diaz, Harry Halpin, and Aggelos Kiayas. 2021. *The Nym Network-The Next Generation of Privacy Infrastructure*. Technical Report. Nym Technologies SA.

[18] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. 2021. The Nym Network. https://nymtech.net. Whitepaper. Accessed March 2022.

[19] Claudia Diaz, Steven J Murdoch, and Carmela Troncoso. 2010. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 184–201.

[20] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. 2002. Towards measuring anonymity. In *International Workshop on Privacy Enhancing Technologies*. Springer, 54–68.

[21] Roger Dingledine. 2010. Bittorrent over Tor isn't a good idea. https://blog.torproject.org/bittorrent-over-tor-isnt-good-idea/. Accessed: May 2022.

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

[23] Roger Dingledine and Paul Syverson. 2002. Reliable MIX cascade networks through reputation. In *International Conference on Financial Cryptography*. Springer, 253–268.

[24] Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. 2012. Changing of the guards: A framework for understanding and improving entry guard selection in Tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*. 43–54.

[25] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. 2021. Express: Lowering the Cost of Metadata-hiding Communication with Cryptographic Privacy. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 1775–1792. https://www.usenix.org/conference/usenixsecurity21/presentation/eskandarian

[26] S. Farrel and H. Tschofenig. 2014. *Pervasive Monitoring Is an Attack*. RFC 7258. RFC Editor. 1–5 pages. https://www.rfc-editor.org/rfc/rfc7258.txt

[27] Michael J Freedman and Robert Morris. 2002. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*. 193–206.

[28] Iness Ben Guirat and Claudia Diaz. 2022. Mixnet optimization methods. *Proceedings on Privacy Enhancing Technologies* 1 (2022), 22.

[29] Incorporate Gurobi Optimization. 2018. Gurobi optimizer reference manual. https://www.gurobi.com/documentation/9.1/refman/index.html.

[30] Aaron D Jaggard, Aaron Johnson, Sarah Cortes, Paul Syverson, and Joan Feigenbaum. 2015. *20,000 in league under the sea: Anonymous communication, trust, MLATs, and undersea cables*. Technical Report. NAVAL RESEARCH LAB WASHINGTON DC.

[31] Aaron D Jaggard, Aaron Johnson, Paul Syverson, and Joan Feigenbaum. 2014. Representing network trust and using it to improve anonymous communication. *arXiv preprint arXiv:1406.3583* (2014).

[32] Rob Jansen and Aaron Johnson. 2021. On the accuracy of Tor bandwidth estimation. In *International Conference on Passive and Active Network Measurement*. Springer, 481–498.

[33] Aaron Johnson, Rob Jansen, Nicholas Hopper, Aaron Segal, and Paul Syverson. 2017. PeerFlow: Secure Load Balancing in Tor. *Proc. Priv. Enhancing Technol.* 2017, 2 (2017), 74–94.

[34] Aaron Johnson, Rob Jansen, Aaron D Jaggard, Joan Feigenbaum, and Paul Syverson. 2015. Avoiding the man on the wire: Improving tor's security with trust-aware path selection. *arXiv preprint arXiv:1511.05453* (2015).

[35] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. 2013. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 337–348.

[36] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. 2017. Atom: Horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 406–422.

[37] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. 2016. Riffle. *Proceedings on Privacy Enhancing Technologies* 2016, 2, 115–134.

[38] Albert Kwon, David Lu, and Srinivas Devadas. 2020. XRD: Scalable Messaging System with Cryptographic Privacy. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 759–776. https://www.usenix.org/conference/nsdi20/presentation/kwon

[39] David Lazar, Yossi Gilad, and Nickolai Zeldovich. 2018. Karaoke: Distributed Private Messaging Immune to Passive Traffic Analysis. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 711–725. https://www.usenix.org/conference/osdi18/presentation/lazar

[40] David Lazar and Nickolai Zeldovich. 2016. Alpenhorn: Bootstrapping secure communication without leaking metadata. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 571–586.

[41] Hemi Leibowitz, Ania M Piotrowska, George Danezis, and Amir Herzberg. 2019. No right to remain silent: isolating malicious mixes. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1841–1858.

