OptiMix: Scalable and Distributed Approaches for Latency Optimization in Modern Mixnets

Mahdi Rahimi
COSIC, KU Leuven
mahdi.rahimi@esat.kuleuven.be

***Abstract*—Mixnets provide network-level anonymity, traded off with increased communication latency, which consequently limits their applicability to only latency-tolerant applications, shrinking the anonymity set to clients engaged in such use cases. Addressing this issue requires optimizing latency, as recently explored in LARMix (NDSS'24) and LAMP (NDSS'25) through node arrangement and strategic routing. However, these approaches are tailored to specific mixnet designs, rely on simplified models and trust assumptions, or suffer from limited practical efficiency.**

In contrast, OptiMix bridges these gaps by introducing a general low-latency mixnet model adaptable to all well-established designs. To this end, we first propose an efficient distributed protocol for arranging nodes in mixnets that achieves low-latency properties while maintaining unbiasability against adversaries. Second, we introduce novel strategic routing schemes that optimize communication latency. Third, we design a load-balancing algorithm that evenly distributes traffic without undermining the latency-optimized characteristics of the routing strategies. Fourth, we conduct extensive evaluations using data from the deployed Nym mixnet, demonstrating substantial latency reductions with minimal anonymity loss across various mixnet designs—achieving up to $4\times$ performance gains over state-of-the-art solutions. Finally, considering that latency reduction incurs either anonymity degradation or increased bandwidth overhead—as stated by the anonymity trilemma—we propose a cover-routing mechanism that enables clients to benefit from low-latency mixnets without compromising anonymity, at the modest cost of generating additional cover traffic.

I. Introduction

Mix networks (mixnets) [8], [15], [7], [42], [25], [41], [21], [13] are advanced overlay networks that provide anonymity for clients by forwarding their traffic through a sequence of intermediary nodes, known as *mixnodes* [8]. These mixnodes perform mixing operations that alter the patterns of received traffic to conceal the relayed path from a *Global Passive Adversary* (GPA) capable of observing all network links. In this setting, as long as at least one mixnode on the message path behaves honestly, a GPA is effectively prevented from deanonymizing the client's communication [12].

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.232680
www.ndss-symposium.org

Mixnets, on the other hand, are deployed in various configurations, with two widely adopted topologies being the *cascade* and *stratified* designs. In the cascade topology, mixnodes are organized into predefined chains (cascades), each consisting of h mixnodes. Clients select one of these cascades for message delivery [42], [7], [41], [21]. In contrast, the stratified topology arranges mixnodes into W layers (or sets), where clients form a path by selecting one mixnode from each layer [25], [13]. To generalize across both designs, OptiMix adopts a mixnet model based on the *butterfly* topology [9], as illustrated in Fig. 1. The butterfly structure consists of W wings, each with depth d (i.e., the number of chains), where each chain contains h intermediary mixnodes (hops), resulting in a total of $N = W \cdot d \cdot h$ mixnodes. Notably, when $W = 1$, the structure reduces to a cascade mixnet, and when $h = 1$, it becomes stratified, with each wing corresponding to a layer. This generality allows our framework to remain topology-agnostic while capturing the essential characteristics of both configurations. Similarly, the mixing process performed by mixnodes can be implemented in various ways. However, the most widely adopted scheme (which we also consider in OptiMix) is *stop-and-go mixing* [18], as deployed in the Nym mixnet. In this scheme, each message is delayed independently before being forwarded, with the delay drawn from an exponential distribution, $\mathsf{Exp}(\lambda)$.

