Artifact Evaluated NDSS
Available
Functional
Reproduced

LARMix: Latency-Aware Routing in Mix Networks

Mahdi Rahimi
COSIC, KU Leuven
mrahimi@esat.kuleuven.be

Piyush Kumar Sharma
COSIC, KU Leuven
pkumar@esat.kuleuven.be

Claudia Diaz
COSIC, KU Leuven
NYM Technologies SA
cdiaz@esat.kuleuven.be

***Abstract*—Anonymous communication systems such as mix networks achieve anonymity at the expense of latency that is introduced to alter the flow of packets and hinder their tracing. A high latency however has a negative impact on usability. In this work, we propose LARMix, a novel latency-aware routing scheme for mixnets that reduces propagation latency with a limited impact on anonymity. LARMix can achieve this while also load balancing the traffic in the network. We additionally show how a network can be configured to maximize anonymity while meeting an average end-to-end latency constraint. Lastly, we perform a security analysis studying various adversarial strategies and conclude that LARMix does not significantly increase adversarial advantage as long as the adversary is not able to selectively compromise mixnodes after the LARMix routing policy has been computed.**

I. Introduction

Mix networks [8], [11], [25], [30], or *mixnets*, are anonymous overlay networks that provide communication anonymity towards global adversaries, who are capable of observing all communications in the network. To achieve anonymity, messages are routed through multiple intermediary *mixnodes*, where each mixnode cryptographically transforms and reorders messages by randomly shuffling and delaying them before forwarding along the route. Mixnets incur high latency compared to direct transmissions due to both sending messages via multiple overlay hops and randomly delaying messages at each hop. Given a volume of traffic routed via the mixnet, reductions to latency in mixnets typically have a negative impact on the anonymity provided to messages, according to known fundamental tradeoffs between latency and the resulting anonymity [10].

While a variety of mixnet designs have been proposed in the literature [30], current deployed mixnets such as the Nym network [1] have an architecture that is based on Loopix [25], i.e., made up of continuous-time mixes (which delay messages independently for an exponentially distributed amount of time [19]) that are arranged in a layered topology [12]. Such mixnets can easily support private communications for delay-tolerant applications such as email, crypto wallets, file sharing, *etc.* Applications with mid-latency tolerance like instant messaging are feasible to use, but latency starts to degrade user experience. Low-latency applications such as web browsing become rather uncomfortable to use if there is high latency. Reducing latency in mixnets thus broadens the range of applications that can use them to protect communications while remaining usable. In this work we focus on a Loopix-based layered network as base design.

[1] https://nymtech.net

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2023, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23221
www.ndss-symposium.org

In particular, we are interested in finding latency efficiencies that do not have a fundamental tradeoff with anonymity [10], so that latency can be reduced while limiting the anonymity impact. To this end, we note that besides *mixing delays* that are introduced for anonymity purposes, the *propagation* time between mixnodes has a significant contribution to the overall latency.

We thus propose LARMix, a latency-aware routing scheme that reduces end-to-end latency by choosing routes with shorter propagation times – instead of choosing random routes as is currently the case. LARMix includes techniques that can be applied (independently or together) to assist mixnet creation and to define the routing policy. Mixnet creation involves the arrangement of mixnodes to layers in the network, and LARMix does so in a way that maximizes the availability of low-latency paths. Given an arrangement, LARMix computes the routing policy that defines how message paths are selected. The policy can be more or less biased towards faster routes according to a tuneable parameter τ.

We evaluate the impact of the LARMix latency-reduction techniques, applied in isolation as well as in combination, on both latency and anonymity (measured as entropy). Our results show that LARMix can significantly reduce latency in mixnets while having a limited anonymity impact. We find in experimental scenarios that the mixnet propagation latency can be reduced as much as 8x (from 120ms to 15ms) while the anonymity reduction is 2 bits in terms of entropy. Route selection that is biased towards low latency paths can result in an imbalanced routing policy where some mixnodes route more traffic than others. If load balancing is desired, LARMix includes techniques to ensure that all mixnodes route the traffic in a balanced way – while still maintaining a bias toward faster routes. We perform an empirical evaluation of the re-balancing techniques and find that the propagation latency reduction is 3.5x while less than halving the anonymity set (0.8 bits of entropy reduction). Our findings remain consistent when we scale up the number of mixnodes.

Further, we consider an average target end-to-end latency and examine the impact of having different relative contributions from *mixing* latency (introduced inside each mixnode) and *propagation* latency (introduced between mixnodes), as well as empirically finding the configuration that maximizes anonymity for various values of the target latency. We find that for a lower target latency (200ms – 400ms), the optimal configuration balances mixing time with optimizing propagation latency. However, for larger values of the target latency ($>$ 1000ms), the benefits of minimizing propagation latency become smaller and uniform random routing offers the best tradeoff, highlighting that the LARMix approach is particularly useful in situations with tight latency constraints.

We also evaluated LARMix in adversarial conditions. Specifically, we consider a *mixnode adversary* that controls a subset of mixnodes and may thus fully compromise message routes when all the mixnodes in the path are malicious. We evaluate adversarial advantage in LARMix comparing the fraction of end-to-end corrupted paths (FCP) to the baseline policy where all routes are equally likely (uniform). We review adversarial strategies to increase the FCP, such as deploying multiple mixnodes in close proximity that have small propagation latency from each other. The LARMix routing policy has increased chances of selecting paths through these nodes, as those are faster. Our results show however that our approaches do not dramatically increase the adversary's advantage as long as the adversary is not able to cause a worst case scenario (i.e., compromise the combination of nodes that maximizes FCP for a given routing policy, which is extremely unlikely to happen by chance).

Finally, we note that there are existing techniques that aim to minimize latency in other anonymous overlay networks such as Tor. Most such designs [1], [17] cannot be used for mixnets due to inherent limitations (detailed in Sec. III). However, CLAPS [26] approach of using linear programs could be applied to mixnets. Thus, in Sec. VII we describe how CLAPS can be applied to the problem at hand and perform a comparison between the results of CLAPS and LARMix. We find that the CLAPS based approach does not provide better latency-anonymity tradeoffs than LARMix. Additionally, the approach does not scale well and incurs large computational overhead (300x more than LARMix). Such overhead can make it infeasible to calculate solutions in a timely manner.[2]

To summarize, LARMix proposes a set of novel and efficient techniques to define routing policies that reduce propagation latency in mixnets, while having a limited impact on anonymity and on the ability of an adversary to fully compromise message routes. These techniques can thus help broaden the range of applications supported by mixnets.

II. Problem statement

A. Anonymity-latency tradeoffs in mixnets

As shown by the *anonymity trilemma* [10], in mix networks (*mixnets*) anonymity is traded with traffic volume and end-to-end latency. According to this *trilemma*, for a given traffic rate, reducing communication latency comes necessarily at the expense of anonymity. In practice, this limits the usability of mixnets for applications that have a sub-second latency tolerance, such as instant messaging or web browsing.

We observe however that the end-to-end latency of a message routed through a mixnet has multiple components. Given a message routed via L intermediary mixes in a Loopix-like network, the average end-to-end latency can be expressed as:

$$\bar{l}_{e2e} = L \cdot \mu + (L+1) \cdot \delta + (L+1) \cdot \bar{l}$$

where μ is the average *mixing latency* introduced at each mix for anonymity purposes (reordering of messages); δ is the time needed (by both intermediary mixes and the recipient) to cryptographically process a message; and $\bar{l}$ is the average network *propagation latency* per transmission, i.e., the amount of time the message spent traveling on the Internet per hop of the route. Each of these latency components has a different relationship to anonymity:

The mixing delay μ directly impacts anonymity as modeled by the *anonymity trilemma*, with larger μ benefiting anonymity by allowing for more reordering of messages per mix. This is a design parameter that may take any arbitrary value, according to the desired anonymity-latency tradeoff (currently $\mu = 50$ ms per mix in the Nym network).

The processing time δ has no anonymity benefit and should thus be minimized as much as possible, e.g., with more efficient cryptographic algorithms. Existing cryptographic message formats such as Sphinx [9] require $\delta < 1$ ms of processing per hop, and thus δ already contributes minimally to $\bar{l}_{e2e}$.

The propagation latency $\bar{l}$ does not have an obvious tradeoff with anonymity and may thus be reduced in order to lower $\bar{l}_{e2e}$, making mixnets more usable for applications with sub-second latency tolerance. Reductions of $\bar{l}$ may however still indirectly involve an anonymity tradeoff: a preference for faster routes biases route selection, which becomes more predictable and thus less anonymous. Propagation latency can range from 10 ms to 150 ms per hop, depending on geographical distance and network connectivity in each hop of the message route – thus potentially contributing more than half the end-to-end latency, particularly when $\mu \leq 100$ ms.

The parameter $\bar{l}$ can be further broken down into components: $\bar{l} = \frac{1}{(L+1)}(l_{s,mix} + \bar{l}_{mix} + l_{mix,r})$, where $l_{s,mix}$ is the propagation latency between the message sender s and the mixnet, $l_{mix,r}$ is the propagation latency from the mixnet to the final recipient r, and $\bar{l}_{mix}$ is the aggregation of $L-1$ link propagation delays *within* the mixnet. Of these three propagation latency components, $l_{s,mix}$ and $l_{mix,r}$ depend on the network locations of individual senders and receivers; while $\bar{l}_{mix}$ is a characteristic of the mixnet that is the same (on average) for all clients. The latency $\bar{l}_{mix}$ depends on the mixnet routing scheme and whether it takes propagation latency into account when selecting message routes through the mixnet. LARMix thus aims at reducing $\bar{l}_{mix}$ with the definition of a routing policy biased towards faster links.

B. Design goals and system model

Our main goal in this work is to develop a latency-aware routing policy for a mixnet that provides a good tradeoff between *(i)* the average aggregate mixnet propagation latency

[2]Both Tor and Nym require refreshing the routing weights every hour, which is longer than the time taken by CLAPS to compute a solution for a large network.

$\bar{l}_{mix}$; and *(ii)* the anonymity impact of biased selection of faster routes. We define a parameter τ $(0 \leq \tau \leq 1)$ that tunes this tradeoff, with $\tau = 0$ meaning that $\bar{l}_{mix}$ is maximally optimized (reduced) and $\tau = 1$ that no routing optimization is made to limit $\bar{l}_{mix}$, which corresponds to a uniform routing policy.

We furthermore consider two constraints, namely: (1) *load balancing*, meaning that the scheme selects routes so that all mixes process on average the same amount of traffic; and (2) a *layered* network topology LxW with L layers and W mixes per layer, such that all messages are part of the same anonymity set when the network scales up (larger W) to serve more clients. This layered design is common in various mixnet proposals, such as Loopix [25] or Nym [11], and is in contrast to mix cascade topologies where the anonymity sets of W parallel cascades [7] are disjoint and thus do not benefit from scaling the user base. Also following the baseline of prior proposals [11], [25], we consider that the traffic load must be distributed equally among all W mixnodes of each layer. Our methods are however generalizable to setups where mixnodes have different capacities that must be accounted for when balancing the traffic load (ref. Appendix. A).

Note that LARMix does *not* consider the client or destination locations when selecting message paths, instead focusing on reducing propagation latency in the links between mixnodes. Dependencies on client or destination location make it impossible to bound the share of traffic routed by each mixnode, as well as causing routing choices themselves to leak information about the location of the end clients sending or receiving the anonymous messages. We consider a single routing policy that is used by the entire set of clients, meaning that routing choices are the same for all clients, and thus leak no information about message the sender or receiver.

C. Threat model

There are two main types of adversaries to consider when evaluating anonymity in overlay networks such as mixnets. The *global passive adversary* is assumed to have visibility over all the network links. Concretely, such an adversary can observe all the messages transmitted between network entities and derive probabilistic relationships between input and output messages. A measure of a message's anonymity is then given by the entropy of the probability distribution linking that message at one end of the communication to all the possible corresponding messages at the other end [13], [27]. The *mixnode adversary* additionally controls a subset of malicious nodes that are part of the mixnet. For those compromised nodes, the adversary knows the mapping between incoming and outgoing messages. If all the mixnodes in a message's route are corrupted, then the adversary is able to trace the message end to end, meaning that the message has zero anonymity. For the purpose of our evaluation, we consider both adversaries and assess anonymity under various adversarial conditions.