[42] Isis Lovercruft, George Kadianakis, Ola Bini, and Nick Mathewson. 2016. *Another algorithm for guard selection*. Technical Report 271. https://gitweb.torproject.org/torspec.git/tree/proposals/271-another-guard-selection.txt

[43] Akshaya Mani, T. Wilson-Brown, Rob Jansen, Aaron Johnson, and Micah Sherr. 2018. Understanding Tor Usage with Privacy-Preserving Measurement. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18https://gitweb.torproject.org/torspec.git/tree/proposals/310-bandaid-on-guard-selection.txt)*. Association for Computing Machinery, New York, NY, USA, 175–187. https://doi.org/10.1145/3278532.3278549

[44] Pere Manils, Chaabane Abdelberi, Stevens Le Blond, Mohamed Ali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous. 2010. *Compromising Tor Anonymity Exploiting P2P Information Leakage*. Research Report. https://hal.inria.fr/inria-00471556

[45] J.L. Massey. 1994. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*. 204–. https://doi.org/10.1109/ISIT.1994.394764

[46] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. 1999. Verifiable random functions. *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)* (1999), 120–130.

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

[48] Steven J Murdoch and Robert NM Watson. 2008. Metrics for security and performance in low-latency anonymity systems. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 115–132.

[49] Zachary Newman, Sacha Servan-Schreiber, and Srinivas Devadas. 2022. Spectrum: High-bandwidth Anonymous Broadcast. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 229–248. https://www.usenix.org/conference/nsdi22/presentation/newman

[50] Nym. 2019. The Nym Project's sphinx implementation. https://github.com/nymtech/sphinx. Accessed: April 2022.

[51] Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. 2017. The loopix anonymity system. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 1199–1216.
[52] The Tor Project. 2022. CollecTor - Tor Project. https://metrics.torproject.org/collector.html. Accessed: April 2021.
[53] Florentin Rochet and Aaron Johnson. 2019. *Towards load-balancing in Prop 271.* Technical Report 310. https://gitweb.torproject.org/torspec.git/tree/proposals/310-bandaid-on-guard-selection.txt
[54] Florentin Rochet and Olivier Pereira. 2017. Waterfilling: Balancing the Tor network with maximum diversity. *Proceedings on Privacy Enhancing Technologies* 2017, 2 (2017), 4–22.
[55] Florentin Rochet and Olivier Pereira. 2018. Dropping on the Edge: Flexibility and Traffic Confirmation in Onion Routing Protocols. *Proceedings on Privacy Enhancing Technologies* 2018, 2 (2018), 27–46.
[56] Florentin Rochet, Ryan Wails, Aaron Johnson, Prateek Mittal, and Olivier Pereira. 2020. *CLAPS: Client-Location-Aware Path Selection in Tor.* Association for Computing Machinery, New York, NY, USA, 17–34. https://doi.org/10.1145/3372297.3417279
[57] Guntram Scheithauer. 2017. *Introduction to cutting and packing optimization: Problems, modeling approaches, solution methods*. Vol. 263. Springer.
[58] Andrei Serjantov and George Danezis. 2002. Towards an information theoretic metric for anonymity. In *International Workshop on Privacy Enhancing Technologies*. Springer, 41–53.
[59] Andrei Serjantov, Roger Dingledine, and Paul Syverson. 2002. From a trickle to a flood: Active attacks on several mix types. In *International Workshop on Information Hiding*. Springer, 36–52.
[60] Andrei Serjantov and Richard E Newman. 2003. On the anonymity of timed pool mixes. In *IFIP International Information Security Conference*. Springer, 427–434.
[61] Emin Gün Sirer, Sharad Goel, Mark Robson, and Doğan Engin. 2004. Eluding carnivores: File sharing with strong anonymity. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. 19–es.
[62] Robin Snader and Nikita Borisov. 2009. EigenSpeed: secure peer-to-peer bandwidth evaluation.. In *IPTPS*. 9.
[63] Paul Syverson. 2009. Why I'm not an entropist. In *International Workshop on Security Protocols*. Springer, 213–230.
[64] Paul Syverson, Roger Dingledine, and Nick Mathewson. 2004. Tor: The second-generation onion router. In *Usenix Security*. 303–320.
[65] The Tor Project. 2022. Tor directory protocol, version 3. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt. Accessed: May 2022.
[66] The Tor Project. 2022. Tor Protocol Specification. https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt. Accessed: May 2022.
[67] Matthew Traudt, Rob Jansen, and Aaron Johnson. 2021. Flashflow: A secure speed test for tor. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 381–391.
[68] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. 2017. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 423–440.
[69] Nik Unger and Ian Goldberg. 2015. Deniable Key Exchanges for Secure Messaging. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 1211–1223. https://doi.org/10.1145/2810103.2813616
[70] Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. 2015. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*. 137–152.
[71] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. 2012. Dissent in numbers: Making strong anonymity scale. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*. 179–182.
[72] Matthew K Wright, Micah Adler, Brian Neil Levine, and Clay Shields. 2004. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Transactions on Information and System Security (TISSEC)* 7, 4 (2004), 489–522.

