LARMix++: Latency-Aware Routing in Mix Networks with Free Routes Topology

Mahdi Rahimi[1] [0009−0003−0223−9082]

COSIC (KU Leuven), Leuven, Belgium
mahdi.rahimi@esat.kuleuven.be

Abstract. Mix networks (mixnets) enhance anonymity by routing client messages through multiple hops, intentionally delaying or reordering these messages to ensure unlinkability. However, this process increases end-to-end latency, potentially degrading the client experience. To address this issue, LARMix (NDSS, 2024) proposed a low-latency routing methodology specifically designed for stratified mixnet architectures. Our paper extends this concept to Free Routes mixnet designs, where, unlike stratified topologies, there are no restrictions on node connections. We adapt several state-of-the-art low-latency routing strategies from both mix and Tor networks to optimize the Free Routes topology. Despite the benefits, low-latency routing can cause certain mixnodes to receive disproportionate amounts of traffic. To overcome this challenge, we introduce a novel load-balancing algorithm that evenly distributes traffic among nodes without significantly compromising low-latency characteristics. Our analytical and simulation experiments demonstrate a considerable reduction in latency compared to uniform routing methods, with negligible loss in message anonymity, defined as the confusion an adversary experiences when correlating messages exiting the mixnet to an initially targeted input message. Additionally, we provide an analysis of adversarial strategies, revealing a balanced trade-off between low latency and adversary advantages.

Keywords: Mix network · Anonymity · Latency.

1 Introduction

Anonymous communication systems, designed to conceal the identities of communicators within network infrastructures, can be implemented using various methodologies [24]. Among the most prominent is the Tor overlay network [11], which supports over two million daily users. In Tor, a client selects three intermediary nodes—termed Guard, Middle, and Exit relays—and sequentially encrypts the message with the public keys of these relays. Each relay, upon receiving the message, decrypts it with its private key and forwards it to the next relay, with knowledge limited only to the identities of the adjacent relays in the message path, thereby preserving anonymity. However, Tor's architecture has vulnerabilities to adversaries who observe both ends of the communication

channel—specifically, the interactions between the client and the Guard relay, and between the Exit relay and the destination [27]. These vulnerabilities can be exploited through compromised Guard and Exit relays or by controlling Autonomous Systems (ASes) located between these communication points. Such exposures allow adversaries to conduct effective correlation attacks that can potentially deanonymize the communication endpoints [2].

The vulnerabilities of Tor to correlation attacks render it less secure and suboptimal for scenarios requiring high anonymity guarantees. To address these vulnerabilities and propose an anonymous system that not only mitigates Tor's weaknesses but is also immune to a global passive adversary (GPA) observing all network communications, the concept of a mix network (mixnet) has been followed in several studies [8,16,19,28]. Mixnets enhance the anonymity provided by Tor through implementing traffic mixing at each intermediary node, known as a mixnode, a concept pioneered by Chaum [5]. This mixing process ensures that the output from each mixnode is not directly linkable to its input, thus preserving anonymity. Consequently, as long as one mixnode correctly performs this mixing along the message route, it can effectively thwart a GPA's attempts to correlate traffic [7].

This obfuscation (traffic mixing) provided by mixnodes can be achieved through various methods. The first method involves threshold mixes [5], which accumulate messages until a preset count is reached before forwarding them. The second method employs pool mixnodes [10], requiring both a threshold number of messages and a time limit before messages are forwarded. The third method, stop-and-go mixes [14], introduces a random, exponentially distributed delay to each message before release. Among these methods, stop-and-go mixing offers a high degree of anonymity due to the memoryless property of the exponential distribution and has been recently put into practice due to its manageable average latency [8].

Like mixnode types, mixnet topologies can also be implemented in several distinct ways to enhance security and anonymity [9]. For a mixnet consisting of L intermediary hops, a common configuration is the cascade topology, where mixnodes are organized into cascades, each containing L mixnodes. Clients select one of these cascades for message routing [4, 16, 28]. An alternative configuration, known as Free Routes, involves clients randomly selecting L distinct nodes from all available mixnodes to establish a message path [17].[1] Another approach is the stratified topology, which organizes mixnodes into L different layers. A message path is then formed by selecting one mixnode from each layer, ensuring diversified routing and enhanced anonymity due to the many possible routing paths compared to the cascade topology [8, 19].[2]

These topological choices significantly influence the network's anonymity and its handling of cover traffic. Cover traffic consists of dummy messages introduced by clients or mixnodes that do not have a designated destination and are eventu-

[1] The Tor network also employs a Free Routes topology for message routing.

[2] However, the stratified topology provides less anonymity compared to the Free Routes topology.

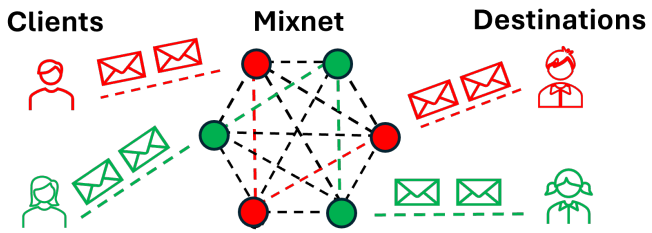


Fig. 1: Clients communicating through a Free Route mixnet.

ally dropped. Cover traffic strategies are often designed to mislead the GPA and enhance anonymity. Typically, cover traffic is added based on the input traffic to ensure that all mixnodes handle nearly the same amount of traffic. As a result, when routing in the mixnet is more restricted, the input traffic for most nodes is similar due to fewer diverse routes. In contrast, less restricted routing results in mixnodes with diverse input traffic. Consequently, the cascade topology allows for substantial integration of cover traffic but generally provides lower levels of anonymity compared to other configurations. Free Routes offer the highest degree of anonymity but are less effective at incorporating cover traffic. The stratified topology, however, represents an optimal design that maximizes anonymity while minimizing overhead [9]. Due to these advantageous characteristics, the stratified network topology has recently gained in popularity [8].

Although the stratified topology is recognized as the optimal mixnet topology for anonymizing communications, this type often requires decentralized parties to decide on numerous aspects, such as the number of mixnodes at each layer, the specific layer each mixnode should occupy, and the protocols for how, when, and where cover traffic should be added and removed. This coordination is often managed by external entities involved in mixnets, like NYM[3], which necessitates that clients wishing to join the network must subscribe and be verified by these parties to ensure they are authorized to access the network [8]. Specifically, such a design can become prohibitive in cost for a local group of clients with few mixnodes that wish to establish a local mixnet. In these cases, the best option is the Free Routes topology, which offers the highest level of anonymity with a simpler structure, eliminating the need for an external party. However, as Free Routes are less effective for incorporating cover traffic, it is crucial to ensure sufficient traffic volume when using this structure.[4]

Design Goals. Despite the low-cost and easy deployment characteristics of Free Routes mixnets, there has been limited effort to optimize message routing

[3] `https://nymtech.net`

[4] One pertinent example of this case is opportunistic social networks, where users utilize mobile devices to communicate and share data without a centralized control point. To achieve greater anonymity in these networks, the Free Routes topology of mixnet is recommended [6].

within this framework. This paper aims to provide optimized routing for Free Routes mixnets, with a focus on reducing latency. Latency optimization is crucial in mixnets as any configuration of mixnets inherently increases latency. This is due to the forwarding of clients' messages through multiple mixnodes before reaching their final destinations, with additional delays incurred by mixnodes for mixing purposes. For example, Fig. 1 illustrates a Free Routes mixnet topology where clients first forward their messages to the mixnet, incurring client-mixnet link delay. The messages are then forwarded through mixnodes, incurring both mix-link delays and mixing delays. Finally, the messages are sent to the destination, leading to mix-destination link latency. To reduce latency in such scenarios, the selection of mixnodes within the mixnet can be strategically biased towards low-latency paths where most mixnodes are either close or at least not excessively far apart, thus facilitating faster connections. Therefore, our goal is to design a mixnet routing strategy that reduce link delays within mixnet while preserving a high degree of anonymity and ensuring balanced load distribution among mixnodes.

Related Works. Despite investigating latency optimization of Free Route mixnets in this work, significant efforts to reduce latency in anonymous communications have predominantly focused within Tor networks [1, 13, 23]. These studies, despite being centered on Tor, could potentially inform strategies for Free Routes mixnets, although the differences in threat models between Tor networks and mixnets make them only indirectly applicable. For example, Lastor [1] introduces low-latency methods that suggest partitioning the network into regions where each contains a few Tor relays. Clients successively select the closest region, and within it, they randomly choose their relay. However, this approach has two main issues. First, partitioning the network in such a manner, assuming dynamic changes, is not entirely realistic, especially for a mixnet where the primary threat is considered to be a GPA rather than AS-level adversaries. Secondly, their method could lead to an imbalance of traffic distributions in the mixnet, which should be addressed to maintain network efficiency.

In other research, ShorTor [13] investigates the inefficiencies of the Border Gateway Protocol (BGP) in optimizing for the shortest or lowest latency paths, suggesting that sometimes routing through intermediate nodes rather than direct transmission can accelerate connections. This principle, similar to methods used by Content Delivery Networks (CDNs) to reduce latency, implies forming a multi-hop overlay network atop the Tor network. Although potentially useful for mixnets, this approach would require significant infrastructure modifications for adaptation to mixnets. Moreover, the paper does not thoroughly address load balancing in such configurations.