III. Related Work

To our knowledge, there has been no previous attempt to develop latency-aware routing for layered mixnets. There are however multiple proposals [1]–[3], [5], [15], [17], [23], [26], [29], [34] to add similar capabilities to Tor [14].[3] Most of these solutions are specifically tailored to Tor and are not easily adaptable to the context of mixnets. Some of the designs could potentially be used for mixnets, but present limitations. For instance, Lastor [1] introduces a methodology to select relays to minimize latency but does not consider balancing the load in the network. The recent ShorTor [17] paper takes a different approach to minimize latency by suggesting an overlay network on top of Tor. The idea rests on the observation that BGP is designed to respect business relationships among ASes and is not optimized for the shortest or lowest latency paths. Thus, the direct path between two endpoints on the Internet may not be the fastest and if the path passes via some other intermediate endpoint it can be faster. This approach is used by CDNs to optimize for low latency and thus they suggest creating a multi-hop overlay network of intermediate nodes to reduce latency in Tor without needing to change their existing routing policy. An approach like ShorTor could thus be applied in conjunction with our proposal, though once the routing policy is biased towards choosing low-latency links, chances are that not much room is left for the kind of optimization performed by ShorTor, which adds intermediaries from the set of nodes. Furthermore, ShorTor may also lead to unbalanced load as nodes have to now process two kinds of traffic: the traffic routed through them as part of the routing policy, and the additional traffic received as part of the new multi-hop overlay network.

The prior work that is most relevant to LARMix is CLAPS [26]. It provides a framework to improve any existing location-aware scheme proposed for Tor. CLAPS uses linear programming, where an objective function is defined together with a set of constraints. Once defined, the linear programs are solved for maximizing or minimizing the objective function while respecting the said constraints. These constraints can be defined on properties such as node capacities, adversarial advantage in specific attacks, *etc.* A CLAPS-based approach can thus be adapted to mixnets, defining latency minimization as the objective function and accounting for anonymity and load balance via constraints. In Sec. VII, we develop a CLAPS-based approach to obtain a lower-latency routing policy for mixnets, which we compare to LARMix. We observe that the CLAPS-based approach does not provide better tradeoffs than LARMix while requiring ≥ 300x computational overhead.

IV. Approach

Considering that $N = LW$ mixnodes are available, we identify two stages in the setup of a layered mixnet, shown in Fig. 1: (1) assigning nodes to L mixnet layers; and (2) defining the routing policy that determines how paths are selected. Since messages must traverse one mixnode per layer, the routing policy is defined by the fraction of messages that each node in a layer forwards to each node of the following layer. Our approach includes strategies that can be applied at both stages to facilitate the reduction of the average latency $\bar{l}_{mix}$. We note however that the approach is modular and the strategies

[3]Additionally, there are approaches [4], [22], [32], [33] that are location-aware but focus on protecting Tor from traffic analysis and website fingerprinting attacks. The goal of these solutions is to make it harder for malicious ASes or ISPs to have visibility over of both the entry and exit relays of a Tor circuit. These considerations are not applicable to mixnets as they already assume a global adversary who observes all network links.

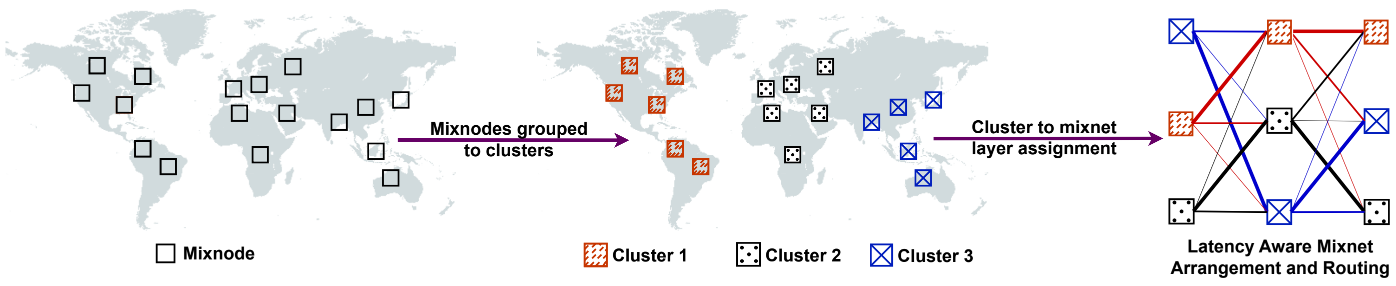

Mixnodes grouped to clusters
Cluster to mixnet layer assignment
Mixnode
Cluster 1
Cluster 2
Cluster 3
Latency Aware Mixnet Arrangement and Routing

Fig. 1: Overview of LARMix.

can be applied together as well as separately. We additionally introduce methods to re-balance the traffic load across the mixnodes of each layer after it becomes imbalanced due to latency optimizations.

A. *Selection and arrangement of mixnodes*

In this phase we consider there are $N = LW$ mixnodes to arrange in L layers of W nodes. As baseline, existing approaches [11], [25] assign nodes to layers at random. Such random arrangements may however lead to configurations where a high proportion of routes have high propagation latency. As extreme example, consider an arrangement where nodes belong to layers populated by other nodes in their vicinity, while traversing from one layer to another always involves an inter-continent leap. In this case $\bar{l}_{mix}$ cannot be reduced much by the routing algorithm and will always be high. Conversely, arranging mixnodes in layers with maximum diversity, i.e., so that each layer includes mixnodes from all locations, ensures there will be some choices of low latency paths, a feature that can then be leveraged by a routing policy that seeks to minimize $\bar{l}_{mix}$.

The method we propose to achieve this first clusters mixnodes according to their geographical location, and then uses the obtained clusters to arrange mixnodes in layers. We assume that average values for the propagation latency between pairs of mixnodes are publicly available, e.g., through the VerLoc protocol [20], which also allows verifying mixnodes' approximate geolocation. We represent geolocation by its Cartesian coordinates and feed them as input to the clustering method. We considered K-means, K-medoids and FCM [18] as clustering methods because they can take the coordinates and iteratively identify the best K clusters. (we discuss in Appendix B how to select an appropriate K).

After this step we obtain K clusters along with their centroids. The purpose of the layer arrangement algorithm is to ensure that each of the L mixnet layers of size W has geographically diverse (i.e., cluster diverse) mixnodes. The algorithm, outlined in Appendix C, depends on K and W, and on whether $W < K$, $W > K$ or $W = K$.

When $W < K$, we start by randomly selecting a cluster and mixnode from that cluster to be the first mixnode in the first layer. Next, we measure the distance between the centroids of the chosen cluster and the other clusters and select the cluster with the largest distance. We randomly select a mixnode from it to be the second mixnode in the layer. We choose as third cluster the one with maximum distance from the two previous clusters, and select a mixnode from it. We continue this process till we have selected W nodes. And then repeat the process for the other layers until all mixnodes are assigned (note that mixnode selections are without replacement). When $W = K$, we simply choose one mixnode from each cluster to assemble a layer. The final scenario occurs when there are fewer clusters than mixnodes per layer *i.e.* $W > K$. In this case, we first select one mixnode from each cluster for the layer. The remaining nodes are sampled (without replacement) from clusters proportionally to cluster size.

B. *Latency Aware Routing (LARMix)*

Existing mixnet routing schemes select message routes through the layers uniformly at random [11], [25], without considering the propagation latency of different mixnet links. LARMix is a latency aware routing scheme that reduces the average propagation latency of messages routed through the mixnet. This is achieved by biasing route selection towards lower latency mixnet paths. The parameter τ introduced in Section II-B tunes the level of bias in LARMix, with $\tau = 0$ meaning that lowest latency routes are deterministically selected, and $\tau = 1$ meaning that routes are selected uniformly at random (same as in preexisting schemes).

A routing scheme that minimizes propagation latency considering all hops of the route end-to-end would need to account for sender and receiver locations, which may not be known when selecting a route. Furthermore, such routing scheme would result in distinct routing choices per client, which in turn has two effects: (1) making load balancing impossible; and (2) having message routes that are correlated with clients' locations and choice of correspondent, which opens avenues for inference attacks. To avoid this, LARMix assumes that all users choose message routes following the same policy, independently of their own location and that of their correspondent. Once an entry (first layer) mixnode M^1 has been selected for a message route, however, the choice of mixnodes in the following mixnet layers $M^2 \cdots M^L$ may be biased (depending on τ) towards faster links.

Let $M^1 = m_i^1$ be the mixnode that was randomly chosen as the first node in a message route, and let l_{ij} be the propagation latency between m_i^1 and mixnode m_j^2 in the next layer. We order nodes m_j^2 by increasing l_{ij} and define a function $R_i(j)$ that outputs the position (rank) of node m_j^2 in this ordered list, with $R_i(j) = 0$ for the node with the lowest propagation latency and $R_i(j) = W - 1$ for the node with the largest propagation latency. LARMix selects the next mixnode M^2

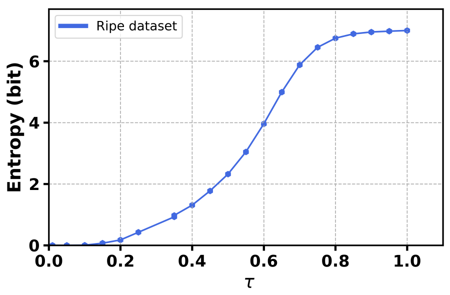


Fig. 2: Average entropy of the distributions $\mathbb{P}[M^2 = m_j^2 | M^1 = m_i^1]$ with respect to τ, considering a mixnet with $L = 2$, $W = 128$, and propagation latencies l_{ij} sampled from RIPE atlas.

with probability $\mathbb{P}[M^2 = m_j^2 | M^1 = m_i^1]$ computed as:

$$\mathbb{P}[M^2 = m_j^2 | M^1 = m_i^1] = \frac{\left(\frac{1}{e}\right)^{R_i(j)\frac{(1-\tau)}{\tau}} \left(\frac{1}{l_{ij}}\right)^{(1-\tau)}}{\Sigma_k \left(\frac{1}{e}\right)^{R_i(k)\frac{(1-\tau)}{\tau}} \left(\frac{1}{l_{ik}}\right)^{(1-\tau)}} \quad (1)$$

In the case of $\tau = 0$, note that $\mathbb{P}[M^2 = m_j^2 | M^1 = m_i^1] = 1$ if $R_i(j) = 0$ and $\mathbb{P}[M^2 = m_j^2 | M^1 = m_i^1] = 0$ if $R_i(j) > 0$. For $\tau = 1$ we obtain $\mathbb{P}[M^2 = m_j^2 | M^1 = m_i^1] = \frac{1}{W}$ for all m_j^2. For intermediate values $0 < \tau < 1$, routes are more likely to be chosen when they have lower latency than the alternatives. Appendix E further discusses the intuition behind Formula 1.

The amount of randomness (or conversely, determinism) in the LARMix routing choices can be characterized by the Shannon entropy [28] of the distribution $\mathbb{P}[M^{\ell+1} = m_j^{\ell+1} | M^\ell = m_i^\ell]$ per mixnode m_i^ℓ in layer $1 \leq \ell \leq L - 1$ with respect to its W successors $m_j^{\ell+1}$ in layer $\ell + 1$. We show in Fig. 2 how τ influences the entropy (randomness) of this distribution considering an example with $L = 2$ layers and $W = 128$ mixnodes per layer. We sample from the publicly available RIPE atlas dataset [31] inter-node latency values l_{ij} between the 128 mixnodes in the first layer and the 128 mixnodes in the second layer. We then compute the routing weights $\mathbb{P}[M^2 = m_j^2 | M^1 = m_i^1]$ with Eq. (1) for different values of τ. We compute the entropy of the weights distribution for each of the 128 nodes m_i^1 and obtain the average. The average entropy values range from 0 bits when $\tau = 0$ (deterministic routing) and $\log_2(128) = 7$ bits when $\tau = 1$ (random routing).

C. Re-balancing the network load

For $\tau < 1$, the distributions $\mathbb{P}[M^{\ell+1} = m_j^{\ell+1} | M^\ell = m_i^\ell]$ obtained following the approach described in the previous section will normally result in an uneven load across mixnodes, since there is no guarantee that $\sum_i \mathbb{P}[M^{\ell+1} = m_j^{\ell+1} | M^\ell = m_i^\ell]$ will add up to $\frac{1}{W}$ of the total traffic routed in layer $\ell + 1$. Thus, beyond the first layer (which is still chosen uniformly at random), some mixnodes receive more traffic than others.

Load imbalance may be undesirable as, e.g., it can result in performance bottlenecks due to some mixnodes receiving a disproportionate share of traffic and becoming overloaded. In the rest of this section we propose two balancing approaches, which we call *greedy balancing* and *naive balancing*.

For compactness, we denote the conditional probability $\mathbb{P}[M^{\ell+1} = m_j^{\ell+1} | M^\ell = m_i^\ell]$ as γ_{ij}^ℓ. For each layer ℓ, we organize the values γ_{ij}^ℓ in a *scattering matrix* Γ^ℓ of size WxW:

$$\Gamma^\ell = \begin{bmatrix} \gamma_{11}^\ell & \cdots & \gamma_{1W}^\ell \\ \vdots & \ddots & \vdots \\ \gamma_{W1}^\ell & \cdots & \gamma_{WW}^\ell \end{bmatrix}_{W \times W} \quad (2)$$

The goal of balancing is to obtain a modified scattering matrix $\tilde{\Gamma}^\ell$ that still prioritizes low latency routes but that also fulfills the balancing condition $\sum_i \tilde{\gamma}_{ij}^\ell = 1$. Note that as in the previous case, it still must hold that $0 \leq \tilde{\gamma}_{ij}^\ell \leq 1$ and $\sum_j \tilde{\gamma}_{ij}^\ell = 1$.