A GUARD'S POSITION CONSIDERATIONS

In Section 4.2, we present the Guard idea for Continuous-time mixnets which aims at reducing users' exposure to malicious mixnodes. Choosing the position of the Guard in users' path of length L is an interesting question leading the following analysis:

- Choosing the last layer could allow a malicious guard to perform re-identification attacks based on prior knowledge. For example, if the network is used to connect to user-dependent destinations (Services, set of contacts), then the *a priori* knowledge of this relation would reveal the identity of the mixnetwork user.
- Choosing a layer in $[2..L-1]$ has the advantage, compared to Tor, to not directly bind the long-lived guard to the user. That is, discovering the identity of a user's guard does not lead directly to the user, i.e. the Guard's ISP can not be compelled to reveal the client IP addresses connecting to the guard relay. Low-latency anonymity networks such as Tor cannot move the guard's position into some layer $[2..L-1]$ as their threat's model expect end-to-end traffic confirmation to succeed in deanomymizing a user-destination relation. Therefore, for a low-latency design, moving the guard to a layer $[2..L-1]$ would achieve nothing. We do not have this issue with Continuous-time mix-networks.
- Choosing the first layer has a massive performance advantage in continuous-time mix networks using Sphinx packets [15], the state of the art packet format specification for Continuous-time mix networks. Indeed, currently, a full cryptographic handshake is performed for each Sphinx packet, which is needlessly costly when all packets are sent to the same first node (the guard), and only one cryptographic handshake for a determined session period would lead to much lower performance impact.
  One possible method is for clients to perform L-1 Sphinx processing (for hops $2..L$) and 1 TLS processing for the first hop. We do a small scale experiment for preliminary indicative results. We compare the throughput of a Rust sphinx implementation [50] with a `AES-128-GCM` openssl benchmark, the most used cipher in TLS1.3, over 1024 bytes blocks. The choice of 1024 bytes comes from the default Sphinx packet size choice. Over a AMD Ryzen 7 3700X, we were able to perform 8261 Sphinx unwrap/s for a payload of 1024 bytes. With `AES-128-GCM`, we processed $\approx 500\times$ more packets per seconds. Moreover, TLS has a maximum payload size of 16KiB, which means that multiple sphinx packets can be encrypted within the same record, leading to a performance improvement of $\approx 600\times$, on average from our benchmarks.

Therefore, choosing the guard layer position is a trade-off between user anonymity and performance. Users' anonymity benefits from guards in layer $[2..L-1]$, while network performance benefits from guard position in the first layer, and making the upper layer of the onion encryption scheme being a TLS session instead of a Sphinx encryption.

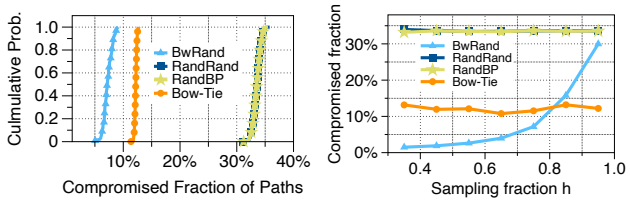
Our experimental analysis and results in this paper are independent of the Guard's position within user's path. We leave choice of trade-off to the implementer.