Another particularly innovative approach, CLAPS [23], potentially can enhance location-aware schemes in Tor by employing linear programming to optimize routing for reduced latency, targeted reduction in ASes correlation attacks, or improved relay resilience against IP hijacking. While CLAPS can be adapted to mixnet applications with some changes and also provides load balancing, it

faces high computational costs due to the nature of linear programming. In the best case, using Interior Point Methods, the complexity of linear programming is $O(n^3 b)$, where n represents the number of constraints, reaching up to a million in the case of Tor, and b represents the number of bits needed to represent numbers.[5]

Within mixnet research, LARMix [22] stands out as the only initiative that has explicitly targeted reducing end-to-end latency. LARMix investigates a stratified mixnet model and proposes a node selection scheme that ensures a diverse assignment of mixnodes across network layers, incorporating a sufficient number of nodes from varied jurisdictions. Additionally, their routing formula prioritizes proximity in node selection along the message paths, regulated by a tunable parameter $0 \leq \tau \leq 1$, which strategically balances latency reduction with anonymity and load balancing. However, this approach is specifically designed for stratified mixnets and does not directly translate to Free Route mixnets. Building upon these findings, our research aims to extend the results of LARMix to develop a low-latency Free Route mixnet, adapting the principles to fit a different network structure.[6]

Our Contributions. To develop low-latency routing for mixnets with a Free Routes topology, we consider a setup where there are N available mixnodes, and each client selects L distinct mixnodes to form message paths.[7] Building on this framework, we first adapt the LASTor [1] and LARMix [22] routing equations to our scenario, specifically by tuning the parameter $0 \leq \tau \leq 1$ to manage latency effectively. Applying these routing policies can result in some mixnodes receiving a higher traffic rate than normal. To counteract this, we have developed the Rebalancing Load Distribution (RLD) algorithm, which adjusts routing policies in mixnets to ensure traffic is fairly distributed among mixnodes while minimally impacting the low-latency property of the routing. Implementing RLD incurs a computational expense of $O(kN^2)$ when k is usually much less than N. On the other hand, it also curtails the advantage of adversaries who could potentially exploit the routing strategies from LARMix or LASTor to corrupt mixnodes in positions that receive a high proportion of traffic.[8]

Furthermore, we incorporate a linear programming approach inspired by the CLAPS framework [23] to derive a routing policy that prioritizes low-latency paths. We make this linear programming tunable with the parameter τ, introducing a tunable load balancing option for CLAPS. Here, $\tau = 0$ represents the

[5] Other location-aware Tor strategies [18, 26, 29] primarily focus on limiting the visibility of ASes or ISPs, or strengthening the resilience of Tor relays against active IP hijacking. These considerations are less relevant to mixnets, given the assumption of a GPA and their non-applicability for low-latency routing in mixnets.

[6] CLAM [21] is another similar work to LARMix, designed to extend LARMix's results to include low-latency message forwarding from clients to the mixnet.

[7] Note that clients can generally choose more or fewer than L hops; however, in practice, to ensure that the load received by mixnodes is balanced and that anonymity and latency are controlled, the number of hops is considered a fixed parameter.

[8] This type of attack is known as Guard Replacement Attacks in Tor [29].

most biased routing with the least load balance, while $\tau = 1$ signifies a fully balanced approach, less biased towards low-latency paths. Although this approach is computationally demanding, it effectively reduces latency while maintaining balanced load distribution across mixnodes.

We evaluate the developed routing strategies in two phases. **Firstly**, we perform an analytical assessment, similar to the approach taken by LARMix, by measuring the entropy of the distinguishing likelihood of message paths as an indicator of analytical anonymity, and by measuring the average link delay caused by the mixnet as analytical latency. **Secondly**, we simulate a Free Route mixnet using the discrete event simulator *SimPy* [20] in *Python*, where we measure the end-to-end latency (caused by both link delays and mixing processes) and the entropy of messages based on the algorithm in [3]. Our experiments, under the constraint of a mixnet with 100 nodes and modeling the link delay between mixnodes using the RIPE Atlas dataset [25], show that using an adapted version of LARMix routing can reduce the link delay latency by up to 61% compared to uniform routing while compromising at most 1 bit of message entropy. Further, applying the RLD algorithm to balance the mixnet results in a 40% reduction of this latency while losing less than 0.5 bits of entropy and ensuring fair load distribution among the mixnodes. Additionally, experiments suggest that using CLAPS-based routing can achieve latency on par with LARMix while maintaining load balance, albeit at the cost of high computational demands and reducing anonymity dramatically.

Finally, we provide an analysis to ensure that our modifications do not significantly enhance the capabilities of mixnode adversaries who corrupt (own) some mixnodes in the mixnet and, in collaboration with GPA, attempt to deanonymize client connections. This analysis considers random corruption of mixnodes as well as corruption of mixnodes from a particular location, as suggested by LARMix [22]. Furthermore, we have developed the Intelligent Corruption algorithm (IC), which allows adversaries to strategically control mixnodes to maximize their advantage, measured in terms of the fraction of fully corrupted paths (FCP). Our findings reveal that in the worst-case scenario, while maintaining a balanced mixnet, a mixnode adversary can corrupt paths at a rate twice as high as that observed with uniform routing.

2 Approach

This section details routing strategies aimed at reducing latency within a Free Routes mixnet. This involves a scenario where N mixnodes are available, and for each message route, L must be selected. To better understand this, consider an example depicted in Fig 2. In this example, $N = 4$ and $L = 3$. A client willing to form a path for her message first picks the initial mixnode (1st hop) uniformly at random as M_1. Based on this choice, the client then selects the next mixnode (2nd hop) by considering the latency from the first mixnode M_1 to the unselected ones. If M_3 is chosen for the 2nd hop, this process is repeated for selecting the third mixnode (3rd hop) by evaluating the latency between M_3

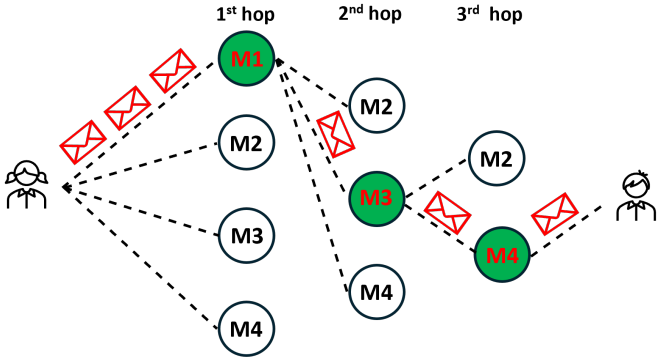


Fig. 2: Example of mixnode selection in a Free Routes mixnet.

and the two remaining mixnodes (M_2 and M_4). Considering such scenarios, we begin by adapting both LASTor [1] and LARMix [22] routing strategies for the Free Routes topology. We then enhance these strategies with load balancing by developing the RLD algorithm. Furthermore, we introduce a linear programming approach inspired by CLAPS [23] to optimize low-latency routing.

2.1 LASTor Routing

We adapt LASTor routing as shown in Eq. (1), defining the routing strategy from node i at hop k to any node j at hop $k+1$. This formula specifically ties the selection of the next mixnode to the inverse of its latency (l_{ij} represents the latency between node i and j), implying that the closer the node, the higher the probability it will be selected. In other words, Eq. (1) specifies R_{ij}^k as the probability of forwarding a message from mixnode i at the k^{th} hop to mixnode j at the $(k+1)^{th}$ hop, given that j belongs to S_k, where S_k is the set of available mixnodes for selection at hop $k+1$, excluding any mixnodes that were previously chosen for hop 1 to k.

In Eq. (1), the parameter τ moderates the level of bias: when $\tau = 0$, the routing is fully biased towards the inverse of latency, whereas increasing τ makes the routing more uniform. Specifically, when $\tau = 1$, the routing becomes uniformly random among all mixnodes that can be selected as the next hop. For instance, as depicted in Fig. 2, consider the latency between M_3 and M_2 is 10 ms, and between M_3 and M_4 is 5 ms; when $\tau = 0$, M_4 will be chosen twice as often as M_2 on average with LASTor routing.

$$R_{ij}^k = \frac{\frac{1}{l_{ij}}^{(1-\tau)}}{\sum_{j \in S_k} \frac{1}{l_{ij}}^{(1-\tau)}}. \tag{1}$$

2.2 LARMix Routing

LARMix routing formula can also be adapted for Free Routes mixnets as described in Eq. (2), showing R_{ij}^k for choosing mixnode j at hop $k+1$. Similar to LASTor, LARMix prioritizes mixnodes with lower latency by considering the inverse of latency raised to the power of $1-\tau$. Additionally, LARMix introduces a ranking function f_{ij}, which receives nodes i and j and assigns a rank indicating the closeness of j to i, starting from 0 for the closest mixnode to $|S_k|-1$ for the farthest mixnode. This ranking mechanism ensures that as τ approaches 0, the selection of the closest mixnode occurs with a probability approaching 1, effectively tuning the routing from a fully deterministic choice of the closest mixnode to a uniform routing when $\tau = 1$. This flexibility allows for tuning bias in the network design according to specific requirements. For example, consider the scenario of routing from the 2nd to the 3rd hop in Fig 2. If $\tau = 0$ and $f_{34} = 0$, M_4 would be selected with a probability of 1 as the 3rd hop.

$$R_{ij}^k = \frac{\left(\frac{1}{e}\right)^{f_{ij}\frac{(1-\tau)}{\tau}} \cdot \left(\frac{1}{l_{ij}}\right)^{(1-\tau)}}{\sum_{j\in S_k}\left(\frac{1}{e}\right)^{f_{ij}\frac{(1-\tau)}{\tau}} \cdot \left(\frac{1}{l_{ij}}\right)^{(1-\tau)}}. \tag{2}$$

2.3 Rebalancing Load Distributions (RLD)

After biasing the routing using either the LASTor or LARMix formulas in a mixnet, some mixnodes receive loads greater than those typically experienced under a uniform routing policy. If not addressed, this imbalance can confer an advantage to adversaries by allowing them to deploy high-capacity mixnodes capable of handling significant traffic, potentially leading to substantial de-anonymization of network traffic. This type of threat in the Tor network is known as Guard Replacement Attacks [29]. To mitigate this, the load distribution must be rebalanced.