1) Greedy balancing: This approach identifies nodes that are overloaded, i.e., those for whom $\sum_i \gamma_{ij}^\ell > 1$, and decreases their weights $\tilde{\gamma}_{ij}^\ell$, while increasing $\tilde{\gamma}_{ij}^\ell$ for underloaded mixnodes for whom $\sum_i \gamma_{ij}^\ell < 1$. The algorithm is *greedy* because it attempts to keep the $\tilde{\gamma}_{ij}^\ell$ distributions as biased as possible towards faster routes, while still balancing the load. The algorithm runs multiple iterations, redistributing the load from overloaded nodes to underloaded ones, as described in Algorithm 1 in Appendix C, until the routing for the layer is balanced. The balancing is done sequentially per layer, starting from the first layer.

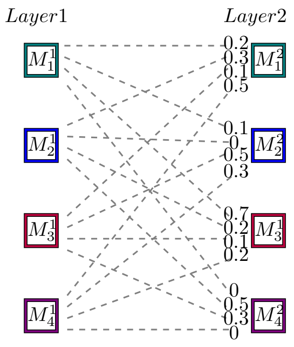


Fig. 3: Example of initial unbalanced routing weights.

As an example to illustrate how the greedy algorithm functions in practice, consider a first layer of width $W = 4$ as depicted in Fig. 3, with a scattering matrix Γ^1 before balancing given by:

$$\Gamma^1 = \begin{bmatrix} 0.2 & 0.1 & 0.7 & 0 \\ 0.3 & 0 & 0.2 & 0.5 \\ 0.1 & 0.5 & 0.1 & 0.3 \\ 0.5 & 0.3 & 0.2 & 0 \end{bmatrix}$$

To balance the routing, we first compute the aggregate load per mixnode in the second layer, which is equal to $[1.1, 0.9, 1.2, 0.8]$. We thus label the first and third mixnodes as overloaded, and the other two as underloaded. We then multiply the first column of the scattering matrix by $\frac{1}{1.1}$ and the third column by $\frac{1}{1.2}$, this reduces the overall traffic received by the two overloaded nodes by scaling down the updated values for $\tilde{\gamma}_{i1}^1$ and $\tilde{\gamma}_{i3}^1$. The difference between the initial and updated values, given by $\gamma_{ij}^1 - \tilde{\gamma}_{ij}^1$, contains the 'leftover' probabilities, and is equal to $[0.0182, 0.0273, 0.009, 0.0455]$ and $[0.117, 0.033, 0.0167, 0.033]$ in the example.

Next, we distribute these leftover weights among the underloaded nodes proportionally to their initial weight γ^1_{ij}, leading to an updated scattering matrix $\tilde{\Gamma}^1$

$$\tilde{\Gamma}^1 = \begin{bmatrix} 0.1818 & 0.2352 & 0.583 & 0 \\ 0.2727 & 0 & 0.167 & 0.5606 \\ 0.0909 & 0.516 & 0.0833 & 0.309 \\ 0.4545 & 0.3785 & 0.167 & 0 \end{bmatrix} \tag{3}$$

This resulting scattering matrix is however not yet balanced, as the second mixnode is now overloaded while the fourth is still underloaded. We thus iterate the algorithm to obtain an updated scattering matrix $\tilde{\Gamma}^1$:

$$\tilde{\Gamma}^1 = \begin{bmatrix} 0.1818 & 0.208 & 0.583 & 0.07 \\ 0.2727 & 0 & 0.167 & 0.5606 \\ 0.0909 & 0.456 & 0.0833 & 0.368 \\ 0.4545 & 0.335 & 0.167 & 0.043 \end{bmatrix} \tag{4}$$

As the $\tilde{\Gamma}^1$ resulting from this second iteration is balanced, the algorithm terminates for the first layer and can be applied to the next layer.

2) Naive balancing: The greedy algorithm achieves a balanced network while maintaining the bias towards low latency path selection. To achieve that however, it may require multiple iterations, increasing the computational overhead. The *naive balancing* algorithm is an alternative that achieves balancing in just one iteration.

Similar to the greedy approach, we first calculate the load per node by summing columns in the scattering matrix Γ^1 and scale down the weights to balance those overloaded nodes. The leftover load is also calculated as in the greedy case, but in this case the leftover is distributed to underloaded nodes according to how underloaded they are. Thus, unlike greedy approach that prioritizes the latency bias (or proximity) while assigning load, the naive approach only looks at the available capacity of underloaded nodes.

Considering again the example in Fig.2, the two underloaded mixnodes 2 and 4 have loads of 0.9 and 0.8, respectively, and thus spare capacities of 0.1 and 0.2 to achieve balance. The naive algorithm distributes the overall leftover given by $0.1 + 0.2 = 0.3$ by distributing $\frac{0.1}{0.3}$ of the leftover weight to the second node and $\frac{0.2}{0.3}$ to the fourth node. This leads to an updated scattering matrix $\tilde{\Gamma}^1$:

$$\tilde{\Gamma}^1 = \begin{bmatrix} 0.1818 & 0.045 & 0.583 & 0.19 \\ 0.2727 & 0.5201 & 0.167 & 0.04 \\ 0.0909 & 0.308 & 0.0833 & 0.517 \\ 0.4545 & 0.026 & 0.167 & 0.352 \end{bmatrix}$$

We evaluate these two balancing approaches in terms of their latency and anonymity in the next section.

Rebalancing operation: Balancing the network load is a one-time operation per network configuration. It is then repeated when the network is reconstituted due to churn or periodic updates. In Nym, this rearrangement (selecting nodes and assigning them to layers) happens once every hour. The rebalancing operation is global and requires knowledge about latency between all pairs of nodes. In Nym, information on pairwise latency is publicly available via the implemented Verloc protocol [21]. The node assignment to layers and routing policy (weights) are agreed by a set of entities orchestrating the network. In Nym these entities are a set of validators running a consensus protocol.[4] Note that, given the latency dataset, each validator can locally run our algorithms to determine the routing weights. If all entities with the same input and algorithm arrive at the same routing policy, they sign it in a consensus. This policy is then distributed to clients to choose packet routes.

V. Evaluation

We evaluate LARMix using two approaches. The first is an analytical approach that evaluates the predictability of message routes for a given routing policy. If message routes are fully deterministic (as is the case in mix cascades) then an adversary who wants to trace a message sent by a client to a first mixnet node can narrow down the anonymity set to the messages output by the last node in that deterministic route. On the other extreme, if message routes are completely uniform, all nodes in the last layer are equally likely to output a target input message sent to any of the first-layer nodes, maximizing the uncertainty of the adversary. Solutions in between result in some nodes being more likely than others as last hop of a message route, given the entry node of that route.

The second approach simulates a mixnet that routes messages with a discrete event simulator. Unlike the first approach that isolates and studies the impact only due to predictability of routes, this approach helps us to study the anonymity and latency of individual messages due to the combined effect of predictability of routes as well as the mixing of packets within mixnodes. The simulator and analysis scripts are written in Python with about 10K LOC. The details of the implementation are provided in Appendix G. (Link to code: https://github.com/larmix/larmix)

The rest of this section first describes the metrics used for the evaluation and the assumptions made in the system model. We then present the experimental setup and the results obtained by performing various experiments. An overview of the experiments conducted and the system parameters are listed in Tab. I and Tab. II, respectively.

A. Evaluation metrics

We evaluate LARMix with respect to both average latency and anonymity, which are calculated as follows.

1) Latency: In the analytical approach we consider all possible paths through the mixnet and their probability of being chosen according to the routing policy. The latency l_{mix} of a path is given by the sum of the propagation delays experienced at each link of the route in the mixnet. We compute the average mixnet latency $\bar{l}_{mix}$ as the weighted average of all paths, considering their aggregate propagation delay and their probability of being chosen.

[4]Tor also periodically distributes to clients a consensus document (signed by 10 authorities) specifying the routing weights that clients follow to construct circuits.

Experiment	Variables	Results
Latency aware routing	Arrangement + Routing + Balancing	Entropy + Latency
Meeting end-to-end delay constraints	Network & Mix latency + Routing	Value of τ with max entropy
Varying network size	Network size + Routing + Balancing	Entropy + Latency

TABLE I: Overview of experiments: Arrangement = Random, diversified, and worst case. Routing = τ ranging from 0 (deterministic) to 1 (random). Balancing = Imbalanced, naive balancing and greedy balancing.

In the simulation-based evaluation, we record the latency of each message that traverses the mixnet. Note that this recorded latency includes the aggregation of l_{mix} propagation delay and the mixing delay experienced at each intermediary mixnode, which is exponentially distributed with mean μ. On average, a message going through three mixes is delayed by 3μ for mixing purposes. We collect latency samples for all the messages in the simulation and show their distribution in the form of a box plot.

2) Anonymity: Traditionally, for mixnet systems, anonymity is conceptualized as the uncertainty of an adversary attempting to identify the output corresponding to a target input message; or vice versa, identify the sender of the input corresponding to a target output message. Intuitively, if an input message can be potentially mapped to a large set of outputs with equal likelihood, then the adversarial uncertainty is very high, while more deterministic mappings lead to lower adversarial uncertainty. The uncertainty is modeled as the Shannon entropy [28] of the probability distributions relating inputs to outputs, which are obtained given knowledge of the mixnet's parameters, the routing policy and the observation of all messages transmitted between network entities with their exact arrival and departure timings [13], [27]. A resulting entropy of b bits represents the effective size of the target's anonymity set, and corresponds to an adversarial uncertainty equivalent to perfect indistinguishability among 2^b messages as possible matches for the target.

In the simulator we select target messages and calculate the probability distributions of each target input being any of the possible outputs, based on approaches outlined in [6], [24]. Each target thus provides a latency sample and an entropy sample, denoted as $H(m)$. We then show boxplots that include all the obtained samples. Note that in this approach the entropy of output messages is due to a combination of the mixing of messages within a mixnode (with the help of the mixing delay) and the randomness of routing choices.

Since the proposed routing approach aims to minimize propagation latency without modifying the mixing latency μ, in the analytical approach we isolate and study the effect of the routing policy on anonymity. To do this, we represent the routing weights of the mixnet as matrices and we calculate the routing entropy by performing matrix operations. To understand this method in detail, we need to define some terms. The first term referring to the scattering matrix is already defined in Sec. IV-C. We now also define the transformation matrix.

Definition 1. ***The transformation matrix* $\mathbf{T}$ *is the result of the multiplication of the scattering matrices for the different mixing layers. The matrix element* β_{ij} *defines the probability that a message entering the mixnet via mixnode* m_i^1 *in the first layer exits the mixnet via mixnode* m_j^L *in the last layer.***

$$\mathbf{T} = \Gamma^1 \times \Gamma^2 \cdots \times \Gamma^{L-1}. \tag{5}$$

$$\mathbf{T} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1W} \\ \vdots & \ddots & \vdots \\ \beta_{W1} & \cdots & \beta_{WW} \end{bmatrix}_{W \times W} \tag{6}$$

Given the transformation matrix, we calculate the entropy for matching the first and last mixing layers as follows:

$$\begin{aligned} \mathsf{H}(M^1 \to M^3) &= \sum_{i=1}^{W} \mathsf{H}(M^1 \to M^3 | M^1 = m_i^1)\mathbb{P}(M^1 = m_i^1) \\ &= \frac{1}{W}\sum_{i=1}^{W} \mathsf{H}(M^1 \to M^3 | M^1 = m_i^1), \\ &= -\frac{1}{W}\sum_{i=1}^{W}\sum_{j=1}^{W} \left(\beta_{ij}\log(\beta_{ij})\right). \end{aligned} \tag{7}$$

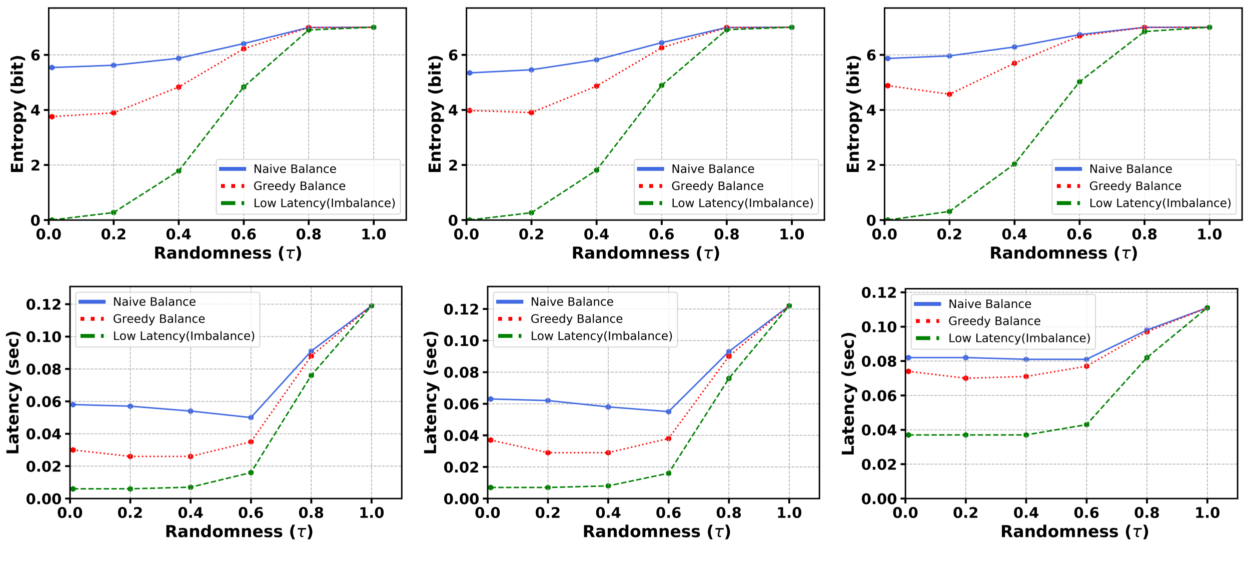
Note that this entropy measures the uncertainty on the possible output mixnodes for a route given an input mixnode, and thus can, at maximum, be equal to $\log_2 W$. For individual messages the uncertainty on the output mixnode is compounded with the number of messages output by each of the last layer mixnodes during the time period when the target message may be output. Thus in a sense, simulation-based message entropy is an aggregation of the entropy of the transformation matrix and the entropy due to the mixing of multiple messages within individual mixnodes (see Appendix F for details).