Problem Statement. Mixnets, regardless of their structure or internal mixing processes, provide anonymity at the cost of high end-to-end communication latency. While such latency is tolerable for applications like email or crypto-wallets, it becomes a critical barrier for latency-sensitive use cases such as instant messaging or web browsing. As a result, clients may be discouraged from adopting mixnets in these scenarios, inadvertently shrinking the anonymity set that mixnets can offer. To explore the sources of high latency, note that the overall delay experienced by a message in a mixnet arises from three main components: (1) *mixing delay*, denoted by $\ell_{\texttt{Mix}}$, which is imposed by the mixing operations, delaying messages for reordering or shuffling purposes, performed by all mixnodes along a message route; (2) *propagation delay*, denoted by $\ell_{\texttt{P}}$, which results from network link latency when messages are forwarded between consecutive mixnodes; (3) δ_0, which accounts for additional delays due to cryptographic

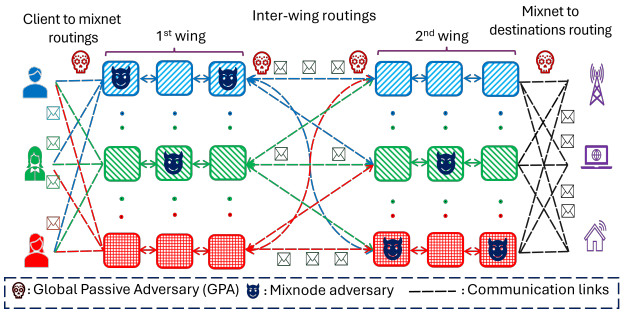


Fig. 1: A butterfly mixnet designed with OptiMix, where mixnodes with the same color or pattern are in close proximity. The mixnet has $W = 2$ wings, each featuring d chains, with each chain consisting of $h = 3$ mixnodes.

operations and queuing under network congestion. Since these can typically be minimized through efficient cryptographic primitives and proper network provisioning, we treat δ_0 as negligible in our analysis. Thus, the total end-to-end latency can be approximately modeled as: $\ell_{\texttt{e2e}} = \ell_{\texttt{Mix}} + \ell_{\texttt{P}}$.

In such cases, reducing overall latency $\ell_{\texttt{e2e}}$ requires minimizing either $\ell_{\texttt{Mix}}$ or $\ell_{\texttt{P}}$. For example, the former can be reduced by adjusting parameters of the mixing process, such as the exponential delay rate λ. However, this directly degrades anonymity by weakening mixing effectiveness and unlinkability [11]. In contrast, reducing $\ell_{\texttt{P}}$, which is primarily influenced by route selection, has a less direct impact on anonymity. Thus, optimizing $\ell_{\texttt{P}}$ offers a promising direction for improving latency without fundamentally compromising anonymity guarantees.

Related Work. While we aim to improve latency in mixnets, optimization efforts for various purposes in anonymous communication have been previously studied in the context of Tor [14], one of the most well-known anonymity networks. Specifically, numerous works [1], [38], [23], [2], [16], [43], [4], [3], [36], [17] have explored improving Tor's security and, in some cases, its performance. Among these, one of the most notable solutions is CLAPS [36], which proposed using linear programming to compute constrained routing strategies that optimize objective functions such as enhancing security or minimizing latency.

While these approaches are effective in the context of Tor, they do not readily translate to mixnets for two fundamental reasons: (1) Tor assumes a weaker adversarial model involving local or partial adversaries, whereas mixnets are designed to resist GPAs capable of observing all network links. Consequently, most strategies in Tor focus on avoiding paths entirely under adversarial control—an approach that is ineffective under the GPA model assumed in mixnets. (2) In Tor, a route is established once per session and reused for multiple message transmissions. In contrast, mixnets generate a fresh route for each message, requiring per-message routing strategies.

Due to these differences, mixnets have motivated the development of new solutions specific to their architecture. For this purpose, two main prior works [34], [35] have been proposed. Specifically, LARMix [34] is the first to explicitly address the high-latency drawback of mixnets for stratified topologies. It introduced a clustering and diversification algorithm to ensure that geographically distributed nodes are hosted at each layer of the mixnet. Additionally, LARMix proposed a routing formula that allows messages to traverse low-latency paths, along with a load-balancing algorithm to evenly distribute traffic across mixnodes after strategic routing is applied. Despite its novelty, LARMix lacks generality, as it only supports stratified mixnets. Moreover, it suffers from inefficiency—requiring up to $\mathcal{O}(N^5)$ computational complexity (as reported in [35])—which becomes prohibitive as $N \gg 100$. LARMix also assumes a centralized authority to coordinate node selection and path computation, introducing trust assumptions that weaken its privacy guarantees.[1]