The concept of balancing loads was first introduced in LARMix [22], using a Greedy approach to ensure network rebalancing while considering the prioritization of low-latency paths. However, this approach is specific to stratified mixnets and is not applicable to Free Routes mixnets. In this context, we introduce a Rebalancing Load Distributions (RLD) algorithm that effectively balances network load while still prioritizing low-latency routings.

Before explaining the RLD, it is important to note that clients are supposed to pick the first hop uniformly at random, similar to the assumption in LARMix [22]. Additionally, we suppose each node is equally likely to be in each position (each hop). Based on this, we balance the received load from each position by the mix nodes. To this end, we define $\mathbf{R}^k$ as the routing matrix from hop k to hop $k+1$, where $\mathbf{R}^k = \left[R_{ij}^k\right]$. Initially, we start with balancing $\mathbf{R}^1$. For $\mathbf{R}^1$, as each mixnode should be used once in a message-route, we set $R_{ij} = 0$ if $i = j$. To rebalance this matrix, we begin by summing up each column, which shows the received loads by mixnodes at the second hop. Columns with a summation

greater than 1 are considered overloaded, those with a summation less than 1 are underloaded, and those with a summation of 1 are balanced.

The RLD algorithm begins by addressing the overloaded columns, attempting to balance them by multiplying each entry in those columns by the inverse of the column's total summation. The surplus load from these columns is then redistributed to underloaded columns. This redistribution takes into account each entry individually and considers the proximity of overloaded mixnodes to underloaded ones. This ensures that the surplus load is distributed based on the probability of message forwarding from overloaded nodes to underloaded nodes. The process is iteratively repeated for all overloaded mixnodes until all columns sum to 1, achieving balanced routing from the first hop to the second hop. However, an exception to this process arises when there is only one underloaded mixnode remaining. In such scenarios, the rebalancing is adjusted to avoid loops or redundant paths. For example, consider that we have only the jth node underloaded at the second hop, while the ith node is overloaded. In this situation, any excess traffic from node i should not be redirected to node j if node j initially transferred this traffic to node i at the first hop, even if j is underloaded. This precaution prevents considering a node multiple times in the same message route, thereby preserving the integrity of the routing paths and avoiding loops.

As an example, consider Fig 2, where we have four mixnodes at the first hop. Suppose after applying the low-latency formula, the initial routing matrix is described as:

$$\mathbf{R}^1 = \begin{bmatrix} 0 & 0.2 & 0.4 & 0.4 \\ 0.1 & 0 & 0.6 & 0.3 \\ 0.5 & 0.5 & 0 & 0 \\ 0.3 & 0.3 & 0.4 & 0 \end{bmatrix}.$$

The summation of columns in this matrix is [0.9, 1, 1.4, 0.7], indicating that the second mixnode in the second hop is balanced, the third is overloaded, and the first and fourth are underloaded. To begin balancing the matrix, we address the overloaded third column by multiplying each entry by $\frac{1}{1.4}$, adjusting the entries to $0.29, 0.43, 0$, and 0.29 respectively.

Now, we have an excess of 0.4 from the third column that should be distributed over the first and fourth columns. Starting with the first entry of the third column, which has a surplus of 0.11 (i.e., $0.4 - 0.29$), it cannot be transferred to the first column due to the diagonal constraint $R_{ij} = 0$ for $i = j$, so it is allocated to the fourth column. As for the second entry of the third column, it has a surplus of 0.18 (i.e., $0.6 - 0.42$), which will be distributed between the first and fourth columns, with the fourth column receiving three times more ($\frac{0.3}{0.1} = 3$) due to its higher probability. The third column's third entry has no surplus as it is a diagonal entry, and the fourth entry has a surplus of 0.11, which is allocated totally to the first column, also because of the diagonal constraint. After first iteration of rebalancing, the matrix looks as follows:

$$\mathbf{R}^1 = \begin{bmatrix} 0 & 0.2 & 0.29 & 0.51 \\ 0.145 & 0 & 0.42 & 0.435 \\ 0.5 & 0.5 & 0 & 0 \\ 0.41 & 0.3 & 0.29 & 0 \end{bmatrix}.$$

However, in the new matrix, the first column is slightly overloaded, and the fourth column is slightly underloaded. To further balance this, we note that the sum of the first column is 1.055. The elements of this column should be multiplied by $\frac{1}{1.055}$, but since we have only one underloaded column, we avoid distributing the surplus to the diagonal element of the fourth column. Instead, we do not touch the 0.41 entry, and we balance the other entries of the first column by a new balancing condition, which should be $1 - 0.41 = 0.59$. Therefore, all the other entries, except for the 0.41 entry, will be multiplied by $\frac{0.59}{0.145+0.5=0.645}$, and their leftovers will be transferred to the corresponding underloaded entries to achieve a fully balanced matrix.

$$\mathbf{R}^1 = \begin{bmatrix} 0 & 0.2 & 0.29 & 0.51 \\ 0.133 & 0 & 0.42 & 0.457 \\ 0.457 & 0.5 & 0 & 0.033 \\ 0.41 & 0.3 & 0.29 & 0 \end{bmatrix}.$$

Furthermore, we note that a similar process can be applied to $\mathbf{R}^k$ beyond the first hop. However, there are additional considerations for these subsequent matrices. For example, when balancing $\mathbf{R}^2$ (the same scenario as in Fig. 2), we must consider that there were four choices for the first hop. Thus, balancing the matrix $\mathbf{R}^2$ should take this into account. That is, if the first hop is M_1, then $\mathbf{R}^2$, the matrix that indicates probability distribution from the 2nd to the 3rd hop, should reflect this by having both diagonal and one additional entry at each row set to zero. This means that if M_1 is the first hop, the second, third, or fourth entries will be zeroed to prevent the repetition of sending messages through the same mixnodes multiple times, and the remaining balancing will proceed as before.

2.4 CLAPS Mix Routing

In this section, we introduce a linear programming approach tailored for designing low-latency routing in Free Routes mixnets. This methodology draws inspiration from the CLAPS framework, originally developed to enhance the resilience of Tor relays against attacks by malicious ASes [23]. Unlike CLAPS, our primary objective is to minimize end-to-end latency rather than improve resilience. Thus, we focus on minimizing the average latency between successive mixnodes, starting with the initial routing matrix, $\mathbf{R}^1$ (shown in Eq. (3)).

To achieve the low-latency routing, we iteratively optimize subsequent routing matrices, such as $\mathbf{R}^2$, enforcing constraints that ensure each matrix preserves the probability of node selection across the first and second hops. Additionally, we incorporate constraints to ensure that diagonal elements of $\mathbf{R}^k$ are zero (to

prevent routing loops), and that the mixnet maintains balanced loading, modulated by the parameter τ. At $\tau = 1$, the routing within the mixnet achieves balance,[9] although not necessarily uniformity, diverging from approaches like LARMix and LASTor. Conversely, at $\tau = 0$, the routing is least balanced but potentially most optimized.[10]

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} R_{ij}^{1} l_{ij}, \\ \text{subject to} \quad & \forall i,j, \quad 0 \leq R_{ij}^{1} \leq 1, \\ & \forall i,j, \quad R_{ij}^{1} = 0 \text{ if } i = j, \\ & \forall i, \quad \sum_{j=1}^{N} R_{ij}^{1} = 1, \\ & \forall j, \quad \tau \leq \sum_{i=1}^{N} R_{ij}^{1} \leq N - \tau(N-1). \end{aligned} \tag{3}$$

Finally, we note that the mentioned linear programming problem can be solved using Interior Point Methods, which operate in $O(n^3 b)$, where b is the number of bits required for the representation of the numbers and n is the number of constraints. Based on Eq. (3), $n = N^2+2N$, leading to a complexity of $O(N^6 b)$. However, the RLD algorithm is much more efficient than linear programming. In this algorithm, for each overloaded node, we need at most $O(N)$ operations to balance it, which should be done in the worst case for at most $N-1$ nodes when only one node is underloaded. This process must be repeated K times, where K is generally much less than N, leading to a complexity of $O(N^2 K)$.

Furthermore, we note that clients executing routing algorithms and the RLD do not need to rely on a third party. Instead, through a decentralized and reliable approach proposed in Verloc [15], mixnodes measure their latency from one another. This data is collected and made publicly available, ensuring that clients only need to trust the majority of honest mixnodes. With these reliable latency measurements, clients with sufficient computational resources can run routing algorithms to derive the routing matrix.[11] However, in cases where the computational load exceeds the client's resources, the client can outsource the computations. In this scenario, a third party or a group of third parties, such as the network provider (like NYM) or some of the mixnodes themselves, compute the routing matrix and provide proof by employing a SNARK like Groth16 [12]. This is based on a circuit that receives the latency measurements as input and outputs the routing matrix. This ensures that the calculations derived by the

[9] Setting $\tau = 1$ actually obviates the need for using RLD algorithm.