Parameter	Value
Topology	Stratified
Mix layers (L)	3
Size of network (N)	384
Layer size (W)	128
Mix latency (μ)	50 ms
Input traffic rate	10000 msgs per sec
Target messages	200
Iterations	400
Number of clusters (K)	5
Clustering method	K-medoids

TABLE II: Baseline parameters for experiments.

B. Experimental setup

We show in Table II the parameter values we considered as baseline for the experiments. Unless specified otherwise, these values are used by default in all experiments. We consider a Loopix-based mixnet architecture that uses a layered topology



(a) Clustering and Diversification (b) Random (c) Bad Case

Fig. 4: Entropy and Latency for the analytical evaluation with different arrangements.

with three layers. Each layer consists of 128 nodes[5] and the total network size is 384 nodes. We assume μ to be 50 ms per mixnode *i.e.,* every message will be delayed on average 50 ms at each layer. Note that the mix latency is only relevant to simulation-based experiments as in the analytical setting we just evaluate the randomness of the routing policy. In the simulations we assume a traffic generation rate for all clients combined of 10000 messages per second, with each mixnode receiving on average $\frac{10000}{W}$ messages per second. We randomly select input messages that we call 'target messages'. For each target we calculate the entropy of the distribution describing the likelihood of corresponding to each possible output. We consider 200 target messages per simulation run and perform 400 runs, each with a new mixnet arrangement and freshly computed routing weights. To assign realistic latency values to the links between mixnodes we consider the latency measurement data from the RIPE atlas project [31], an overlay network of more than 10000 probes spread across the globe. From the publicly available RIPE data we extract the latency values of a subset of nodes, all of which have a latency measurement to every other node in the subset. We obtained a dataset that consists of 568 such nodes that we use for our experiments.

C. Experimental results

We now evaluate LARMix and present results for the different experiments summarized in Table I.

1) Evaluating latency-aware routing: This first experiment evaluates the approaches developed for mixnode arrangement and route selection, with and without load balancing. As previously mentioned, we evaluate the schemes in two settings. We first describe the results of the analytical approach followed by the simulation results. Note that all the parameters for the experiments are as described in Tab. II.

[5]We select W to be an integer power of 2 so that it is convenient to analyze $H(T)$, which is a factor of $\log_2 W$

Analytical evaluation: Fig. 4 shows the entropy $H(T)$ (top figures) and average path latency $\bar{l}_{mix}$ (bottom figures) relative to the the randomness parameter τ in three different cases of mixnode arrangement into layers: Fig. 4a considers cases where the diversification algorithm developed in Sec. IV-A was used to place mixnodes in layers; Fig. 4b considers instead cases where mixnodes are arranged randomly in layers; and Fig. 4c considers unfavorable corner cases, where mixnodes in the same continent (or cluster) are placed in the same layer and thus all available routing links are high-latency.

For each of these three scenarios we depict results for three flavors of the proposed schemes, considering values for the randomness parameter τ ranging between zero (deterministic) and one (uniform). In blue we show results when the 'naive balancing' approach is considered for deriving the routing policy; in red we show results when the 'greedy balancing' approach is considered instead; and in green we show results when the routing policy minimizes latency without concern for equalizing (balancing) the load of traffic across nodes.

We observe that the results of scenarios using the diversification algorithm (Fig. 4a) are essentially identical to scenarios where mixnodes are randomly placed (Fig. 4b), both in terms of latency and anonymity. This is because a random placement when considering $W = 128$ mixnodes per layer is highly likely to offer enough low latency links to allow for a good latency optimization of the routing policy. Employing the diversification algorithm is however still useful to avoid particularly unfavorable corner cases (Fig. 4c) which, even if unlikely, may occasionally happen. In these worst cases, for all three schemes the entropy is only slightly better than in the diversified and random arrangements while (as expected) the latency is significantly worse.

We observe in all figures that both entropy and latency increase with the randomness τ. For $\tau = 1$ all approaches lead to uniform routing policies with maximum entropy (given by

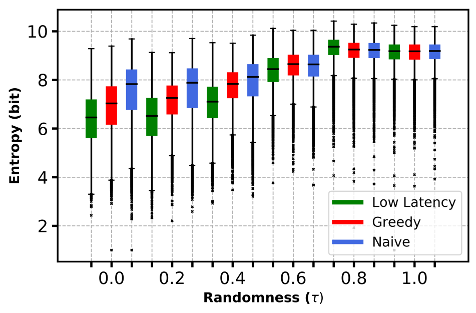
$\log_2 W = 7$ bit for $W = 128$) and no latency optimization. In the other extreme, for $\tau = 0$ the imbalanced approach (green line) shows aggressive latency optimizations of about 90% latency reduction compared to uniform routing – though at the cost of zero entropy in the transformation matrix, meaning that given the first mixnode in the message's route, the last layer mixnode where the message will come out is fully predictable. Balanced approaches do not go as far in latency optimization but are able to offer significantly more entropy in terms of routing. At $\tau = 0$ the naive approach (blue line) cuts latency by 50%, optimizing the least for latency compared to the other two alternatives but also preserving the most anonymity: with 5.76 bits this scheme incurs a loss of less than 1.5 bits compared to uniform routing. The greedy approach (red line) provides an intermediate tradeoff, where the latency reduction is 75% for $\tau = 0$ while the anonymity is around 4 bits, a loss of 3 bits compared to uniform routing.

When observing results for intermediate values of τ, we can see that for all three approaches in all the cases the average latency is rather stable between $\tau = 0$ and $\tau = 0.6$, and then increases sharply between $\tau = 0.6$ and $\tau = 1$. The entropy on the other hand increases steadily between $\tau = 0$ and $\tau = 0.8$, at which point it maxes out and does not increase any further when τ is increased. This means that there are sweet spots with respect to the latency-entropy tradeoff determined by τ. In particular, for $\tau = 0.6$ we obtain all the latency reduction benefits in all the schemes while losing less than one bit (out of seven) of entropy in the balanced approaches, and two bits (instead of all seven as for $\tau = 0$) in the imbalanced approach.

Note that although the sweet spot may vary based on the latency dataset, it should still exist. Any given set of globally distributed nodes will include large and small pairwise latency values depending on geographical distance between the pair. When routing is uniform ($\tau = 1$) the average latency is influenced by the large latency values (worst cases). The largest values are the first to be removed when τ decreases. As τ decreases further, however, there are diminishing returns, as the latency values being removed are not significantly larger than the ones remaining. In contrast, for entropy, removing a few paths from a large number of possible paths in the mixnet has a small impact on overall entropy, since all messages are still being mixed with each other and there remains significant uncertainty on possible message routes (many routes are still possible). Entropy starts decreasing significantly when too many possible paths are removed, such that routing becomes rather deterministic (τ approaching zero).

Simulation evaluation: We now present simulation results for entropy and latency in the diversified arrangement, shown in Fig. 5. The simulation allows us to obtain samples for individual messages, which provide not just the expected average for latency and entropy but their full distribution of values, depicted in a boxplot per simulation setup. The boxplots in the figure show the distribution of latency and entropy samples from experiments with a certain τ parameter value and balancing approach (naive balancing, greedy balancing, or unbalanced). For each scenario, the values in the box contain 50% of the samples while the whiskers show the data range and the dots mark the outliers.

Note that the latency values include the aggregation of both propagation and mixing delays, while the entropy values account for uncertainty introduced by both the mixing at each mixnode as well as the routing choices. Therefore, these results are dependent not only on the routing policy but also on the average delay per mixnode (set to $\mu = 50$ ms) and the volume of traffic (set to 10000 messages per second sent to the network by the entire set of clients).



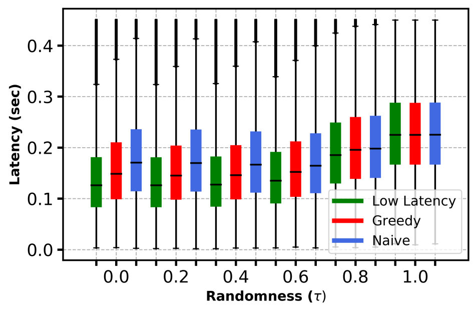


Fig. 5: Entropy and Latency obtained from simulations for diversified arrangement.

In terms of latency there is wide variability among samples: in the best cases messages are transmitted with negligible latency, while in the worst cases they may experience up to half a second of latency. This is in large part given by the variability of the exponentially distributed mixing delay applied at each mixnode, which is long-tailed. Note that in the lowest latency scenarios the median latency is below 150ms, even taking into account that on average 150ms of mixing delay are added. This is because the median of an exponential distribution is smaller than its mean.

In terms of entropy the variability of values is lower, particularly for high values of τ, where only outliers obtain less than 8 bits of entropy. Given the traffic volume and mixing per node we can see that no message is fully traceable (zero entropy) even in the worst cases (imbalanced approach with low τ). Comparing the results for the greedy approach (red boxes), we can see that for $\tau \leq 0.6$ the scheme reduces latency by 100 ms in the first quartile (from 300 ms to 200ms) and by 90 ms in the median (from 240 ms to 150 ms) compared to uniform routing ($\tau = 1$). This is at the cost of a median entropy loss of about 2 bits when $\tau = 0$ but just 0.4 bits when $\tau = 0.6$.

We can see that the median values for both entropy and latency follow a similar trend to the averages of the analytical results in Fig. 4a: latency stays constant between $\tau = 0$ and $\tau = 0.6$, and then increases steadily for higher values of τ. Entropy on the other hand increases steadily between $\tau = 0$

τ	0	.1	.2	.3	.4	.5	.6	**.7**	.8	.9	1
$\bar{l}_{mix}$	68.0	68.0	68.0	68.0	69.0	71.0	75.99	95.0	121.0	139.0	150.0
μ	44.0	44.0	44.0	44.0	43.6	43.0	41.3	**35.0**	26.3	20.3	16.6
H(T)	4.27	3.92	4.18	4.61	5.13	5.70	6.35	6.84	6.99	6.99	7.0
H(m)	6.48	6.63	6.75	7.0	7.28	7.62	7.98	**8.14**	7.68	7.0	6.4

TABLE III: Results for various values of τ and μ considering $l_b = 200$ ms.

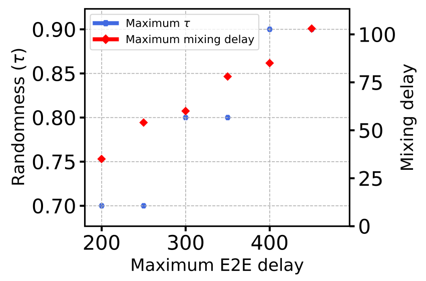


Fig. 6: Optimal τ and μ combinations for different values of l_b

and $\tau = 0.8$, but then in maxes out and higher τ does not anymore translate into higher entropy. Again, $\tau = 0.6$ offers the best tradeoff in these results, in terms of maximum gains (in terms of latency reduction) with minimal impact (in terms of diminished anonymity).

Further, we had similar findings when we compared the diversified arrangement with the random and unfavorable arrangements. We observed a much higher latency for the bad arrangement scenario and a slightly higher latency for the random arrangement.

2) Average end-to-end latency constraints: As we have seen in all the previous results, there is a tradeoff between latency and anonymity depending on parameter values, where reducing latency also diminishes anonymity. Mixnets with continuous-time mixnodes allow tuning this tradeoff though the choice of parameter μ, which determines the added mixing latency per hop. In addition, the parameter τ in our approach allows trading latency for anonymity by making the routing policy more or less biased towards faster routes.