B EMAIL DATASET ETHICS

The University email dataset collection ethics application was filed with the faculty's ethics process (application #41564). This was approved prior to the IT department initating the collection of the data. Only select meta-data (from email headers) was collected relevant to email sending patterns. All personal information in the headers was pseudonymized before we were given access.

C EMPIRICAL RESULTS OF RANDOM PATH SELECTION

We evaluate the security and performance of Bow-Tie and reference methods with random path selection using the metric of *compromised fraction of paths* and *expected queuing delay*. Note that the adversary's best resource allocation policy under random path selection is to inject as many malicious Mixnodes as possible, since the quantity matters more than bandwidth. In our simulation, we instantiate this best strategy as generating thousands of Mixnodes with a minimum of 1 MBps, since there could be an infinite number of malicious Mixnodes if we do not set a lower bound.



(a) Probability distribution on compromised fraction of paths with $h = 0.75$. **(b) Average compromised fraction of paths for h between 0.35 to 0.95.**

Figure 10: Compromised fraction of paths using random path selection, $\alpha = 0.2$.

(1) *Security evaluation.* We set the compromised fraction of paths F_r in a stratified l-layer Mixnet using random message forwarding as

$$F_r = \prod_{i=1}^{l} \frac{\text{Number of Malicious Mixnodes in Layer } i}{\text{Number of Mixes in Layer } i}. \quad (4)$$

Figure 10a shows that, when $h = 0.75$, BwRand limits the compromise rate between 5% and 9% with relatively higher security guarantee in comparison to other methods. By looking at Figure 10b, we also see that BwRand mitigates the adversary's compromising power in a wide range of h and provides the best protect in this case. Therefore, BwRand coupled with a uniform path selection may appear to be an interesting candidate. However, as shown in Figure 10b, the best compromise rate that we can get from BwRand&Random path selection (RPS) is around 1.89% with $h = 0.35$, which is comparable to the worst compromise rate that we obtain from Bow-Tie&Bandwidth-weighted path selection (BPS) is around 1.92% with $h = 0.35$ (Figure 6b). Besides, BwRand&RPS shows a dramatic increase as h increase while Bow-Tie&BPS enjoys a stable security level.

(2) *Performance evaluation.* Suppose there are n nodes in one layer, then the expected queuing time in random path selection setting for this layer is:

$$T_r = \sum_{i=1}^{n} \frac{n^{-1}u_i(2 - n^{-1}u_i\Lambda)}{2(1 - n^{-1}u_i\Lambda)}. \quad (5)$$

Figure 11 shows the expected delay due to queuing for a message going through the Mixnet with random path seleciton. Still, algorithms that sample using bandwidth (i.e., BwRand and Bow-Tie)

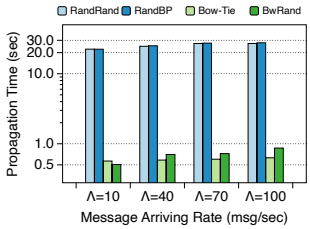


Figure 11: Expected queuing delay, with message arriving rate Λ for the Mixnet based on random path selection.

achieve relatively low processing delay and outperform random sampling schemes. However, the Mixnet takes more time to handle handles one order of magnitude low message arrival rates than in Bandwidth-weighted path selection.

D ADVERSARY RESOURCE ALLOCATION

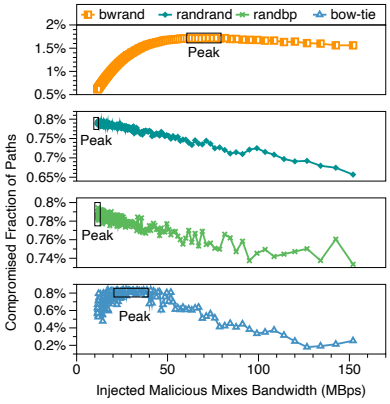


Figure 12: Fully compromised fraction versus bandwidth per injected malicious node. The adversary controls 2280 MBps bandwidth which is allocated to a number of equal-size Mixnodes.