Addressing some of these limitations, LAMP [35] proposed a new family of low-latency routing strategies. However, like LARMix, it remains restricted to stratified mixnet topologies. Unlike LARMix, LAMP removes diversification and balancing mechanisms, instead restricting clients to select nodes from a fixed subset (e.g., based on geographic circles or regions around clients). While this design reduces both latency and computational cost, it introduces two key limitations: (1) LAMP is not designed to ensure balanced traffic distribution across mixnodes, which leads to congestion in popular regions, causing message drops and ultimately increasing latency. (2) LAMP limits route selection to a narrow subset of nodes, excluding all others, which significantly reduces anonymity.

Threat Model and Design Goals. In contrast to prior works, OptiMix aims to reduce communication latency in mixnets through a general construction (Fig.1) applicable to both stratified and cascade topologies, while preventing node exclusion during message routing (unlike LAMP), ensuring even load distribution across mixnodes, and relying on minimal computational overhead and trust assumptions—all while preserving strong anonymity.

Additionally, anonymity must be rigorously evaluated under two adversarial models: (1) A *GPA* capable of observing all communication links but not the internal operations of mixnodes; consequently, the GPA can probabilistically correlate the inputs and outputs of each mixnode, thereby probabilistically linking messages entering the mixnet to those exiting. (2) A *mixnode adversary* capable of compromising a subset of mixnodes and gaining access to exact input-output mappings within those nodes. If all nodes along a message path are compromised, full deanonymization can occur.

Under these assumptions, the goals of OptiMix are specified as follows: (1) To construct mixnet arrangements where

[1]Two follow-up systems, CLAM [27] and LARMix++ [28], extend LARMix to support client-to-mixnet routing and Free Route topologies, but both inherit the same computational inefficiencies. Additionally, other works [32], [29] explore different scenarios of anonymity or latency optimization, which fall outside the scope of this paper. In contrast, [30] introduces novel methods based on differential privacy to optimize mixnet routing for general purposes; however, it incurs high computational costs and relies on non-decentralized trust assumptions.

mixnodes are placed to minimize propagation delay ℓ_{P}, while maintaining unbiasability against adversaries. (2) To introduce strategic routing schemes that allow clients to tune the trade-off between latency and anonymity via a routing parameter, while selecting the nodes along their message paths. (3) To ensure even load distribution across all mixnodes, preventing traffic bottlenecks that could otherwise increase latency and leak routing information. (4) To introduce a cover-routing mechanism that shifts part of the anonymity–latency trade-off to client-side computation, enabling low-latency performance without degrading privacy guarantees. This design is grounded in the anonymity trilemma [11], which states that latency reduction is inherently coupled with either anonymity degradation or increased bandwidth overhead.

Contributions. To achieve our design goals in OptiMix, we make the following key contributions: **First**, we introduce a novel and efficient mixnode arrangement strategy with minimal trust assumptions, deployed in a distributed manner, called the *Latency-Optimized Node Arrangement* (LONA). LONA arranges mixnodes to minimize latency between nodes within the same chain, while assigning the initial node in each chain using uniformly random values, leading to geographically diverse starting nodes across chains. This design makes the arrangement particularly conducive to latency reduction in both cascade and stratified topologies. Importantly, LONA preserves the property of *unbiasability*, ensuring that a mixnode adversary controlling up to half of the network cannot, prior to protocol execution, bias the placement of any mixnode. To the best of our knowledge, this is the first work to formally define and prove such a property (see Appendix E in the full version [33]). Additionally, LONA is computationally efficient, requiring only $\mathcal{O}(N^2)$ operations during its offline phase, making it practical even for large-scale mixnets.[2]

Second, building upon the LONA arrangement, we design latency-aware routing mechanisms for both client-to-entry-wing and inter-wing routing. These strategies offer tunable trade-offs between latency and anonymity by assigning node selection probabilities based on measured latencies between consecutive nodes. Unlike LAMP, our approach considers *all* available nodes when selecting each hop to preserve higher anonymity, while reducing the likelihood of selecting distant nodes via a tunable parameter $\tau \in [0, 1]$. When $\tau = 0$, routing is fully biased toward the lowest-latency paths; conversely, when $\tau = 1$, routing becomes fully randomized, maximizing anonymity. This design gives clients the flexibility to determine how much anonymity they are willing to trade off for improved latency.