[10] It is important to acknowledge that linear programming approaches are computationally intensive. However, investigating whether this computational effort can effectively reduce latency can be useful.

[11] Note that Verloc is deployed in NYM.

designated parties are valid. Importantly, this SNARK can be optimized since achieving zero-knowledge property is unnecessary; only the succinctness of proofs is required.

3 Evaluation

In this section, we evaluate the average link delay within a mixnet, which is promised to be reduced with the introduced routing methodologies discussed in Section 2. Additionally, we assess how much anonymity is compromised in terms of increasing the advantage of a GPA while employing low-latency routing methods.

3.1 Experimental Setup

We consider a Free Route mixnet topology with $N = 100$ mixnodes, where each client selects $L = 3$ mixnodes to form their message routes.[12] To model the link delay between network mixnodes, we utilize the RIPE Atlas dataset [25], as its globally scattered endpoints offer a representative model of latency.

To measure the average latency, we first compute all possible paths in the mixnet, which total $N(N-1)(N-2)$ considering $L = 3$. We then calculate the probability of each path using the routing matrix $\mathbf{R}^k$ and subsequently derive the average latency as the weighted sum of the latencies of these paths. Additionally, we consider the entropy of the probability distribution of these paths as $\mathsf{H}(P)$, representing the entropy of paths. This metric indicates how biased routing increases the advantage of a Global Passive Adversary (GPA). When the routing within the mixnet is uniformly random, the GPA has no better guess than a uniform distribution for the paths selected by the clients. However, biased routing can improve the GPA's guess.[13]

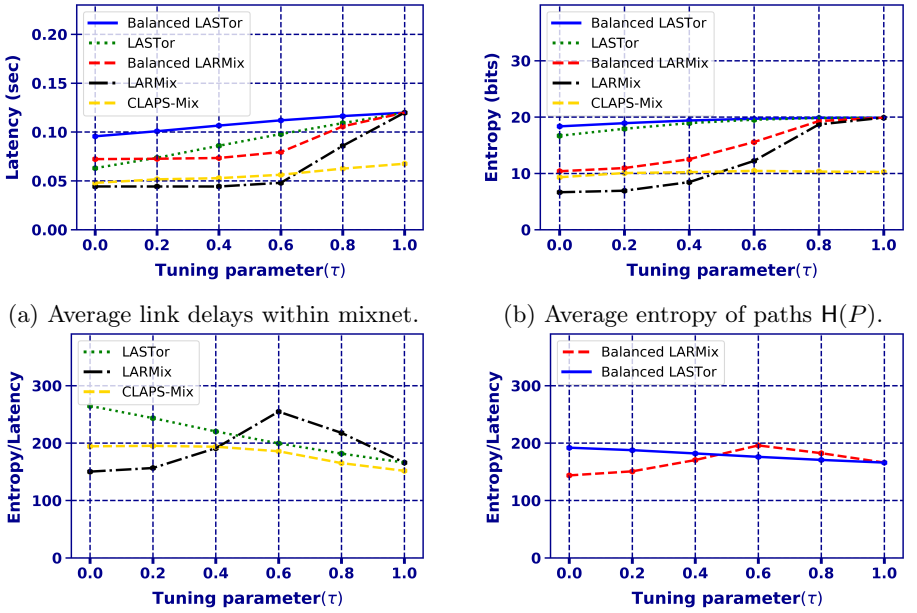
For instance, in the case of uniform routing, the entropy of paths $\mathsf{H}(P)$ equals $\log(N(N-1)(N-2))$, which approximates $3\log(N)$ for $L = 3$ and $N = 100$, giving $\mathsf{H}(P) = \log(10^6) \approx 20$. This metric allows us to quantify how anonymity is compromised using different routing bias approaches.

3.2 Latency

We initiate our analysis by examining the average link delay, as depicted in Fig. 3a. This figure illustrates the impact of varying τ on latency in mixnet link delays when employing different routing algorithms: LASTor, LARMix, and

[12] Usually, in Tor [11] or the NYM network [8], the number of hops is set to 3 to manage latency while maintaining sufficient anonymity, and $N = 100$ as it is within the range of the number of nodes in each layer in deployed mixnets [8].

[13] A similar concept was used in LARMix to measure the advantage of such an adversary, albeit with an abstract consideration of the probability of connecting the entry mixnodes to the exit mixnodes.



(a) Average link delays within mixnet.

(b) Average entropy of paths $\mathsf{H}(P)$.

(c) Entropy/Latency for imbalance routings.

(d) Entropy/Latency for balanced routings.

Fig. 3

CLAPS Mix, under both balanced and imbalanced network conditions. Specifically, an increase in τ consistently results in higher latency across all strategies. In LASTor and LARMix, τ acts as a tuning parameter that injects a varying amount of randomness into the routing policies. This randomness influences routing decisions by deprioritizing paths with lower latency, thereby increasing the average link delay. In contrast, in CLAPS Mix, τ primarily functions as a bias control factor, rather than a randomness factor. Adjusting τ in CLAPS Mix affects the network's balance rather than its randomness. As a result, higher τ values lead to a preference for more balanced paths, which naturally exhibit higher latencies compared to faster, less balanced alternatives. This trend also holds for balanced configurations in both LARMix and LASTor, where balanced setups tend to experience greater latencies than their imbalanced counterparts.

Additionally, when routing is uniform ($\tau = 1$) for LASTor and LARMix, the average link latency is approximately 120 ms. Utilizing LARMix can reduce this latency to 46 ms, representing a 61% reduction when $\tau = 0$. When LARMix is balanced, the latency is minimized to 73 ms, achieving a 40% reduction. In contrast, using LASTor in both imbalanced and balanced configurations results in latencies of 65 ms and 98 ms, respectively. Remarkably, CLAPS Mix matches the performance of an imbalanced LARMix with $\tau = 0$. Intriguingly, when CLAPS Mix is balanced ($\tau = 1$), the latency stabilizes at 65 ms, equivalent to LAS-

Tor in its imbalanced state with $\tau = 0$. This demonstrates the efficacy of linear programming in reducing latency across various configurations while ensuring a balanced distribution of traffic.

3.3 Anonymity

Fig. 3b depicts the entropy of paths versus tuning parameter τ. In the uniform case, where $\tau = 1$ for LARMix or LASTor routing, the entropy is approximately 20 bits. Using CLAPS Mix, which provides very low latency, results in a dramatic decrease in entropy to about 10 bits for almost all values of the tuning parameter τ. This reduction could be detrimental as it increases the likelihood that an adversary can guess the paths selected by clients. In contrast, LARMix exhibits variable effects depending on the value of τ. For example, at $\tau = 0.6$, a balanced configuration of LARMix reduces the entropy to 16.5 bits while providing a latency of 75 ms. On the other hand, LASTor demonstrates the best overall performance; in both imbalanced and balanced configurations, it reduces entropy to 17 bits and 18.5 bits, respectively. For instance, with $\tau = 0$ in an imbalanced LASTor configuration, the latency is reduced to 65 ms and the entropy to 17 bits, which appears to be a favorable trade-off. Moreover, a balanced LASTor configuration offers very high entropy, but it does not reduce latency below 100 ms, which may not meet practical performance requirements.

3.4 Trade-Offs

To facilitate a better comparison of different approaches and to determine which one offers the optimal trade-off, we further explore the ratio of Entropy to Latency for both imbalanced and balanced routing configurations. These comparisons are depicted in Fig. 3c for imbalanced configurations and Fig. 3d for balanced configurations. Fig. 3c illustrates the imbalanced configurations of LASTor, LARMix, and CLAPS Mix. Notably, CLAPS Mix is always considered imbalanced, except when $\tau = 1$.

It is observed that for $\tau < 0.43$, LASTor yields the highest Entropy/Latency ratio, suggesting a favorable trade-off. In this range, LASTor's latency varies from 65 ms to 80 ms, while its entropy ranges from 17 bits to 19 bits. However, for $\tau > 0.43$, LARMix leads in performance, particularly at $\tau = 0.6$ where it reduces latency to 50 ms and achieves an entropy of 12 bits. This suggests that LARMix tends to favor lower latency, whereas LASTor prioritizes higher entropy. Conversely, CLAPS Mix does not exhibit very high performance due to its consistent entropy of approximately 10 bits across all values of τ. Given this consistent performance, CLAPS Mix may be recommended when minimizing latency is more critical than maximizing the entropy of paths in the network.

On the other hand, Fig. 3d illustrates the Entropy/Latency ratio for the balanced approaches, including Balanced LARMix and Balanced LASTor. As observed, the trend between these two algorithms is similar to their imbalanced counterparts, although the Entropy/Latency ratio in balanced approaches is slightly lower. This suggests that achieving a fair distribution of loads across

Approaches	N = 50	N = 100	N = 200
LASTor	100 ms	88 ms	82 ms
Balanced LASTor	117 ms	109 ms	105 ms
LARMix	60 ms	45 ms	25 ms
Balanced LARMix	87 ms	75 ms	68 ms
CLAPS-Mix	71 ms	53 ms	45 ms

(a) Average latency.

Approaches	N = 50	N = 100	N = 200
LASTor	16.2 bits	17.9 bits	21.8 bits
Balanced LASTor	16.5 bits	19 bits	22.3 bits
LARMix	7.6 bits	8.8 bits	9.6 bits
Balanced LARMix	11.3 bits	12.2 bits	13.8 bits
CLAPS-Mix	8.6 bits	10 bits	13.6 bits

(b) Entropy of paths ($\mathsf{H}(P)$).