We now consider scenarios were the latency tolerance of the traffic sent through the mixnet is given as a design constraint and examine anonymity for different combinations of μ and τ that lead to the same average latency. Let l_b be the target average latency that we want to optimize for, where l_b includes both the mixing delays at each layer and the propagation between mixnodes, i.e., l_b is a constraint on the aggregate latency from entry to exit of the mixnet. Given three mixing layers we note that: $l_b = \bar{l}_{mix} + 3\mu$, where $\bar{l}_{mix}$ varies depending on τ.

We perform a systematic parameter sweep to find the μ and τ value combinations that maximize anonymity while satisfying the latency constraint l_b. To do so, we first compute $\bar{l}_{mix}$ for each value of τ (recall that $\bar{l}_{mix}$ only depends on the routing policy which is tuned by τ). For a given value of τ and its corresponding $\bar{l}_{mix}$, we calculate the remaining budget for mixing latency by subtracting $\bar{l}_{mix}$ from l_b. We then compute: $\mu = \frac{l_b - \bar{l}_{mix}}{3}$. Once we have obtained the pairs of μ and τ that satisfy l_b, we evaluate anonymity using both the analytical approach (entropy of the transformation matrix) and the simulation approach (entropy of actual messages).

We show in Tab. III the results of applying this evaluation method considering $l_b = 200$ ms. Note that higher values of τ result in higher $H(T)$ but also in higher $\bar{l}_{mix}$, which decreases the remaining latency budget available for μ. Since higher μ *also* contributes to higher entropy for messages, the value of $H(m)$ is dependent on both τ and μ. Starting from $\tau = 0$ we can see that $H(m)$ grows as τ increases, thanks to increased randomness in the routing choices. The message entropy $H(m)$ peaks at 8.14 bits at $\tau = 0.7$. After that point, the amount of latency consumed on average by link propagation constrains the mixing delay μ to be too small to be effective. For random routing ($\tau = 1$), meeting the $l_b = 200$ms constraint requires setting $\mu = 16.6$ms, which leads to the worst (lowest) anonymity configuration of all options for that average latency with $H(m) = 6.4$ bit. Thus, for an average mixnet delay $l_b = 200$ ms, we find that $\tau = 0.7$ and $\mu = 35$ ms maximize the entropy of messages.

We performed the same evaluation for l_b values of 250, 300, 350, 400 and 450 ms. We plot in Fig. 6 the values of τ and μ that maximize the entropy for these values of l_b. We observe that increasing the latency budget l_b gives more room for randomness in routing (i.e., higher τ) with the mixing delay μ taking whatever latency budget is left; while tighter latency constraints require a more careful balance between routing randomness and mixing delay (i.e., leading to lower τ). Essentially, if l_b is high enough then latency-aware routing may not be required to provide the maximum possible anonymity, as simple uniform routing (corresponding to $\tau = 1$) provides a good configuration. However, for lower l_b constraints, latency-aware routing along with appropriate τ and μ selection are required to optimize the anonymity provided to routed messages while still meeting latency constraints.

3) Effect of network size: In this experiment we study how the proposed approaches fare at different network scales. In the previous experiments we considered a default network size of $N = 384$ mixnodes (i.e., a mixnet of 128x3), which is an order of magnitude larger than previous mixnet evaluations [16], [25]. For comparison with deployed systems, our baseline size of $N = 384$ nodes is slightly larger than the Nym mixnet [11] which, at the time of writing, includes $N = 240$ active mixnodes in three layers (i.e., size 80x3), randomly sampled each hour from a set of about 400 available nodes. Here we evaluate the routing entropy $H(T)$ and the average latency $\bar{l}_{mix}$ for various network sizes N: 102, 204, 306, 408 and 510 mixnodes arranged in three layers; considering three different values of τ: a low value of $\tau = 0.1$ that leads to nearly deterministic routing; a high value of $\tau = 0.9$ that corresponds to nearly uniform routing policies; and an intermediate value $\tau = 0.6$ that provides a good tradeoff between latency and routing entropy as shown in previous experiments. We consider that the clustering and diversification steps are applied to arrange mixnodes in layers and that greedy balancing is used to derive the routing policy.

The results are shown in Fig. 7. In terms of latency (dashed lines), we can see that for near-uniform routing ($\tau = 0.9$, blue) the average latency $\bar{l}_{mix}$ stays constant at about 110 ms regardless of network size. For more deterministic routing however ($\tau = 0.1$, green), a larger network size allows for more routing choices, including choices with low propagation latency that give more room for optimization: in networks of $N = 510$ mixnodes the latency $\bar{l}_{mix}$ is 55 ms, compared to 30 ms in smaller networks of just $N = 102$ mixnodes. For intermediate routing randomness ($\tau = 0.6$, red) we observe the best latency reduction as the network scales, which is cut by half from 60 ms to 30 ms. Note that for the larger network sizes the latency is a good (i.e., as low) for intermediate randomness ($\tau = 0.6$) and near-deterministic routing ($\tau = 0.1$).

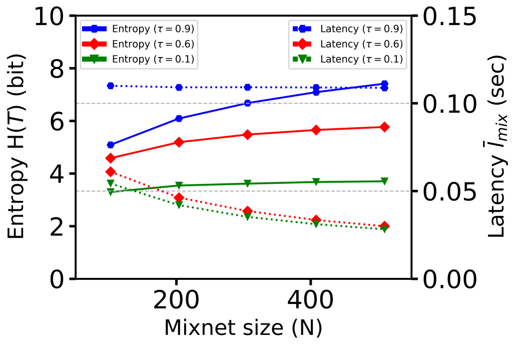


Fig. 7: Routing entropy $H(T)$ and average latency $\bar{l}_{mix}$ for different network sizes.

In terms of entropy (solid lines), we can see that increasing the network size is always beneficial in terms of entropy $H(T)$. The upper bound of $H(T)$ is given by $\log_2 W$, where W is the width of the mixnet layers. A larger W increases the mixnode candidates that may be the last hop in a message route. This is particularly the case for larger values of τ, that more heavily randomize the routing choices. As we can see in Fig. 7, the entropy increase is most pronounced for $\tau = 0.9$ (blue) while being rather minimal for $\tau = 0.1$ (green). For $\tau = 0.6$ the entropy gains are in between but still significant, particularly considering that this configuration provides the same latency as $\tau = 0.1$, which offers significantly lower entropy.

VI. Adversarial analysis

The proposed latency optimizations involve biased routing choices, compared to choosing routes uniformly at random in the vanilla case ($\tau = 1$). This bias can potentially be exploited by an adversary to compromise message anonymity. In particular, we are concerned about the 'mixnode adversary', which is an adversary that controls a subset of mixnodes. This adversary can trace messages end to end if their route is fully made up of adversarial nodes. We first evaluate LARMix by computing the fraction of fully corrupted paths given a routing policy (whose bias is dependent on τ) and various adversarial settings. In a separate experiment we assess the distribution of anonymity for all messages (instead of only messages with fully corrupted paths), taking into account that the adversary observes all communication links in addition to controlling a subset of mixnodes.

A. *Fraction of fully corrupted routes*

We denote by $|C|$ the total number of adversarial mixnodes and by C^ℓ the set of corrupted nodes in layer ℓ, of size $|C^\ell|$, such that $\sum_{\ell=1}^{L} |C^\ell| = |C|$. The distribution of corrupted mixnodes among the layers may vary widely depending on the instance. Taking into account that all mixnodes in the first layer are selected with equal probability $\frac{1}{W}$, we compute the probability of a fully corrupted route by taking into account the conditional probabilities that define the routing weights. For three layers we obtain:

$$\begin{aligned}
&\mathbb{P}\left[M^1 \subset C^1, M^2 \subset C^2, M^3 \subset C^3\right] \\
&= \sum_{m_i^1 \in C^1} \sum_{m_j^2 \in C^2} \sum_{m_k^3 \in C^3} \mathbb{P}\left[M^1 = m_i^1, M^2 = m_j^2, M^3 = m_k^3\right] \\
&= \sum_{m_i^1 \in C^1} \sum_{m_j^2 \in C^2} \sum_{m_k^3 \in C^3} \frac{1}{W} \mathbb{P}\left[M^2 = m_j^2 | M^1 = m_i^1\right] \\
&\qquad \mathbb{P}\left[M^3 = m_k^3 | M^2 = m_j^2\right]. \qquad (8)
\end{aligned}$$

Given an adversary able to run $|C|$ malicious mixnodes, we consider several possible adversarial settings. Note that adversaries may not have the power to determine the layer where their adversarial mixnodes are placed, particularly if the assignment process is randomized. We consider four possible scenarios as follows.

1) Random placement: In this scenario the adversary controls a subset of $|C|$ mixnodes that are chosen uniformly at random from all N mixnodes. Note that there is wide variability in the results of this adversarial setting depending on the experiment. To account for that we run a large number of instances, each with a freshly chosen malicious subset C, and compute the average fraction of corrupted paths over all experiments. We use this adversarial setting as a baseline.

2) Worst case: Finding the set of $|C|$ corrupted mixnodes that maximize the fraction of fully corrupted paths requires enumerating all possible combinations of C adversarial nodes, which scales with $\binom{N}{C}$. In other words, looking for the worst case configuration is an NP-hard problem. We develop a greedy algorithm to approximate this worst case, described in Appendix D. We note that even when the adversary possesses complete knowledge about all the parameters such as the location and routing weights of all the nodes, it is still a daunting task to find out which specific mixnodes maximize route compromise. Note that if mixnodes are randomly assigned to layers in a way that the adversary cannot bias, then the adversary may not have capability to arrange its mixnodes to maximize route capture. We nevertheless study worst case results as an upper bound on the fraction of paths that may be corrupted in the most unfavourable corner cases. Note that, at the other end of the spectrum, an adversary cannot compromise *any* message route whenever it fails to have at least one adversarial node in every layer of the mixnet.

3) Single location: In this setting we consider that the adversary runs $|C|$ nodes that are close to each other, i.e., in the same geographical region. This will ensure that those mixnodes are part of the same cluster after the clustering phase. Subsequently, the arrangement algorithm distributes nodes in the same cluster among the different layers. Such diversification benefits the adversary as its nodes will most likely be in different layers and have high routing weights among themselves, which increases the probability of being chosen as part of the same route.

4) Diverse locations: For comparison we also study the opposite case, where adversarial nodes are placed in distant geographical locations, ensuring that they are assigned to different clusters. Having the malicious nodes in different clusters should be unfavorable for the adversary compared to the previous case, as it increases the chances that adversarial mixnodes are assigned to the same mixnet layer. Even when adversarial nodes are in adjacent layers, their routing weight will not be high because the link between them is not low latency. This effect is more pronounced if routing is more biased towards low latency links (i.e., with small τ as well as with imbalanced approaches).

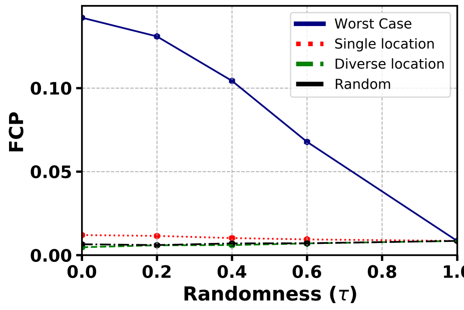


Fig. 8: Fraction of corrupted paths vs τ in different adversarial settings (greedy balancing).

B. Results

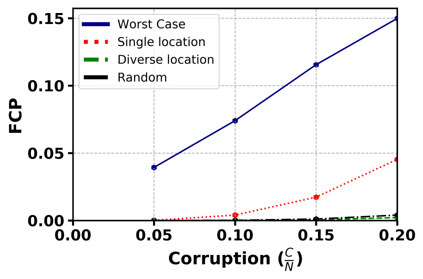
We performed three different experiments considering a network of size 96 mixnodes (32 per layer). We use a smaller network than in previous experiments due to the variance of results (and the need for large numbers of samples to compute accurate averages) in larger networks. We use a cluster size of 5 and run 1000 experiments per setting. Other parameters are as defined in Tab. II In the first experiment we quantify the fraction of corrupted paths (FCP) for different values of τ and adversarial settings, considering that 20% of the nodes in the network are adversarial (i.e., 19 out of 96 mixnodes are adversarial). The second experiment fixes τ at an intermediate value while varying the number of corrupted nodes to be between 5% and 20% of all the nodes, and studies the FCP for the different levels of compromise. Finally, the third experiment computes entropy for the full set of messages, and not just the fraction of corrupted paths. We calculate the per-message entropy $H(m)$ using simulations, with varying τ and for each of the three proposed balancing approaches.