Third, since latency-aware routing may introduce the risk of overloading certain mixnodes or chains, we propose the *Load Balancing Algorithm* (LBA). Assuming that clients generate traffic at similar rates, LBA adjusts node selection such that each mixnode handles approximately the same number of messages, while preserving the characteristics of the applied strategic routing. LBA operates with a computational complexity of $\mathcal{O}(N^2)$, significantly outperforming the algorithm used in LARMix (which incurs over $\mathcal{O}(N^4)$), and achieving improved trade-offs between latency and anonymity.

Finally, we introduce the *Cover Routing Generation* (CRG) mechanism, which provides clients with sufficient computational resources the option to trade off reduced latency not for anonymity, but for additional message generation and transmission. In this approach, clients generate dummy messages that are routed through the mixnet and eventually return to them, with the goal of deceiving adversaries by neutralizing the statistical bias introduced by latency-aware routing. As a result, the overall path selection distribution appears highly uniform. We further show that CRG operates as a probabilistic mechanism that preserves the client's original routing preferences, offering (ϵ, δ)-differential privacy guarantees. Under more precise calibrations, it further enhances the anonymity provided by mixnets, as quantified by Shannon entropy [37].

Moreover, we provide a thorough evaluation of OptiMix, using both an analytical approach—which allows us to gauge the effect of node arrangement, strategic routing, and LBA on reducing ℓ_{P} and mixnet anonymity (while isolating the impact of the mixing process)—as well as a simulation-based methodology that captures the combined effect of mixnode arrangement, strategic routing, and the mixing process on both latency (ℓ_{e2e}) and total anonymity. For evaluation, we consider latency datasets extracted from the deployed Nym mixnet, as well as the RIPE Atlas dataset [40], where we measure average latency in seconds and assess anonymity using Shannon entropy (measured in bits).[3]

Our findings reveal that LONA reduces latency by at least 58% on both Nym and RIPE datasets compared to vanilla mixnet arrangements under a cascade topology, with negligible cost to anonymity against adversaries. When low-latency routing is applied, our proposed strategies, combined with LONA, achieve latency reductions of up to 88% across both cascade and stratified topologies compared to vanilla configurations, with only a reasonable anonymity trade-off. Furthermore, compared to the state-of-the-art LAMP (under stratified topologies), we demonstrate that OptiMix achieves overall $2\times$ better anonymity/latency trade-offs, while also offering considerably improved anonymity. When load balancing is required, our analysis shows that LBA effectively distributes traffic equally across mixnodes while preserving the low-latency characteristics of our routing strategies. Specifically, when combined with LONA and our routing methods, LBA achieves a $6\times$ better anonymity/latency trade-off compared to vanilla settings, and under stratified topologies, a $4\times$ improvement over LARMix.

We further analyze the impact of mixnode adversaries that corrupt a subset of nodes and collaborate with a GPA to deanonymize client connections. To evaluate this, we define two adversarial strategies: (1) *Naive corruption*, where nodes

[2]Mixnets like Nym operate in epochs; during each epoch, while mixnodes process traffic, LONA can run in parallel to prepare the node arrangement for the next epoch.

[3]All code and resources for evaluation are available at https://github.com/OptiMixnet/OptiMix, and all experiments can be reproduced by following the instructions provided in Appendix E.

are compromised at random, and (2) *Greedy corruption*, where adversaries target geographically close nodes. Under these scenarios, we evaluate the adversary's ability to compromise full communication paths. Results show a moderate increase in adversarial advantage compared to traditional mixnet setups. Lastly, our analysis of CRG demonstrates that introducing a moderate rate of cover routing traffic can effectively mitigate the advantage of both GPA and mixnode adversaries, while still achieving the same level of latency reduction—at the cost of generating a moderate amount of additional traffic per client.