Fig. 4: Effects of network size ($\mathbf{N}$) on latency and anonymity.

mixnodes comes at the expense of less favorable trade-offs. However, in balanced cases when $\tau < 0.42$, LASTor exhibits the best performance, thanks to the highest entropy provided in this interval. Nevertheless, the balanced version of LASTor cannot provide latency lower than 100 ms, making LARMix a preferable option unless the reduction in entropy is not tolerable by the clients. For $\tau > 0.42$, Balanced LARMix consistently delivers better performance, with an optimal balance achieved at $\tau = 0.6$, where it offers a substantial reduction in latency to 75 ms while maintaining an entropy of 12 bits.[14] Lastly, it is noted that at $\tau = 1$, CLAPS Mix provides a balanced version which, compared to Balanced LASTor and Balanced LARMix, is not optimal; however, it can offer very low latency, which may be advantageous in scenarios where lower latency is favored while a reduction in entropy is tolerable.

3.5 Scaling the Mixnet

Up till now, the mixnet size was set to $N = 100$ and the number of hops to $L = 3$, specifically to align with real-world deployed mixnets such as NYM. However, in this section, we analyze how increasing the network size (N) and the number of hops (L) affect both latency and anonymity. To do so, considering all introduced routing strategies, we set the tuning parameter $\tau = 0.4$. Firstly, we fixed $L = 3$ and varied the network size, recording the average link delay and entropy of paths, as illustrated in Fig. 4.

More accurately, Fig. 4a delineates the effect of network size on latency. Increasing the network size across all routing strategies results in a reduced average link latency. Specifically, when the network size increases from $N = 50$ to $N = 200$, the latency induced by LARMix shows an approximate 58% reduction. This significant effect can be attributed to the fact that a larger number of nodes increases the likelihood of having geographically closer peers, enabling clients to select paths through the mixnet that minimize latency. Thus, augmenting the number of mixnodes can potentially reduce latency.

Conversely, Fig. 4b indicates that increasing the number of active mixnodes enhances the entropy of paths. This outcome arises because a greater number of

[14] Note that this entropy is sufficient for comparing different approaches, but will not reflect the exact impact of low-latency routing on anonymity. We will examine this impact in terms of message entropy in Section 4.

Approaches	L = 2	L = 3	L = 4
LASTor	40 ms	88 ms	130 ms
Balanced LASTor	50 ms	109 ms	163 ms
LARMix	21 ms	45 ms	67 ms
Balanced LARMix	38 ms	75 ms	114 ms
CLAPS-Mix	23 ms	53 ms	72 ms

(a) Average latency.

Approaches	L = 2	L = 3	L = 4
LASTor	12.8 bits	17.9 bits	24.4 bits
Balanced LASTor	13 bits	19 bits	25.4 bits
LARMix	5.9 bits	8.8 bits	13.1 bits
Balanced LARMix	8.9 bits	12.2 bits	17.8 bits
CLAPS-Mix	7.2 bits	10 bits	14.6 bits

(b) Entropy of paths ($\mathsf{H}(P)$).

Fig. 5: Effects of number of hops ($\mathbf{L}$) on latency and anonymity.

nodes provides more options for constructing a message route, thereby increasing the entropy values.[15]

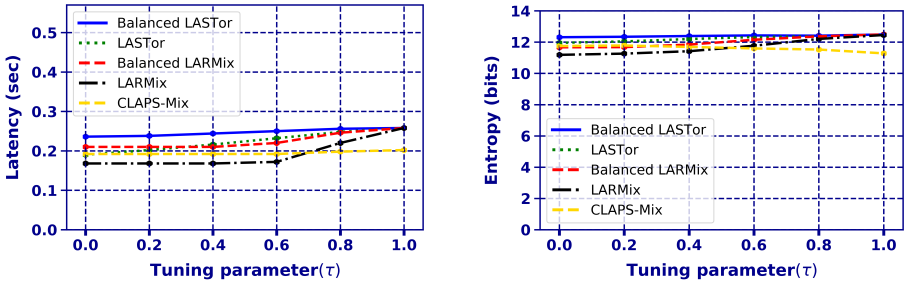
Fig. 5 illustrates the relationship between the number of hops (L) and the average link delay or entropy of paths, with the mixnet size fixed at $N = 100$. More specifically, Fig. 5a demonstrates that increasing the number of hops proportionally increases the average latency. Indeed, adding more hops necessitates longer travel distances for messages, resulting in heightened latency. Consequently, as the number of hops increases, the number of possible ways to create a message route also increases. Therefore, as depicted in Fig. 5b, increasing the number of hops consistently enhances anonymity. Hence, the number of hops is a crucial parameter that needs to be tuned to balance latency and anonymity. In networks such as Tor and NYM, this is typically fixed at $L = 3$.

4 Simulation of Free Routes Mixnet

In our analysis so far, we've examined the effects of low-latency routing on mixnet link delays and the entropy of paths ($\mathsf{H}(P)$). While these metrics are sufficient for comparing different approaches, they may not fully capture the real effect of each approach on the anonymity of messages, which is primarily determined by the mixing process. Therefore, this section focuses on such analysis by simulating the action of mixnodes within a mixnet together with clients, and assessing anonymity measures based on message entropy while also recording end-to-end latency.

To conduct this analysis, we simulate a Free Routes mixnet using *SimPy* [20], a discrete event simulator implemented in Python. We employ the mixing process via the stop-and-go mixing strategy [14], which introduces delays to input messages based on an exponential distribution with an average delay parameter μ. For our simulation, we set μ at 50 ms to mirror NYM network conditions. Additionally, the input traffic is modeled to follow a Poisson distribution with a rate parameter $\lambda = 10000$, implying that, on average, 10,000 messages enter each mixnode in the network.

[15] It is important to note that increasing the number of nodes is advantageous only if there is sufficient traffic for each node. In a network with low traffic, adding more nodes can negatively impact message anonymity. If there are more mixnodes than necessary, the traffic will be distributed among more nodes, diminishing the chances of messages being mixed with a sufficient number of other messages.



(a) Average end-to-end latency.

(b) Average entropy of messages $\mathsf{H}(m)$.

Fig. 6

We record the entering and exiting times of each message to compute end-to-end latency. Furthermore, we utilize a methodology outlined in [3] to quantify the anonymity of messages. This approach evaluates the probability that a message entering the mixnet, targeted by the adversary, can be correlated to its counterpart at the exit among other messages.[16]

4.1 Basic Results

In our mixnet simulator, we conducted two key experiments to evaluate **first**, the end-to-end latency, which encompasses both the link delay within the mixnet and the additional delay imposed by the mixing process at mixnodes; and **second**, message anonymity, quantified by $\mathsf{H}(m)$. We conducted around 100 iterations for each experiment, configuring a new mixnet each time to ensure the reliability of our results.

Fig. 6a illustrates the end-to-end latency across all routing approaches, including LASTor, LARMix in both balanced and imbalanced cases, and CLAPS Mix. We adjusted the tuning parameter τ for all approaches and observed behavior almost identical to that seen for the average link delay within the mixnet versus varying τ, with the notable exception of higher average latency. In this context, latency includes delays from mixnodes, which, with an average contribution of $50 \times 3 = 150$ ms, can be further tuned according to network requirements but typically has a negative impact on the overall end-to-end latency. With a 50 ms average exponential delay, the minimum achievable latency through imbalanced LARMix or CLAPS Mix is 85 ms, marking a 44% reduction compared to the uniform routing ($\tau = 1$), which increases latency to 250 ms. Conversely, in this scenario, Balanced LASTor behaves almost like a uniform approach without substantial improvement in latency, while imbalanced LASTor and Balanced LARMix can reduce latency to as low as 200 ms, amounting to a 20% reduction in end-to-end latency.

[16] Further details of the simulation process are provided in Appendix B.

Fig. 6b presents the entropy of messages across all routing approaches mentioned in this paper, analyzed against varying values of τ. The observations indicate that, irrespective of the τ values, the message entropy typically fluctuates around 12 bits. This consistency suggests that the real anonymity of the mixnet (specifically, the anonymity provided by the messages themselves) remains relatively stable despite the entropy of path selection showing dramatic decreases. This reveals that while $\mathsf{H}(P)$ is useful for analyzing biased routing, it does not fully capture the nuances of message anonymity. In scenarios involving biased routing, both the LARMix imbalances and CLAPS Mix can lead to a reduction of up to 1 bit in message entropy ($\mathsf{H}(m)$). Conversely, imbalanced LASTor and Balanced LARMix exhibit similar behavior, whereas Balanced LASTor slightly enhances the entropy of messages, benefiting from less biased routing toward low-latency paths. Overall, the approaches show a slight increase in message entropy as τ increases, likely due to the introduction of more randomness into the network.

4.2 Constraint on End-to-End Latency

In certain scenarios, clients utilizing mixnets may necessitate meeting an average end-to-end delay latency to ensure fast and reliable communication. In default mixnet configurations, this requirement can generally be addressed by adjusting the average latency induced by mixnodes (μ). However, this straightforward approach often leads to a significant reduction in anonymity, as the mixing delay directly influences message anonymity.