1) FCP vs τ: Fig. 8 shows the results for FCP in the four adversarial settings, considering that 20% of all nodes are compromised and a routing policy balanced with the greedy approach. As expected, all settings lead to the same FCP when $\tau = 1$, which corresponds to uniform routing policies where the adversary gains no advantage from compromising some nodes instead of others. As τ decreases and the routing policy becomes more biased towards faster routes, we see different effects depending on the adversary. The 'worst case' adversary gains a significant advantage for low values of τ. We note however that achieving this worst case would require the adversary to selectively compromise the mixnodes that maximize FCP *after* the routing policy has been derived, since it is not possible to *a priori* predict which mixnodes will maximize FCP. Note also that given a routing policy, even *identifying* those mixnodes requires solving an NP-hard problem, which may be impractical in systems with churn where routing policies are frequently updated, as is the case in deployed networks. The best practical strategy an adversary can follow is to place all the malicious nodes in close proximity (i.e., in a 'single location'). As we can see in the results, this adversarial strategy outperforms the random placement in terms of FCP. This is because adversarial mixnodes have higher than average routing weights among themselves, enabling an increased rate of full route compromise compared to 'random'. The adversarial advantage is however rather limited, even for low values of τ. The 'random' compromise of mixnodes leads to a flat FCP rate on average — though the actual FCP has high variance depending on the adversary's "luck" with the corrupt selection. Finally, locating adversarial mixnodes in distant places ('diverse location') slightly underperforms the 'random' baseline, as those nodes have high propagation latency between themselves and consequently a low weight in the routing policy, making fully corrupted paths less likely to be chosen.

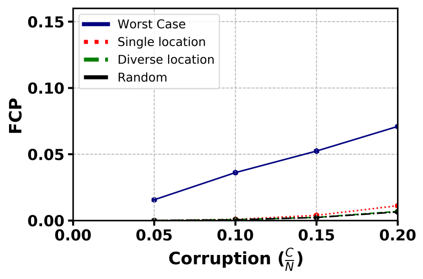
2) FCP vs number of corrupted mixnodes: Here we examine the effect of varying $|C|$ on the fraction of corrupted paths while keeping the routing randomness constant at $\tau = 0.5$. We consider adversaries that corrupt 5%, 10%, 15% and 20% of all the nodes, and the adversarial nodes are chosen according to the four adversarial settings introduced earlier in this section. The network is arranged with the clustering and diversification algorithms and we consider imbalanced (low latency) routing (Fig. 9a), greedy (Fig. 9b), and naive balancing (Fig. 9c).

The results in Fig. 9 show that balancing the network significantly reduces the adversarial advantage compared to the low latency approach, in particular for the worst case adversary as well as for the adversary that places all adversarial nodes in the nearby geographical locations. Moreover, as expected, increasing the number of malicious nodes increases the FCP in all scenarios. The worst case is significantly worse than all the other adversarial settings, though it may be infeasible for the adversary to achieve such favourable placement for its mixnodes. Placing the malicious nodes in a single location is a better strategy than random placement for improving attack results. The single location approach provides however much lower adversarial success compared to the worst case, particularly when a balancing approach (whether naive or greedy) is used.

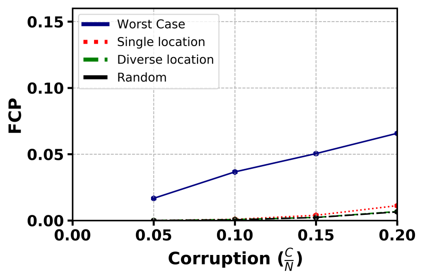
3) Entropy vs τ: In the last experiment we consider an adversary that observes all network links in addition to corrupting a set of mixnodes. Besides fully deanonymizing messages routed via a fully corrupted path, this adversary is also able to reduce anonymity for messages that are not fully traceable.



(a) Low Latency



(b) Greedy



(c) Naive

Fig. 9: Fraction of corrupted paths V.S. Fraction of corrupted mixnodes for $\tau = 0.5$

We measure the entropy $H(m)$ of sample messages using our simulator, considering that 20% of nodes are malicious. We modify the entropy calculation formula to account for adversarial mixnodes (following the approach described in [16]): when messages traverse a corrupt mixnode, the adversary knows the exact input-output correspondence, rather than having a probabilistic guess, meaning that adversarial nodes do not add any entropy to messages they route.

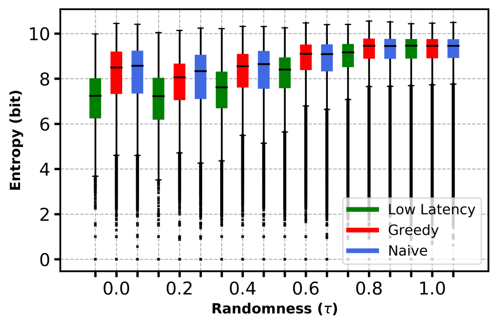


Fig. 10: Entropy vs τ for single location adversary.

Fig. 10 shows the results considering a single location adversary and the three proposed balancing approaches. We observe that this strategy leads to an median drop of 0.3 bit in entropy when we select $\tau = 0.6$, compared to vanilla routing ($\tau = 1$). The entropy loss is similar for greedy and naive balancing, but it becomes 1 bit of loss for an imbalanced (low latency) network. We note that compared to Fig. 5 the median and higher values of entropy are similar, while in Fig. 10 the lower end of the range and outliers have lower values, as they correspond to messages that go through routes that are partially or fully compromised. We obtained similar results when considering the random selection adversary, with 0.3 bits drop in entropy for the balanced approaches. Overall, we observe that the bias of latency-aware routing does provide a potential gain to the adversary but, unless the adversary is able to engineer a *worst case* arrangement, the attack impact is rather limited in comparison to scale of latency reduction achieved.

VII. Comparison with CLAPS

Given a set of constraints and an optimization function, CLAPS [26] can output a routing policy (i.e., set of routing weights) that meets all the constraints while maximizing the objective function, by solving a linear program. Even though CLAPS was originally proposed for improving Tor's path selection, by setting the appropriate constraints the CLAPS framework can be used to obtain a latency-aware routing policy for mixnets, as an alternative to LARMix.

To instantiate CLAPS we must first define the objective function that we want to optimize. In our case we define this as the minimization of the weighted average of link latency, given by:

$$min(\sum_{\ell} \sum_{i,j=1}^{W} \gamma_{ij}^{\ell} * l_{ij}), \tag{9}$$

where γ_{ij}^{ℓ} represents the routing weight from mixnode m_i^{ℓ} in layer ℓ to mixnode $m_j^{\ell+1}$ in the next layer, and l_{ij} is the average propagation latency between m_i^{ℓ} and $m_j^{\ell+1}$. Note that l_{ij} is provided as input to the framework whereas the values of the weights γ_{ij}^{ℓ}, which define the resulting routing policy, are the desired output obtained at the end of the process. To conform a valid routing policy, the weights γ_{ij}^{ℓ} must additionally meet a number of constraints: (1) their value must be between zero and one ($0 \leq \gamma_{ij}^{\ell} \leq 1$); (2) all the weights leaving a node m_i^{ℓ} must add up to one to represent a valid probability distribution ($\sum_{j=1}^{W} \gamma_{ij}^{\ell} = 1$); and all the weights incoming to a mixnode $m_j^{\ell+1}$ must add up to one to ensure that the traffic load is balanced ($\sum_{i=1}^{W} \gamma_{ij}^{\ell} = 1$).

CLAPS lacks a built-in tuning mechanism to allow trading average latency for entropy of the routing policy. Note that it is also not possible for CLAPS to optimize for entropy in its objective function, since entropy is a non-linear function. We introduce tuneability of the latency-entropy tradeoff for CLAPS by defining a parameter $0 \leq \tau \leq 1$ that further constrains the range of values for the routing weights γ_{ij}^{ℓ}. As before, larger values of τ lead to more uniform (random) routing, while lower values of τ lead to more deterministic routes (optimized for low latency). The tuning constraint is formulated as:

$$(\frac{1}{W})^{\frac{1}{\tau}} \leq \gamma_{ij}^{\ell} \leq (\frac{1}{W})^{\tau} \tag{10}$$

To evaluate this approach, we solved the CLAPS linear programs for the objective function and constraints defined above and obtained the routing weights γ_{ij}^{ℓ}. These weights were then evaluated with our analytical approach to quantify the average latency $\bar{l}_{mix}$ and the entropy $H(T)$ of the resulting routing policy. We compared the results obtained from CLAPS to those of LARMix (for both naive and greedy balancing), for different values of τ. In order to assess the latency-entropy tradeoff, we compute $\frac{H(T)}{\bar{l}_{mix}}$, a quantity that is maximized for the best tradeoff, since it grows both with higher entropy and with lower latency, both of which are desirable. The results are shown in Fig. 11. We can see that LARMix with greedy balancing outperforms CLAPS (as well as LARMix with naive balancing) by offering the best entropy-latency tradeoff for all values of τ. As in previous experiments, we see that $\tau = 0.6$ provides a good entropy-latency tradeoff.

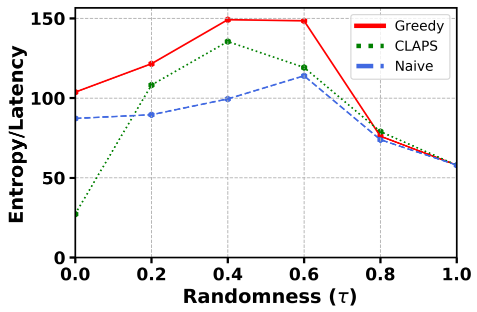


Fig. 11: Entropy/Latency $\frac{H(T)}{\bar{l}_{mix}}$ vs τ for CLAPS and LARMix.

Additionally, we recorded the time it takes for CLAPS and LARMix to find a solution for the set of γ_{ij}^{ℓ} values. We found out that it takes $\approx$ 300x more time for CLAPS to find the solution using linear programs than for LARMix. Further, we observed a non-linear increase in time while scaling the network (doubling N resulted in tripling the runtime). In a deployment setting, one may need to frequently recompute routing weights (these are refreshed about once an hour in Tor and Nym). Scaling CLAPS to the current Tor network size of >6000 relays would require substantially more than an hour. LARMix on the other hand can be used to recompute routing weights on a frequent basis.

VIII. Discussion

A. Comparison with simpler approaches

We now consider approaches simpler than LARMix that may offer similar tradeoffs for latency and anonymity. First, we consider reducing the number of mixnet layers to two, while maintaining a uniform routing policy. This eliminates one link from end-to-end connections, thus reducing latency. We devised an experiment considering 2-layers with uniform routing, and compared it with the 3-layer LARMix routing with $\tau = 0.6$. For both cases we considered a total average mixing latency of 150 ms, distributed equally among 2-layers in the first case (75 ms of average mixing latency per layer) and among 3-layers in the second case (50 ms of average mixing latency per layer). We performed the experiment for 1000 iterations. The results from the experiment are summarized in Tab. IV below.

As we can see, the average latency in 2-layer random routing is higher in comparison to LARMix for both analytical

Parameter	2-layer random routing	3-layer LARMix
Analytical Latency	46.9 ms	34.5 ms
Simulation Latency	170.2 ms	150.3 ms
Simulation Anonymity	7.7 bits	8.8 bits

TABLE IV: Latency and anonymity comparison for natural trade-offs.

as well as simulation evaluation. This result may be slightly counter-intuitive but can be explained. The LARMix approach optimizes for selecting lower latency paths by removing high latency outliers that can be selected in random routing, leading to higher average latency across all message paths. Moreover, as expected, the anonymity in 2-layers is reduced in comparison to 3-layers LARMix (7.7 bits from 8.8 bits). Thus, overall, the simple approach of removing a layer in the mixnet for latency reduction is not better than LARMix neither in terms of latency nor anonymity.

Other approaches such as creating multiple localized mixnets instead of one global mixnet, or considering a single cascade of mixnodes per client instead of a layered mixnet with a global routing policy, can also reduce latency. The evaluation such approaches requires assumptions on the distribution of client locations to determine the size of the localized networks and the amount of traffic routed, making it hard to compare such approaches on equitable grounds. Note however that we can expect a significant anonymity reduction in such approaches, as clients are partitioned in subsets rather than forming one large anonymity set.

B. Considering endpoint connection latency

In terms of latency, all the evaluations performed in this work report the minimization of l_{mix}. From a client's usability perspective the end-to-end (client-to-destination) latency ($\bar{l}$) is the most important value.

However, it is non-trivial to determine what the actual distribution of clients may be in a live network, which makes it challenging to perform realistic simulations of such scenarios. We can approximate the average latency between clients and the mixnet and add a constant client latency to the existing latency evaluation. To that end, we randomly selected 350 RIPE nodes (considered to be clients) and recorded their latency to all the mixnodes in the first layer.[6] The average client latency obtained from this evaluation was 63.2 ms. Thus, a client that uses the mixnet with uniform routing options would observe on average 246.4 ms propagation latency i.e., 63.2 ms (client-to-mixnet latency) + 120 ms (mixnet-propagation latency) + 63.2 ms (mixnet-to-destination latency). A client using LARMix with greedy balancing and $\tau = 0.6$ will reduce it to an average of 160.9 ms (63.2+34.5+63.2); a reduction of 34.8% in client-to-destination propagation latency. If we also consider the 'mixing' latency then the reduction in end-to-end latency is 22%.