II. Methodology

This section introduces the OptiMix methodology, beginning with the LONA protocol for assigning mixnodes to the mixnet. We then describe the strategic routing mechanisms used to select nodes or chains to form a message's path. Finally, we present the LBA algorithm, which ensures an even distribution of traffic across mixnodes.

A. Latency-Optimized Node Arrangement

Traditionally, mixnets have been constructed by randomly assigning mixnodes [8], [15], [7], [42], [25], [41], [21], [13]. While this random assignment simplifies deployment, it often results in configurations where mixnodes along a message path are geographically distant from one another, leading to high end-to-end latency (ℓ_{e2e}). To address this limitation, we introduce *LONA*, a novel node arrangement protocol designed to organize mixnets such that mixnodes within each chain are latency-proximate. Additionally, LONA assigns the initial node of each chain based on uniformly random values, ensuring that the entry nodes of chains are geographically diverse. Together, these two design features make the structure well-suited for latency optimization in both cascade and stratified topologies. Specifically, in cascade topologies, intra-chain proximity directly reduces propagation delay across hops, while in stratified topologies, the presence of geographically diverse nodes within each wing (i.e., layer) increases the likelihood of selecting low-latency links between consecutive layers, thereby reducing overall propagation delay. As an example, after applying LONA (as illustrated in Fig. 1), mixnodes with low inter-node latency (e.g., blue, green, and red) are grouped into the same chain, while the initial nodes of chains across wings are spatially diverse. Moreover, LONA ensures that the resulting mixnet arrangement remains *unbiasable*, a property achieved by randomly selecting the first node of each chain, preventing mixnode adversaries from gaining advantage through prior knowledge of the arrangement.

LONA Initialization. Consider a set of mixnodes $\mathcal{N}$, where $|\mathcal{N}| = N = W \cdot d \cdot h$, available to construct a mixnet. To initialize the process, each mixnode M_i generates a secret key $sk_i \in \mathbb{Z}_q$ and a corresponding public key $pk_i = g_1^{sk_i}$.[4] We assume the existence of a broadcast channel, as in [22], through which all mixnodes publish their public keys. These keys also serve as unique identifiers for the nodes. Thus, we define the set of mixnodes as: $\mathcal{N} = \{M_{pk_1}, M_{pk_2}, \ldots, M_{pk_N}\}$. For simplicity, since pk_i values are large, we define a fixed but arbitrary ordering of public keys as the set $\mathcal{S}_{ID}$, which is used throughout the protocol to refer to mixnodes by their index in $\mathcal{S}_{ID}$. Next, all mixnodes collaboratively execute the VerLoc mechanism [20], which enables peer-to-peer latency estimation among mixnodes with high accuracy and robustness[5]—tolerating up to 30% corrupted nodes (see Appendix B for more details). After running VerLoc, each mixnode M_{pk_i} publishes to the broadcast channel a sorted list $\mathcal{S}_{L_i}$, containing its measured latencies to all other mixnodes in ascending order.

Furthermore, to ensure unbiasability in LONA, we require uniformly distributed randomness during the assignment of the first node in each chain. We achieve this using methods such as [6]. In this setting, mixnodes collectively generate a uniform and verifiable randomness pool consisting of t^2 values, which is robust against up to $f = \frac{N-t}{2}$ corrupted mixnodes. For convenience, we abstract this randomness pool as the set $\mathcal{S}_R = \{r_i \mid 1 \leq i \leq t^2\}$, and refer the interested reader to Appendix B for further details. In LONA, we set $t = \lceil \sqrt{d \cdot W + h - 1} \rceil$. Among the elements of $\mathcal{S}_R$, the first $d \cdot W$ values are used to determine the initial mixnode of each chain. The next $h - 1$ values, indexed as $d \cdot W + 1 \leq i \leq d \cdot W + h - 1$, guide the prioritization of chains during the assignment process, forming the set $\mathcal{S}_P = \{r_i \in \mathcal{S}_R \mid d \cdot W + 1 \leq i \leq d \cdot W + h - 1\}$.