A more practical method to manage end-to-end latency involves the use of low-latency approaches introduced in this study. For instance, consider a scenario where the end-to-end latency is represented as l_{e2e}. One straightforward method to achieve this end-to-end latency is to set the mixing delay and subsequently adjust the tuning parameter for low-latency routing, seeking a combination that satisfies the end-to-end latency requirement. Although this trial-and-error method can be effective, it cannot fully optimize message anonymity while satisfying the end-to-end latency constraint. This limitation arises because increasing the tuning parameter τ improves the entropy of paths $\mathsf{H}(P)$ but also heightens the link delay within the mixnet. Consequently, there is less room for mixing delays, leading to a reduction in message anonymity.

To attain a more optimized setting, we propose an approach wherein the parameter τ is first set, followed by measuring the link delays within the mixnodes in such settings. With this information, the mixing delay can be established as $\mu = \frac{l_{\text{e2e}} - \text{link delays}}{L}$. By adjusting the value of τ, one can identify the optimal combination of τ and μ that maximizes message anonymity while adhering to the average end-to-end latency constraints.

Following this approach, we performed an experiment considering the LARMix routing for a scenario where the average end-to-end latency is set to 150 ms. In this case, we first varied the value of τ and then measured the link delays and subsequently the mixing delays for the mixnodes, with a setting of $N = 100$ and $L = 3$. The results are shown in Tab. 1. As illustrated, increasing τ increases

Table 1: Optimizing anonymity while meeting an end-to-end average latency.

Tuning parameter τ	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Link delays ms	58.0	58.1	58.15	58.2	58.3	58.3	60	70	91	108.2	121
Mix delay (μ) ms	30	30	30	30	29.9	29.9	29.8	27	19	14	10
Entropy of Messages ($\mathsf{H}(\mathsf{m})$) bits	10.1	10.2	10.21	10.42	10.8	11	11.4	11.6	11.5	11.2	10.8
Entropy of Paths ($\mathsf{H}(\mathsf{P})$) bits	6.64	6.65	6.9	7.6	8.5	9.9	12.5	16.2	18.8	19.7	19.9

the entropy of paths as well as link delays within the mixnet, consequently decreasing mixing delays. However, the anonymity of messages, influenced by both τ and mixing delays, demonstrates different behavior. When $\tau \leq 0.7$, message anonymity increases by increasing τ, but when $\tau \geq 0.7$, this metric started decreasing after further increasing τ. This suggests that the optimal combination for this end-to-end constraint is to set $\tau = 0.7$ and subsequently set the mixing delay $\mu = 27$ ms.

5 Adversarial Mixnodes

In this section, we broaden the scope of adversarial analysis in mixnets, shifting from a GPA to a more localized threat, the mixnode adversary. A GPA can observe all connections and transactions within the communication network but lacks the ability to inspect the internal operations of mixnodes or directly engage in the mixing process. Therefore, to render our analysis more practical, we explore scenarios in which the adversary is capable of corrupting parts of the mixnet by compromising specific mixnodes. Through these compromised nodes, the adversary can link input and output messages, effectively nullifying the anonymity of communications routed through these nodes. Consequently, it is crucial for client messages to pass through at least one uncorrupted mixnode to maintain anonymity.

Moreover, for an adversary to achieve comprehensive de-anonymization, control over at least L mixnodes is necessary to construct complete paths. To augment their capability to de-anonymize clients, adversaries must also increase the number of mixnodes under their control, ensuring their inclusion across numerous paths. This strategic positioning increases the probability of intercepting and compromising client communications. Within this context, we assess the adversary's effectiveness when routing preferences are skewed towards low-latency routers. In these scenarios, the primary objective for the adversary is to maximize the Fraction of Corrupted Paths (FCP), thereby ensuring a significant number of paths within the mixnet are fully compromised. In the next subsection, we will delineate some possible strategies that may be taken into account by adversaries for these purposes.

5.1 Adversarial Strategies

Random Strategy The Random Strategy is the most basic approach to corrupting mixnodes within the mixnet. Here, the adversary controls C mixnodes

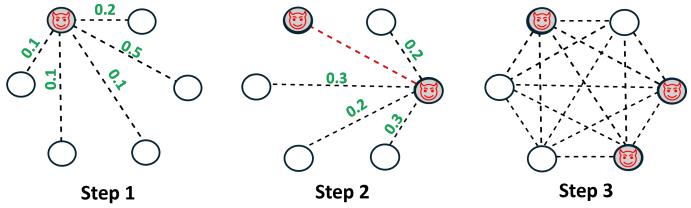


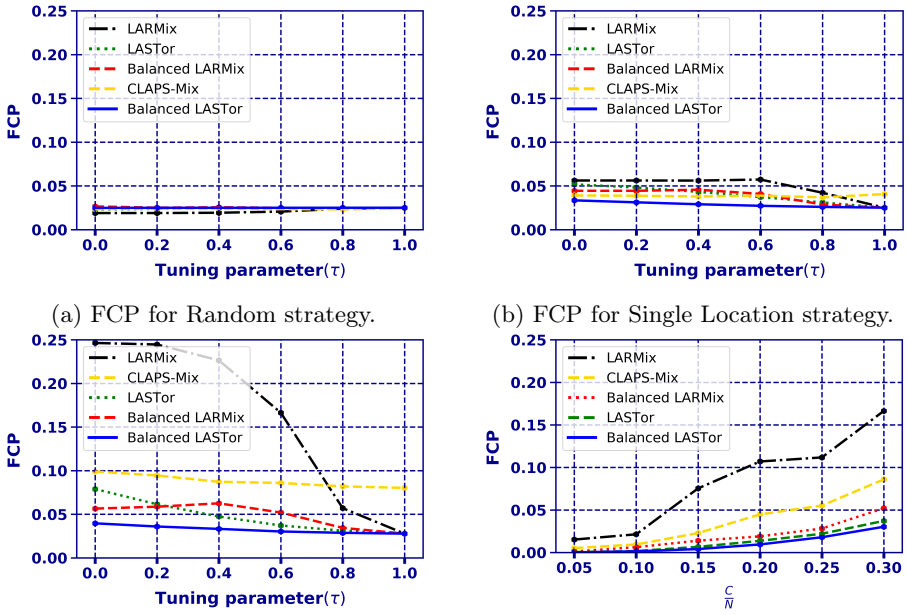
Fig. 7: Intelligent Corruption of mixnodes.

selected at random. Depending on the dynamics of selection, this approach can occasionally yield a high probability of intercepting paths that frequently incorporate these mixnodes, while at other times, it may result in less optimal node placements, targeting paths with lower selection probabilities. For instance, in a Free Routes mixnet with uniform routing, if $L = 3$ and $C = \frac{3N}{10}$ (indicating that 30% of the nodes are corrupted), the FCP would be approximately 2.5%.

Single Location Strategy Another tactical approach involves exploiting the clients' preference for low-latency routing, which often directs traffic through mixnodes located within the same jurisdiction or region. By identifying and corrupting mixnodes within these targeted regions, the adversary can significantly enhance their strategic advantage. This method is particularly effective as it raises the likelihood that messages will pass through adversarially controlled nodes, thus amplifying the potential for fully de-anonymizing clients.

Intelligent Corruption Strategy To comprehensively analyze the capability of a mixnode adversary, we introduce a practical strategy termed Intelligent Corruption, assuming the adversary has complete information about the other active mixnodes in the network. This strategy begins by randomly corrupting a mixnode within the mixnet. The adversary then assesses the distance from this possessed mixnode to all others, selecting those closest or with the highest probability of being the next hop. This process continues, choosing subsequent mixnodes that are nearest to the most recently corrupted node or those most likely to be selected next, until C mixnodes are corrupted, as detailed in Algorithm 2.

For example, as depicted in Fig 7, consider a Free Route mixnet with $N = 6$ mixnodes and an adversary's budget of $C = 3$. Initially, the adversary corrupts a random mixnode. They then evaluate the likelihood of selecting any uncorrupted mixnodes from this node, aiming to maximize the probability of routing through corrupted paths. Subsequently, the adversary corrupts the mixnode with the highest selection probability (e.g., 0.5). Next, from the second corrupted node, they assess two mixnodes, each with a selection probability of 0.3, and corrupt one of them in the final step, completing the Intelligent Corruption of the mixnet.



(a) FCP for Random strategy.

(b) FCP for Single Location strategy.

(c) FCP for Intelligent Corruption Strategy.

(d) Fraction of corrupted mixnodes.

Fig. 8

5.2 Evaluation of Adversarial Strategies

To evaluate the adversarial strategy for maximizing the FCP, we considered all routing strategies—LARMix, LASTor, and CLAPs Mix—in both imbalanced and balanced cases. We then measured the FCP by varying the value of τ and setting the fraction of corrupted mixnodes as $\frac{C}{N} = 0.3$. Fig. 8a illustrates the results of FCP derived from Random corruption of mixnodes. As observed, almost all the routing approaches yield similar results; even as τ varies, the results do not change dramatically. This consistency can be attributed to the nature of Random Corruption, where the adversary might corrupt mixnodes that are either on highly probable paths or, conversely, on paths with very low probability of selection. Therefore, on average, the amount of FCP should be comparable to that with uniform routing, previously described as 2.5%.

Fig. 8b showcases the results of employing the Single Location strategy for corrupting mixnodes within low-latency routings approaches. As the value of τ increases, the FCP generally decreases across most routing configurations. The primary reason for this trend is that increasing τ introduces greater randomness into the routing decisions of LASTor and LARMix, both in imbalanced and balanced scenarios. This increased randomness reduces the probability of selecting low-latency paths that were previously chosen with higher likelihood, thus di-

minishing the adversary's advantage when the corruption of mixnodes adheres to the Single Location strategy for all routing approaches.