The results, displayed in Figure 12, show that the compromised fraction of the adversary (Section 5.1) for different algorithms. The optimal capacity sizes of Mixnodes that maximizes the compromising rate are aligned with the information shown in Figure 4 and also confirms our choices of best resource allocation strategy (in Section 5.3).

E ALGORITHMS

Algorithm 1: Configuring Non-guard Layers

Input: candidate mix pool excludes guard nodes $P' = P - G$; sampling fraction h.
Output: configured two layers L_l, L_r for upcoming epoch i.

$L_l, L_r \longleftarrow \emptyset$
$P_{Active} \longleftarrow$ sample $\frac{2}{3}h * P_{bw}$ Mixnodes uniformly from P'
$n \longleftarrow |P_{Active}|$ `// Binpacking placement starts`
$W \longleftarrow \emptyset$
for $j \longleftarrow 0$ **to** n **do** `// Prepare weights for ILP`
  $W \longleftarrow W \cup b_j$
$c \longleftarrow (\frac{h}{3} + \epsilon) * P_{bw}$ `// Expected capacity for each layer`
$l \longleftarrow 2$ `// Number of projected layers`
$L_l, L_r \longleftarrow ILP(W, l, c)$
return L_l, L_r

Algorithm 2: Configuring Guard Layer

Input: candidate mix pool P with P_{bw} bandwidth; sampling fraction h; tolerance fraction τ.
Output: configured guard layer L_g for upcoming epoch i; updated guard set G.

if *first call* **then** `// Initialize the guard layer`
  Sample $\frac{h}{3} * P_{bw}$ nodes from P, weighted by bandwidth, as a set AG
  Sample $\tau * \frac{h}{3} * P_{bw}$ nodes from $P - AG$, weighted by bandwidth, as a set BG
  $G \longleftarrow AG \cup BG$ `// Give nodes in` AG `and` BG `a common label` G
  Place all AG nodes into guard layer L_g
  foreach *ag in* AG **do**
    $t_{AG} \longleftarrow 1$ `// Track working time as a guard`
  return L_g, G
else `// Maintain the guard layer`
  Update Mixnodes on/off status
  Update Mixnodes stability metric
  G, $DG \longleftarrow$ `MaintainGuardSet(`G`)`
  foreach *g in* $G - DG$ **do**
    if $t_{AG} > 0$ **then** `// Inherit old online ag`
      Move node g to AG
  $BG \longleftarrow G - DG - AG$
  $\delta \longleftarrow$ `TotalBw` $(AG) - T_{low}$
  if $\delta < 0$ **then** `// Insufficient ag`
    $AG +=$ `BSSample(`$BG, |\delta|$`)` `// Add` $|\delta|$ `nodes from` BG
  Place all AG nodes into guard layer L_g
  foreach *ag in* AG **do**
    update t_{AG} `// Track working time as a guard`
  return L_g, G
Function `MaintainGuardSet(`G`)`:
  Gather offline nodes in G to a subset DG
  $\delta_l \longleftarrow$ `TotalBw(`$G - DG$`)` $- T_{low}$
  $\delta_h \longleftarrow$ `TotalBw(`$G - DG$`)` $- T_{high}$
  if $\delta_l < 0$ **then** `// Too few online guards.`
    $G +=$ `BSSample(`$P - G$, min $\{|\delta_l|$, `TotalBw(`$P - G$`)}`
  else if $\delta_h > 0$ **then**`// Too many online guards`
    $S \longleftarrow \{g$ with $t_{AG} = 0\}$
    $G -=$ `IBSSample(`S, min $\{|\delta_h|$, `TotalBw(`S`)}`
  if *Every p epoch* **then** `// Periodically guard elimination.`
    $DG -= \{dg$ with *stability* $<$ *lowerbound*$\}$
  return G, DG
Function `BSSample(`T, k`)`:
  Normalize *WMTBF* to $0 - 1$ scale as *stability* for nodes in T
  Sort all nodes by *bw*×*stability* in descending order
  $S \longleftarrow$ Mixnodes that add up to min $\{k$, `TotalBw(`T`)}` bandwidth in order
  $T -= S$
  return S
Note: function `IBSSample()` is the same as `BSSample()` except sorting all nodes in an inverse order.