[6]We again considered the RIPE dataset as it has globally scattered endpoints representing good coverage.

C. RIPE vs Nym latency dataset

In our evaluation, we used the RIPE dataset for obtaining a distribution of latency between pairs of mixnodes. This is a good data source for generic evaluation. However, the evaluation can be made even more realistic if the latency values are derived from a deployed mixnet such as Nym. These latency values have more recently become accessible using the Verloc protocol, which is run by each mixnode to periodically measure the latency to every other mixnode.[7] We thus obtained the latency values between pairs of mixnodes and compared this Nym mixnet latency dataset to the RIPE dataset. As can be seen in Fig. 12, CDF of the distributions of both the datasets are similar, with the Nym dataset having a slightly lower latency on average. Overall, we can conclude that our existing evaluation captures realistic performance gains that would be observed if our methods are deployed in a real mixnet.

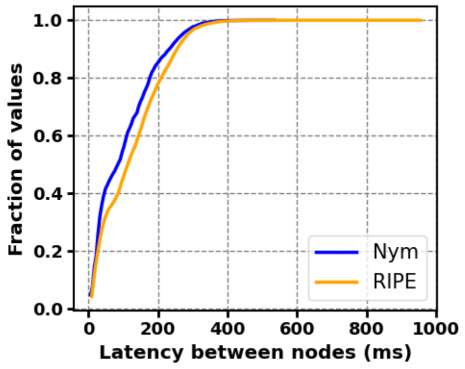


Fig. 12: Nym vs RIPE latency distribution comparison.

IX. Conclusion and future work

LARMix is a novel solution to location-aware routing in layered mixnets, a problem that had not been tackled before. LARMix includes separate methods for both arranging mixnodes in layers and for deriving a routing policy given an arrangement. LARMix can additionally ensure that the traffic load is balanced across nodes while selecting low-latency paths. Our evaluation of LARMix shows there is ample scope for reducing latency while maintaining anonymity even if the adversary can compromise some nodes. Overall, we believe this work is a significant step towards making mixnets more widely usable for applications with latency constraints.

Acknowledgment

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

[7]These values can be accessed by querying each mixnode IP on port 8000 and accessing the `/verloc` resource.

References

[1] M. Akhoondi, C. Yu, and H. V. Madhyastha, "Lastor: A low-latency as-aware tor client," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 476–490.

[2] M. AlSabah, K. Bauer, T. Elahi, and I. Goldberg, "The path less travelled: Overcoming tor's bottlenecks with traffic splitting," in *Privacy Enhancing Technologies: 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings 13*. Springer, 2013, pp. 143–163.

[3] R. Annessi and M. Schmiedecker, "Navigator: Finding faster paths to anonymity," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 214–226.

[4] A. Barton and M. Wright, "Denasa: Destination-naive as-awareness in anonymous communications," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, 2016.

[5] A. Barton, M. Wright, J. Ming, and M. Imani, "Towards predicting efficient and anonymous tor circuits," in *27th $USENIX$ Security Symposium ($USENIX$ Security 18)*, 2018, pp. 429–444.

[6] I. Ben Guirat, D. Gosain, and C. Diaz, "Mixim: Mixnet design decisions and empirical evaluation," in *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, 2021, pp. 33–37.

[7] D. Chaum, D. Das, F. Javani, A. Kate, A. Krasnova, J. De Ruiter, and A. T. Sherman, "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*. Springer, 2017, pp. 557–578.

[8] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[9] G. Danezis and I. Goldberg, "Sphinx: A compact and provably secure mix format," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 269–282.

[10] D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency-choose two," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 108–126.

[11] C. Diaz, H. Halpin, and A. Kiayias, "The nym network," 2021.

[12] C. Diaz, S. J. Murdoch, and C. Troncoso, "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*. Springer, 2010, pp. 184–201.

[13] C. Diaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers*. Springer, 2003, pp. 54–68.

[14] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.

[15] J. Geddes, M. Schliep, and N. Hopper, "Abra cadabra: Magically increasing network utilization in tor by avoiding bottlenecks," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, 2016, pp. 165–176.

[16] I. B. Guirat and C. Diaz, "Mixnet optimization methods," *Proceedings on Privacy Enhancing Technologies*, vol. 1, p. 22, 2022.

[17] K. Hogan, S. Servan-Schreiber, Z. Newman, B. Weintraub, C. Nita-Rotaru, and S. Devadas, "Shortor: Improving tor network latency via multi-hop overlay routing," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1933–1952.

[18] G. M. Jacquez, "Spatial cluster analysis," *The handbook of geographic information science*, vol. 395, no. 416, 2008.

[19] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go MIXes: Providing probabilistic anonymity in an open system," in *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.

[20] K. Kohls and C. Diaz, "Verloc: Verifiable localization in decentralized systems," *arXiv preprint arXiv:2105.11928*, 2021.

[21] ——, "{VerLoc}: Verifiable localization in decentralized systems," in

31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 2637–2654.

[22] R. Nithyanand, O. Starov, A. Zair, P. Gill, and M. Schapira, "Measuring and mitigating as-level adversaries against tor," *arXiv preprint arXiv:1505.05173*, 2015.

[23] A. Panchenko, F. Lanze, and T. Engel, "Improving performance and anonymity in the tor network," in *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2012, pp. 1–10.

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

[25] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The loopix anonymity system," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1199–1216.

[26] F. Rochet, R. Wails, A. Johnson, P. Mittal, and O. Pereira, "Claps: Client-location-aware path selection in tor," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 17–34.

[27] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers 2*. Springer, 2003, pp. 41–53.

[28] C. E. Shannon, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[29] M. Sherr, M. Blaze, and B. T. Loo, "Scalable link-based relay selection for anonymous routing," in *Privacy Enhancing Technologies: 9th International Symposium, PETS 2009, Seattle, WA, USA, August 5-7, 2009. Proceedings 9*. Springer, 2009, pp. 73–93.

[30] F. Shirazi, M. Simeonovski, M. R. Asghar, M. Backes, and C. Diaz, "A survey on routing in anonymous communication protocols," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–39, 2018.

[31] R. N. Staff, "Ripe atlas: A global internet measurement network," *Internet Protocol Journal*, vol. 18, no. 3, pp. 2–26, 2015.

[32] Y. Sun, A. Edmundson, N. Feamster, M. Chiang, and P. Mittal, "Counter-raptor: Safeguarding tor against active routing attacks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 977–992.

[33] G. Wan, A. Johnson, R. Wails, S. Wagh, and P. Mittal, "Guard placement attacks on path selection algorithms for tor," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, 2019.

[34] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware path selection for tor," in *Financial Cryptography and Data Security: 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers 16*. Springer, 2012, pp. 98–113.

Appendix

A. LARMix with variable node capacities

LARMix assumes a fixed and equal capacity of all the nodes in the network. This assumption is common in Loopix-based mixnet designs which is the target architecture of LARMix. However, LARMix can be easily extended to work for nodes with different capacities. The major change will be required in the arrangement of the mixnet where we need to ensure that diversifying the mixnodes does not lead to high disparity in capacities of different layers. This can be achieved by selecting mixnodes from clusters considering their capacities and introducing bounds for the combined capacity of nodes within a layer. The routing algorithm will not be affected by the change in capacities. The balancing algorithms can also easily work for variable capacities by defining the total load on a node based on its capacity and accordingly calculating if a node is underloaded or overloaded. Similar considerations will be taken when the extra load from overloaded nodes is distributed to the underloaded nodes. The details on how it can be achieved are presented below.

To extend the balancing algorithms (Greedy and Naive) for generalized node capacities we consider the individual capacity of the node while labelling them as underloaded or overloaded. Thus in Algo. 2 and Algo. 1 while labelling for overloaded nodes, we will modify the condition to be $\sum_i \gamma_{ij}^{\ell} > node\_capacity_j$ instead of $\sum_i \gamma_{ij}^{\ell} > 1$ for all nodes j. Similarly, we will update for underloaded and balanced nodes labelling. While distributing the load of overloaded nodes to the underloaded ones, we will normalize according to the node capacities. However, as previously stated, for Loopix based mixnet designs the node capacities are considered to be equal and thus in the design of LARMix we take the same assumption.

B. Effect of change in cluster size

As previously highlighted, one needs to appropriately select the size of the cluster to avoid corner cases that compromise the effectiveness of the clustering process. For instance, setting K to be equal to the total number of mixnodes (N) or having only one cluster would be equivalent to performing random routing. However, barring the corner cases, selecting an appropriate K is in itself a non-trivial task as it may depend on factors such as the number of mixnodes, their geographical diversity, consideration of malicious nodes etc. Thus, we devise experiments to study the impact of the number of clusters on the entropy and latency offered and select the value of K that offers the maximum benefit. We performed an experiment where we fixed the value of $\tau = 0.5$ and varied the cluster sizes to obtain the corresponding entropy and latency values for the analytical as well as the simulation setting.

As can be seen in Tab. VI, for most of the lower values of K (until 40) there is no impact on entropy or latency. However, when the values of clusters are high, both the entropy of the transformation matrix and entropy of output messages increase. This is due to the fact that increasing the number of clusters makes the diversification algorithm less effective as it becomes close to a random arrangement, leading to more uniform paths and higher entropy. However, it is important to note that this increase in the number of clusters also has a negative impact on latency, as seen in the table. A high number of clusters can lead to high end-to-end latency, while a low number like 2 can prevent the diversification algorithm from operating properly. Therefore, to achieve the best trade-offs between latency and anonymity, we recommend using a moderate number of clusters, such as 5 - 20, to achieve better tradeoff in anonymity and latency while minimizing the computational overhead. Note that we can even select values of clusters to be 40, but that may increase the computational load on the diversification algorithm

C. Algorithms for different approaches

D. Worst case adversary

We developed a greedy algorithm to find the worst-case adversary that maximizes the fraction of corrupted paths for a fixed number of adversary nodes in the network. We call this approach greedy for fairness, as it selects mixnodes equally in different layers. It is non-trivial to select the set of corrupted

Symbol	Meaning
Γ^ℓ	Scattering Matrix of layer ℓ
$\mathbf{T}$	Transformation Matrix
N	Number of mixnodes
K	Number of clusters
L	Number of mixing layers
W	Number of mixnodes per layer
C	Set of corrupted mixnodes (size $\|C\|$)
C^ℓ	Set of corrupted mixnodes in layer ℓ (size $\|C^\ell\|$)
m_i^ℓ	i-th mixnode of layer ℓ
M^ℓ	Mixnode in a message path at layer ℓ
$\mathbb{P}[M^\ell = m_i^\ell]$	Probability of selecting mixnode m_i^ℓ at layer ℓ
$\mathbb{P}[M^{\ell+1} = m_j^{\ell+1} \| M^\ell = m_i^\ell]$ γ_{ij}^ℓ	Routing weight (probability) of node m_i^ℓ sending messages to successor mixnode $m_j^{\ell+1}$
$\tilde{\gamma}_{ij}^\ell$	Modified routing weight when balancing the network load.
$\mathsf{H}(T)$	Entropy of the transformation matrix
$\mathsf{H}(m)$	Entropy of message m
μ	Average mixing delay
δ	Average processing latency per hop
l_{ij}	Propagation latency between mixnodes i and j
l_{e2e}	Average end-to-end latency
l_{mix}	Average aggregate mixnet propagation latency
$l_{s,mix}$	Propagation latency between sender s and mixnet
$l_{mix,r}$	Propagation latency between mixnet and recipient r

TABLE V: List of key notations and variables

Analysis	Low latency (Imbalance)						Greedy						Naive					
Number of clusters k	2	5	20	40	100	300	2	5	20	40	100	300	2	5	20	40	100	300
Average link delays	10	10	10	10	10	40	30	30	34	34	40	50	50	50	50	50	60	70
E2E latency	126.9	126.8	126.7	126.2	126.3	152.3	145.0	144.1	145.1	145.6	153.3	166.9	160.3	159.7	160.5	161.1	167.7	174.2
Entropy $\mathsf{H}(T)$	3.0	3.0	3.0	3.0	3.1	3.4	5.5	5.5	5.5	5.5	5.8	6.0	6.1	6.1	6.1	6.1	6.2	6.5
Entropy $\mathsf{H}(m)$	7.5	7.5	7.5	7.5	7.8	8.3	8.0	8.0	8.0	8.0	8.2	8.3	8.1	8.1	8.1	8.1	8.1	8.4

TABLE VI: Effect of varying cluster size (K) on LARMix.