In the case of CLAPS Mix, since an increase in τ does not necessarily equate to enhanced randomness, the outcomes remain relatively consistent regardless of τ value adjustments. As depicted in Fig. 8b, LARMix under imbalanced conditions is notably more susceptible to the Single Location strategy, exposing up to a 6% FCP. Conversely, both Imbalanced LASTor and Balanced LARMix exhibit performances closely aligned with that of CLAPS Mix, particularly when $\tau < 0.6$. However, there is a marked reduction in FCP for these strategies relative to CLAPS Mix once τ surpasses this threshold. Contrastingly, Balanced LASTor offers the least benefit to the adversary by maintaining a low FCP, comparable to outcomes derived from the Random Strategy. This observation underscores that the variance between low-latency and high-latency paths in this case is minimal, thereby limiting the adversary's leverage when utilizing the Single Location strategy.

In addition, applying the Intelligent Corruption strategy, as shown in Fig. 8c, reveals that, similar to the results from the Single Location strategy, increasing the value of τ generally leads to a reduction in FCP. However, this strategy results in a consistently higher FCP compared to the Single Location case. Specifically, when using Imbalanced LARMix, the FCP can reach up to 25% at $\tau = 0$, which is up to five times higher than the results observed from the Single Location strategy. This underscores the effectiveness of Intelligent Corruption, particularly when latency is a significant factor within Imbalanced LARMix scenarios.

In contrast, for other routing approaches like CLAPS Mix, the increase in FCP using the Intelligent Corruption strategy does not exceed twice the baseline. Notably, for Balanced LARMix, unlike its imbalanced configuration, the FCP result is only marginally higher than that achieved through the Single Location strategy, and it aligns closely with the results of Imbalanced LASTor. Meanwhile, Balanced LASTor exhibits results nearly identical to those of the Single Location strategy. This suggests that even though balancing the mixnet using the RLD algorithm may potentially heighten end-to-end latency, it can effectively mitigate the impact of the Intelligent Corruption strategy. This is particularly significant when compared to the higher vulnerabilities observed in the Imbalanced LARMix configuration.

Furthermore, to assess the effect of adversary budget, Fig. 8d presents the results of Intelligent Corruption when we set $\tau = 0.6$ for all routing approaches in both balanced and imbalanced cases. We then vary the fraction of corrupted nodes ($\frac{C}{N}$) from 5% to 30%. As demonstrated, increasing the adversary's budget translates into corrupting more mixnodes, thereby increasing the likelihood that adversarial mixnodes will compromise complete paths. Consequently, as seen in Fig. 8d, the FCP increases. This behavior remains consistent across various approaches; specifically, the imbalanced LARMix configuration proves to be the most vulnerable approach to increasing adversary budgets. Conversely, CLAPS Mix is the second most susceptible. Meanwhile, Balanced LARMix and imbal-

anced LASTor exhibit similar trends, and Balanced LASTor shows the greatest resilience towards increasing the adversarial budget.

6 Conclusion

In this work, we introduced the first low-latency routing methodology tailored for mixnets with a Free Routes design. We began by adapting existing routing approaches from LASTor, LARMix, and CLAPS for use in mixnet environments. Additionally, we developed the RLD algorithm that ensures each mixnode, regardless of its position within the mixnet, receives an equal amount of traffic. Our comprehensive analytical and simulation experiments demonstrate substantial latency reductions, while only compromising a negligible amount of anonymity. Moreover, our adversarial analysis reveals no significant advantage for mixnode adversaries compared to a GPA. We anticipate that this framework will prove especially useful in scenarios requiring cost-effective (or low-latency) anonymity.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable feedback. This research is partially supported by CyberSecurity Research Flanders with reference number VR20192203.

References

1. Akhoondi, M., Yu, C., Madhyastha, H.V.: Lastor: A low-latency as-aware tor client. In: 2012 IEEE Symposium on Security and Privacy. pp. 476–490. IEEE (2012)
2. Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against tor. In: Proceedings of the 2007 ACM workshop on Privacy in electronic society. pp. 11–20 (2007)
3. Ben Guirat, I., Gosain, D., Diaz, C.: Mixim: Mixnet design decisions and empirical evaluation. In: Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society. pp. 33–37 (2021)
4. Chaum, D., Das, D., Javani, F., Kate, A., Krasnova, A., De Ruiter, J., Sherman, A.T.: cmix: Mixing with minimal real-time asymmetric cryptographic operations. In: Applied Cryptography and Network Security: 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings 15. pp. 557–578. Springer (2017)
5. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2), 84–90 (1981)
6. Chen, D., Borrego, C., Navarro-Arribas, G.: A privacy-preserving routing protocol using mix networks in opportunistic networks. Electronics **9**(11), 1754 (2020)
7. Diaz, C.: Anonymity and privacy in electronic services. Heverlee: Katholieke Universiteit Leuven. Faculteit Ingenieurswetenschappen (2005)
8. Diaz, C., Halpin, H., Kiayias, A.: The nym network (2021)
9. Diaz, C., Murdoch, S.J., Troncoso, C.: Impact of network topology on anonymity and overhead in low-latency anonymity networks. In: Privacy Enhancing Technologies: 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings 10. pp. 184–201. Springer (2010)

10. Diaz, C., Preneel, B.: Taxonomy of mixes and dummy traffic. In: Information Security Management, Education and Privacy: IFIP 18th World Computer Congress TC11 19th International Information Security Workshops 22–27 August 2004 Toulouse, France. pp. 217–232. Springer (2004)
11. Dingledine, R., Mathewson, N., Syverson, P.F., et al.: Tor: The second-generation onion router. In: USENIX security symposium. vol. 4, pp. 303–320 (2004)
12. Groth, J.: On the size of pairing-based non-interactive arguments. In: Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35. pp. 305–326. Springer (2016)
13. Hogan, K., Servan-Schreiber, S., Newman, Z., Weintraub, B., Nita-Rotaru, C., Devadas, S.: Shortor: Improving tor network latency via multi-hop overlay routing. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 1933–1952. IEEE (2022)
14. Kesdogan, D., Egner, J., Büschkes, R.: Stop-and-go-mixes providing probabilistic anonymity in an open system. In: International Workshop on Information Hiding. pp. 83–98. Springer (1998)
15. Kohls, K., Diaz, C.: {VerLoc}: Verifiable localization in decentralized systems. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 2637–2654 (2022)
16. Kwon, A., Lu, D., Devadas, S.: {XRD}: Scalable messaging system with cryptographic privacy. In: 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20). pp. 759–776 (2020)
17. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster protocol—version 2 (2003)
18. Nithyanand, R., Starov, O., Zair, A., Gill, P., Schapira, M.: Measuring and mitigating as-level adversaries against tor. arXiv preprint arXiv:1505.05173 (2015)
19. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th *USENIX* Security Symposium (*USENIX* Security 17). pp. 1199–1216 (2017)
20. Python: Event discrete, process based simulation for python. `https://pypi.org/project/simpy/` (2013)
21. Rahimi, M.: CLAM: client-aware routing in mix networks. In: Pérez-González, F., Alfaro, P.C., Krätzer, C., Zhao, H.V. (eds.) Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2024, Baiona, Spain, June 24-26, 2024. pp. 199–209. ACM (2024). `https://doi.org/10.1145/3658664.3659631`, `https://doi.org/10.1145/3658664.3659631`
22. Rahimi, M., Sharma, P.K., Diaz, C.: Larmix: Latency-aware routing in mix networks. In: The Network and Distributed System Security Symposium. Internet Society (2024)
23. Rochet, F., Wails, R., Johnson, A., Mittal, P., Pereira, O.: Claps: Client-location-aware path selection in tor. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 17–34 (2020)
24. Shirazi, F., Simeonovski, M., Asghar, M.R., Backes, M., Diaz, C.: A survey on routing in anonymous communication protocols. ACM Computing Surveys (CSUR) **51**(3), 1–39 (2018)
25. Staff, R.N.: Ripe atlas: A global internet measurement network. Internet Protocol Journal **18**(3), 2–26 (2015)
26. Sun, Y., Edmundson, A., Feamster, N., Chiang, M., Mittal, P.: Counter-raptor: Safeguarding tor against active routing attacks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 977–992. IEEE (2017)

27. Sun, Y., Edmundson, A., Vanbever, L., Li, O., Rexford, J., Chiang, M., Mittal, P.: {RAPTOR}: Routing attacks on privacy in tor. In: 24th USENIX Security Symposium (USENIX Security 15). pp. 271–286 (2015)
28. Van Den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: Scalable private messaging resistant to traffic analysis. In: Proceedings of the 25th Symposium on Operating Systems Principles. pp. 137–152 (2015)
29. Wan, G., Johnson, A., Wails, R., Wagh, S., Mittal, P.: Guard placement attacks on path selection algorithms for tor. Proceedings on Privacy Enhancing Technologies **2019**(4) (2019)

A Algorithms and Notations

In this section, we introduce the algorithms and notations used in this paper.

Algorithm 1 Rebalancing Load Distributions

Input: Routing Matrix $\mathbf{R}^k$
Compute the summation of columns in $\mathbf{R}^k$.
for each column j in $\mathbf{R}^k$ **do**
  if $\sum_i R^k_{ij} > 1$ **then**
    Label column j as overloaded.
  else if $\sum_i R^k_{ij} = 1$ **then**
    Label column j as balanced.
  else
    Label column j as underloaded.
  end if
end for
for each column j labeled as overloaded **do**
  Multiply each entry in column j by the inverse of its sum.
  Distribute the surplus probabilities to underloaded columns,
    prioritizing based on their proximity.
  Exclude diagonal entries from redistribution.
end for
if there is exactly one underloaded column **then**
  Rebalance by considering non-diagonal entries.
end if
if balance is not achieved **then**
  Repeat the rebalancing procedure.
end if
Output: Rebalanced matrix $\mathbf{R}^k$.

B Expanding on Simulation of Free Routes Mixnet

B.1 Simulation Settings

As outlined in Section 4, we conducted simulations of a Free Routes mixnet using the discrete event simulator *SimPy* in *Python*. Discrete event simulators

Algorithm 2 Intelligent Corruption

Input: Routing Matrix $\mathbf{R}$, Total Mixnodes N
Initialize: Counter $\leftarrow 0$
Initialize: Corrupted_Set $\leftarrow$ empty set
Randomly pick one mixnode and add to Corrupted_Set
Counter $\leftarrow$ Counter + 1
while Counter $<$ C **do**
    Identify Mix_{next} where $\mathbf{P}(Mix_{\text{next}}|Mix_{\text{current}})$ is maximized
    Add Mix_{next} to Corrupted_Set
    Counter $\leftarrow$ Counter + 1
end while
Output: Corrupted set of mixnodes

Table 2: Notations used in this paper.

Parameter	Meaning
N	Size of the network
L	Number of hops
$\mathbf{R}^k$	Routing matrix at hop k
R_{ij}^k	Routing probability between node i at hop k and j at hop $k+1$
μ	Average mixing delay
λ	Poisson parameter of incoming message rate
l_{ij}	Latency between mixnode i and j
τ	Tuning parameter
$\mathsf{H}(P)$	Entropy of path selection
$\mathsf{H}(m)$	Entropy of message
f	Rank function
S_k	Set of available mixnodes at hop $k+1$

are particularly suitable for modeling mixnets because they inherently handle randomness and can effectively simulate the dynamic nature of network events, such as message transmission through mixnodes.

In our simulations, we employed the stop-and-go mixing strategy, which delays input messages according to an exponential distribution $\mathsf{E}(\frac{1}{\mu})$ with an average delay parameter μ, set at 50 ms. This strategy is a common choice due to the high anonymity it provides, stemming from the memory-less property of the exponential distribution. Specifically, this means that for any two messages arriving at different times, the probability of them exiting the mixnode at any subsequent time is the same, thereby enhancing their anonymity.

Additionally, such mixnodes have the manageable average latency, an option not available in other mixnodes' types that their flushing time depends on other input messages. After setting up the mixnet with these mixnodes, we simulated clients sending messages to the mixnet. Each client sends messages according

to a Poisson distribution with a rate parameter λ, meaning that, on average, λ messages are sent to the mixnet every second. The simulator then applies a routing strategy to route the packets while also measuring the latency based on the RIPE dataset provided for these mixnodes.

We compute two primary parameters in simulations: the average end-to-end latency for each message entering and exiting the mixnodes, using the *simpy.env.now()* command to capture the current simulation time. Furthermore, we assess the anonymity of each message, a critical metric for mixnets. For this, we employed the approach described in [3], which defines anonymity in a scenario where an adversary targets an incoming message and probabilistically determines its counterpart exiting the mixnet. Actually in this method, the first targeted message entering the mixnet is assigned a probability of 1 of being targeted by the adversary. Due to the memory-less nature of the distribution, this message is mixed with others, and any outgoing message may potentially be the targeted message with a reduced probability, dependent on the number of incoming messages. This probability is updated as the message progresses through each mixnode, and ultimately, we compute the entropy of the targeted message being indistinguishable among all outgoing messages. In our simulations, we considered 200 targeted messages. Finally, we compared different routing approaches to describe the variance in anonymity and latency metrics achieved during the simulations.

B.2 Detailed Simulation Results

Table 3: Detailed analysis of end-to-end latency.

Approaches	$\tau = 0$			$\tau = 0.6$			$\tau = 1$		
Statistics	**Average**	**Variance**	**CI**	**Average**	**Variance**	**CI**	**Average**	**Variance**	**CI**
LASTor	0.20	0.01	$[0.20, 0.204]$	0.24	0.01	$[0.240, 0.243]$	0.26	0.01	$[0.26, 0.267]$
Balanced LASTor	0.24	0.01	$[0.24, 0.242]$	0.25	0.01	$[0.25, 0.26]$	0.26	0.01	$[0.26, 0.266]$
LARMix	0.17	10^{-3}	$[0.165, 0.171]$	0.173	10^{-3}	$[0.163, 0.184]$	0.26	0.01	$[0.26, 0.27]$
Balanced LARMix	0.215	0.01	$[0.21, 0.22]$	0.228	0.01	$[0.227, 0.228]$	0.26	0.01	$[0.26, 0.27]$
CLAPS-Mix	0.202	10^{-3}	$[0.202, 0.203]$	0.206	10^{-3}	$[0.206, 0.207]$	0.212	10^{-3}	$[0.212, 0.213]$

Table 4: Detailed analysis of message entropy.

Approaches	$\tau = 0$			$\tau = 0.6$			$\tau = 1$		
Statistics	**Average**	**Variance**	**CI**	**Average**	**Variance**	**CI**	**Average**	**Variance**	**CI**
LASTor	12.03	0.04	$[11.99, 12.07]$	12.38	0.02	$[12.35, 12.41]$	12.49	0.03	$[12.26, 12.53]$
Balanced LASTor	12.32	0.03	$[12.28, 12.36]$	12.43	0.02	$[12.4, 12.47]$	12.54	0.03	$[12.5, 12.57]$
LARMix	11.12	0.1	$[10.96, 11.27]$	11.79	0.07	$[11.74, 11.85]$	12.5	0.02	$[12.47, 12.54]$
Balanced LARMix	11.69	0.09	$[11.64, 11.75]$	12.12	0.04	$[12.08, 12.17]$	12.5	0.02	$[12.47, 12.54]$
CLAPS-Mix	11.71	0.35	$[11.61, 11.85]$	11.69	0.21	$[11.6, 11.8]$	11.37	0.18	$[11.3, 11.46]$

In section 4, we provided the results of end-to-end latency and entropy of messages, respectively, in Fig. 6a and Fig. 6b. However, we postponed the details

of these results, such as the variance and confidence interval of the average results, to this subsection. To ensure the reliability of the results, we applied a 95% confidence interval with a confidence test. The data for the average end-to-end latency and the entropy of messages are presented in Tab. 3 and Tab. 4, respectively. As observed, the variance in all cases is almost negligible, and the confidence interval is tight enough to conclude that the results in Fig. 6a and Fig. 6b are reliable.

C Client Consideration

As described in Section 2, clients choose the first hop of the mixnet uniformly at random. This approach implies that we do not optimize for the link delay between the client and the mixnet. It is important to note that while we could extend the routing to consider client-aware routing as well, several challenges arise in this scenario. Specifically, when attempting to load balance the mixnodes, the diverse preferences of clients in choosing the first hop can make it nearly impossible to balance routing effectively at this stage. Consequently, this could hinder our ability to balance the network for subsequent hops, as some mixnodes might already receive a higher proportion of traffic. Therefore, to maintain fairness and manageability in traffic distribution across the network, we choose to have the first mixnodes selected uniformly at random. This method helps prevent any single node from becoming overloaded and maintains the integrity of the network's performance and security.

D Comparison with Stratified Topology

As mentioned earlier, the motivation behind proposing the Free Route mixnet optimization is to provide a structure viable for cases requiring high anonymity but with more affordable infrastructure. However, it is important to note that the anonymity afforded by a stratified mixnode is generally less than that of the Free Routes topology.

For instance, consider a stratified topology of N mixnodes with L layers. In this case, the number of paths, assuming uniform routing, would be $(\frac{N}{L})^L$, while the same for the Free Route mixnet, as shown in Section 3, would be $\Pi_{i=0}^{I=L-1}(N-i)$. Considering these cases, the entropy of messages for the Free Route mix will be $\sum_{i=0}^{I=L-1}\log(N-i)$, whereas for the stratified routing it is $L\log(\frac{N}{L})$. Given that $L << N$, the latter generally falls behind in terms of anonymity.

However, stratified topology has a firm topology compatible with adding cover traffic, which can significantly enhance anonymity [9]. Although employing such methods for enhancing anonymity can be too costly for a local network, in scenarios where cost is a significant consideration, we recommend using the Free Route mixnet. This approach is particularly suitable for local networks where achieving high levels of anonymity must be balanced with the need for affordable implementation.

E Extension to Tor Network

As mentioned in Section 2, the Free Routes topology is extensively used in the Tor network [11], and many studies have utilized this feature to develop routing policies addressing specific interests. However, it is important to note that the nature of mixnets and the Tor network are generally different, particularly in terms of their threat models and adversaries [24, 27].

In mixnets, we consider a global passive adversary who observes all the links between the nodes within the mixnet alongside the mixnode adversary. Conversely, in the Tor network, the adversary's capabilities are typically limited to observing some parts of the network by corrupting Autonomous Systems (ASes). Therefore, adapting our proposed methodologies to be useful for the Tor network requires careful consideration of these mentioned threats. One needs to investigate more thoroughly to ensure that lowering latency in the Tor network does not inadvertently increase the adversary's advantage significantly. This adaptation involves not only technical modifications but also a deep understanding of how these changes could potentially impact the security and anonymity provided by the Tor network.