Algorithm 1 Greedy Balanced Algorithm

```
1: procedure A BALANCED MIX-NET
2:     Input: Scattering matrix
3:     Summation of scattering matrix over columns
4:     Label the mixnodes
5:     if The next mixnode is overloaded (i.e. $\sum_i \gamma_{ij}^\ell > 1$) then label it as 1.
6:     if The next mixnode is balanced (i.e. $\sum_i \gamma_{ij}^\ell = 1$) then label it as 0.
7:     if The next mixnode is under loaded (i.e. $\sum_i \gamma_{ij}^\ell < 1$) then label it as -1.
8:     Multiply the columns with label 1 by the inverse of $\sum_i \gamma_{ij}^\ell$.
9:     Save the remaining probabilities for these columns and distribute them among the columns with label equal to -1.
10:    Consider the priorities to be biased based on the proximity of mixnodes when distributing over underloaded mixnodes.
11:    Verify if the modified scattering matrix is balanced (i.e. $\sum_i \tilde{\gamma}_{ij}^\ell = 1$).
12:    If not, repeat the algorithm.
13:    The Greedy Algorithm stops when a decimal precision for the balance criteria is achieved.
```

Algorithm 2 Naive Balanced Algorithm

```
1: procedure A BALANCED MIX-NET
2:     Input: Scattering matrix
3:     Summation of scattering matrix over columns
4:     Label the mixnodes
5:     Repeat steps 5 - 8 from the greedy algorithm
6:     Save the remaining probabilities for these columns.
7:     First, calculate the probability distribution of assignments from underloaded mixnodes (i.e. $\mathsf{Pi} = \frac{1-\mathsf{Load_i}}{\sum_j 1-\mathsf{Load_j}}$).
8:     Distribute the remaining loads among the underloaded mixnodes based on the calculated probability distribution.
```

Algorithm 3 Greedy For Fairness

```
1: procedure BEST SUBSET OF CORRUPTED MIXNODES
2:     Select one set of corrupted mixnodes out of $\binom{W}{\frac{C}{3}}$ possible ways(If there is a limitation decrease the set to $3N$ cases).
3:     For remained layers select $\frac{C}{3}$ mixnodes from $ith$ mixing layer such that $\Sigma_j \Gamma^i[j]$ where $j$ describes corrupted mixnodes
4:     Try different options of the set to maximize the mentioned probability.
5:     Return Set of corrupted mixnodes along with the fraction of corrupted paths.
```

mixnodes fairly from each layer to maximize the fraction of corrupted paths. So in this algorithm, we first choose $\frac{C}{3}$ from the first mixing layer in a uniform way. Then we choose $\frac{C}{3}$ mixnodes in the next mixing layer such that they receive the maximum streams from the chosen corrupted mixnodes in the previous mixing layer (i.e. $Max\ \Sigma_j \Gamma^i[j]$). Then we compute the fraction of corrupted paths and repeat for different sets of corrupted mixnodes from the first mixing layer. Ideally, we should try $\binom{W}{\frac{C}{3}}$ cases, but it may lead to an exhaustive search and to avoid it we check $3N$ cases out of all possibilities.

We performed experiments on a smaller topology of 30 nodes to compare the results of the greedy approach with an exhaustive search and observed that the developed algorithm produces almost the same result as that obtained from the exhaustive search.

E. Intuition behind the routing formula

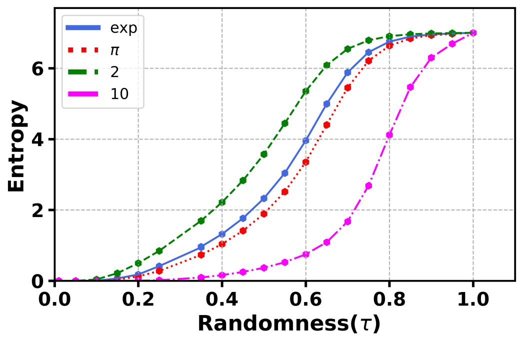


Fig. 13: Entropy vs Tau for varying values of the coefficient.

In Sec. IV-B we introduce the formula used to bias the network towards low latency paths. Now, we discuss the rationale behind selecting different values for the formula. Overall our idea was to use the formula to bias the network while giving the flexibility to the network designer about how much they would want to bias it. The actual parameter that we want to base this bias on is the latency between hops. We include it in the formula in the numerator as $\frac{1}{l_{ij}}$. We then multiply a coefficient of $\frac{1}{e}$ to control the fraction of the latency values to be considered in the probability distributions[8]. However, the extent to which we want to bias towards low latency is controlled by τ. Thus, we need to use the latency factor and the coefficient in an intelligent way such that low values of τ makes the next-hop (or path) selection deterministic, while a high value should make it random. This means that for a value of $\tau = 0$, the probability should be 1 for selecting the lowest latency next hop while the probability of selecting any other next hop should be 0. On the other hand, for the value of $\tau = 1$, the probability of selecting any next hop should be the same $(1/W)$.

To achieve this, we raise the latency factor to a value of $1-\tau$ which ensures that with an increasing value of τ, the value of latency that will be considered in probability calculation will be minimized. Similarly, for the coefficient, we raise it to the power of $(\frac{R_i(j)}{\tau})(1-\tau)$. The factor $(\frac{R_i(j)}{\tau})(1-\tau)$ encompasses multiple considerations. The component $\frac{(1-\tau)}{\tau}$ will range from ∞ to 0 for τ ranging from 0 to 1. This will result in a variation between 0 to 1 for the value of coefficient when τ varies from 0 to 1 respectively. We multiply the component with a variable $R_i(j)$ that represents the index of the next-hop node when they are arranged in ascending order of latency. For instance, in a network with $W = 10$, the node with the least latency will have the value of j as 0 and the highest latency value will have a value of 9. Thus, when $\tau = 0$, the probability distributions will consist of a value of 1 for the 0th node and a value of 0 for all other nodes thereby achieving the desired objective.

[8]One could even consider other values of the coefficient such as $1/2$, $1/3$, *etc.*, depending on how steep they want the bias in distributions to be. $1/e$ provides a good range of bias in comparison to other values (ref. Fig. 13).

F. Relation between H(T) and H(m)

We use two ways of measuring anonymity throughout our evaluation. One was the analytical approach using the entropy of the transformation matrix (H(T)) and the other was the entropy of simulation (H(m)) calculated by mapping the target messages to all potential input messages. We note that H(m) is a sum of H(T) and the randomness introduced by the mixing of messages. Fig. 14 illustrates the entropy of the transformation matrix and the results obtained from simulations, considering the previously established setting for clustering and diversification. As the value of τ increases, both entropies increase, due to the randomization of routing. However, the difference in increase is almost constant. The gain is majorly due to an increase in the value of H(T).

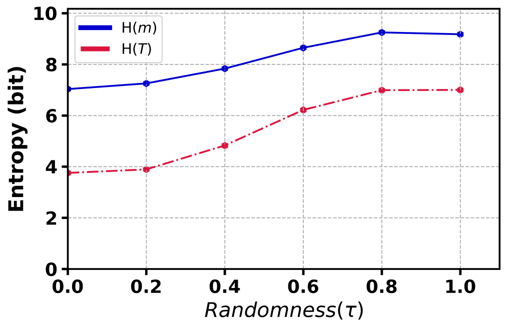


Fig. 14: Relation between H(T) and H(m).

G. Artifact Appendix

1) Description & Requirements: We performed the evaluation of LARMix in two distinct settings, the analytical setting and the simulation setting. For the analytical setting, we modeled and represented the mixnet via matrices and performed appropriate operations. For the simulation setting, we modeled the mixnet in the form of a discrete event simulator using the Simpy Python framework. We defined multiple classes to specify the functioning of different components of the network. Specifically, we define the functionality of a message that traverses the network, the mixnode that performs appropriate operation on messages, the mixnet that control the functionality and structure of the network, etc. Additionally, we also model different operations such as clustering, arrangement, routing and latency that are applied on these components. All these components and operations are used in conjunction to perform the simulation with specific tasks and generate appropriate results. Overall, the analysis scripts and evaluation code is written in Python with about 10K lines of code. For both evaluation settings, we need to provide the latency and geographical coordinates of the mixnodes. To evaluate our approaches on realistic data, we considered the RIPE atlas dataset because it provides a good number (568) as well as a good geographical distribution of nodes in the dataset. The code has been tested to work on a server with Python 3.8 and higher on Ubuntu 20.04 OS with 128 GB RAM and

2 TB disk space. However, the scaled-down version of the experiments (fewer iterations) can be easily executed on a standard laptop/desktop with 16 GB RAM and 50 GB hard disk space.

2) How to access: You can access the artifact of LARMix through the following link on permanent storage with DOI: https://doi.org/10.5281/zenodo.8311051 or also at the Github link: https://github.com/larmix/larmix.

3) Artifact Installation & Configuration: To evaluate the artifact, one needs to set up a Python environment with Python version 3.8 or higher. The dependencies of the code can be easily installed using the `requirements.txt` file bundled with the code. Once this is achieved, the results can be easily generated by executing the `Main.py` file.

4) Major Claims: In this section we mention the main claims of LARMix:

- (C1): LATENCY MINIMIZATION We observed a reduction in latency by 8x and 3.5x when using the imbalance or greedy balance approach as compared to vanilla mixnet routing for a mixnet size of $3 * 128$ with 400 iterations and $\tau = 0.6$. The anonymity loss was 2.2 bits and 0.8 bits of entropy respectively for the two cases. These results are demonstrated in Figure 4a.
- (C2): MAXIMUM ENTROPY WITH END-TO-END CONSTRAINTS Setting a boundary of 200 ms for end-to-end latency of the mixnet with a size of $128 * 3$ and 1000 iterations, we found that the best tradeoff for the highest value of message entropy occurs when $\tau = 0.7$ and the mixing delay is equal to 35ms. This is demonstrated by Table 3 in the original submission.
- (C3): ADVERSARY ANALYSIS We captured the fraction of corrupted paths (FCP) for varying values of adversarial corruption (5%, 10%, 15%, and 20%) and observed two trends. First is that the FCP for the imbalanced approach (Figure 9a) is higher when compared with the balancing approaches (greedy and naive in Figure 9b and 9c respectively). Second is that the best practical approach (barring the worst-case) for the adversary to maximize FCP is to have all corrupted mixnodes in a single location, which is true across Figures 9a, 9b, and 9c.
- (C4): NATURAL TRADEOFFS We compared LARMIX with $\tau = 0.6$ and greedy balancing to a standard uniformly routed mixnet with two layers of mixnodes. We observed that the LARMIX system provides lower analytical and simulation latency while offering higher message entropy. This is not yet demonstrated in the original paper but is part of a minor revision.

5) Evaluation: In this section, we tie each of the claims in the previous section with an experiment that can be used to verify them.[9] All the results will automatically be stored in `Figures` and `Tables` folders.

1) **Experiment (E1):** [Figure 4a and Figure 5] [15 min]: In this experiment, we analyze LARMix with clustering and diversification to arrange mixnodes to layers. We expect a reduction in latency and improved anonymity, as described in **C1**. In Figure 5, we observe that the reduction in latency for the greedy approaches is less than 1 bit, while the latency reduces by approximately 90ms.
   [Preparation and Execution] Uncomment the line with the following function and their default values already filled in `Main.py`:
   `C.Basic_Exp_Diversification()`
   Then execute the `Main.py` as follows:
   `python3 main.py`
   [Results] The results will be automatically saved in the "Figures" folder as Fig4-a-Latency.png, Fig4-a-Entropy.png, Fig5-Latency.png, and Fig5-Entropy.png.
2) **Experiment (E2)**[Table3] [15 min]: In this experiment, we investigate the trade-off between the value of τ and mixing delay to increase the entropy of output messages, as claimed in **C2**. There is an optimum value of tau and mixing delay, which was mentioned earlier and can be achieved with this experiment, around $\tau = 0.7$.
   [Preparation and Execution] Uncomment the line with the following function and the corresponding default values in `Main.py` and execute `Main.py`:
   `C.Table3()`
   [Results] The results will be automatically saved in the "Tables" folder as Maximum-Mixing-delay.pdf and will also be shown in the command window.
3) **Experiment (E3)**[Figure 9] [Duration: 30 min]
   This experiment explores different routing approaches while considering strong adversarial settings. The results from this experiment can be used to verify the two trends as described in **C3**.
   [Preparation and Execution] Uncomment the line with the following function and the corresponding default values in `Main.py` and execute `Main.py`:
   `C.FCP_Cnodes()`
   [Results] The results will be automatically saved in the "Figures" folder as Fig9-a.png, Fig9-b.png, and Fig9-c.png.
4) **Experiment (E4)**[Table 4] [Duration: 20 min]
   As mentioned in **C4**, this experiment aims to derive the results presented in Table 4 of the paper, demonstrating that using LARMix reduces latency while increasing the entropy of output messages compared to two layers mixnet.
   [Preparation and Execution] Uncomment the line with the following function and the corresponding default values in `Main.py` and execute `Main.py`:
   `C.Two_Layers_VS_LARMIX()`
   [Results] The results will be automatically saved in the "Tables" folder as Loopix-LARMix.pdf.

Note that there were two experiments that were not considered in the artifact evaluation as these were performed and added after undergoing a minor revision. These experiments are described along with their results in Section. VIII. These experiments are already part of the code provided on Github and can be easily replicated (using `Main.py`).

[9]Note that all other results in the main paper can also be reproduced. However, here we focus on the major claims and experiments.