LONA Node Assignment. The assignment phase in LONA is guided by two primary goals: (1) ensuring that the assignment process is unbiasable against mixnode adversaries, and (2) promoting latency-aware proximity among nodes within each chain. With these goals in place, and with the randomness ($\mathcal{S}_R$ and $\mathcal{S}_P$), latency measurements ($\mathcal{S}_{L_i}$), and mixnode identity set ($\mathcal{S}_{ID}$) established, we proceed to detail the node assignment procedure in LONA as follows. Let $\mathcal{S}_{C_i}$ denote the set of mixnodes in chain i, for $1 \leq i \leq d \cdot W$ (as in a mixnet with W wings, we have $W \cdot d$ chains). Initially, each $\mathcal{S}_{C_i}$ is empty. Furthermore, we define $\mathcal{C}_{ID} = \{1, 2, \ldots, d \cdot W\}$, where each element l of $\mathcal{C}_{ID}$ corresponds to chain l, representing the ($l \mod d$)-th chain positioned in the $\lceil \frac{l}{d} \rceil$-th wing. Additionally, we define a function $F_{ID}(\cdot)$ that receives an index referring to an element in $\mathcal{S}_{ID}$ or $\mathcal{C}_{ID}$ and returns the identity of the mixnode or chain, respectively.[6]

Following this, LONA progresses by utilizing the $d \cdot W$ randomness values r_i from the set $\mathcal{S}_R - \mathcal{S}_P$. Given $\mathcal{H}(\cdot)$ as a random oracle, to assign the first mixnode to the mixnet, LONA considers the first element of $\mathcal{S}_R - \mathcal{S}_P$, r_1, and computes:

$$\mathtt{Id}_1 = F_{ID}\left(\left[\mathcal{H}(r_1 \| g_1) \mod |\mathcal{S}_{ID}|\right], \mathcal{S}_{ID}\right),$$

$$\mathtt{Ch}_1 = F_{ID}\left(\left[\mathcal{H}(r_1 \| g_2) \mod |\mathcal{C}_{ID}|\right], \mathcal{C}_{ID}\right).$$

[4] Throughout the paper, we consider g_1 and g_2 as generators of a group $\mathbb{G}_q$.

[5] This mechanism is actively deployed within the Nym network (https://nymtech.net).

[6] More precisely, F_{ID} returns the public key of the selected mixnode from $\mathcal{S}_{ID}$, and for chains, it outputs their respective identifier based on $\mathcal{C}_{ID}$.

Here, $\texttt{Id}_1$ maps r_1 to a public key in $\mathcal{S}_{ID}$, while $\texttt{Ch}_1$ maps another random value to a chain in $\mathcal{C}_{ID}$, thereby selecting a mixnode and a chain uniformly at random, thus satisfying the first goal. Following this, the mixnode $M_{\texttt{Id}_1}$ is added to the selected chain $\texttt{Ch}_1$ (i.e., to the set $\mathcal{S}_{C_{\texttt{Ch}_1}}$). To prevent reassigning a mixnode or selecting a chain more than once, $\texttt{Ch}_1$ is removed from $\mathcal{C}_{ID}$, $\texttt{Id}_1$ is removed from $\mathcal{S}_{ID}$, and $M_{\texttt{Id}_1}$ is removed from the set of available nodes $\mathcal{N}$.

To assign the remaining $d \cdot W - 1$ initial mixnodes to chains, the same process is repeated, where $\texttt{Id}_i$ and $\texttt{Ch}_i$ are computed as follows:

$$\texttt{Id}_i = F_{ID}\left(\left[\mathcal{H}(r_i \| g_1) \mod |\mathcal{S}_{ID}|\right], \mathcal{S}_{ID}\right), \quad 2 \le i \le d \cdot W,$$

$$\texttt{Ch}_i = F_{ID}\left(\left[\mathcal{H}(r_i \| g_2) \mod |\mathcal{C}_{ID}|\right], \mathcal{C}_{ID}\right), \quad 2 \le i \le d \cdot W.$$

Following this, the mixnode corresponding to $\texttt{Id}_i$ in $\mathcal{S}_{ID}$ is assigned to the chain identified by $\texttt{Ch}_i$ in $\mathcal{C}_{ID}$. After each assignment, the set $\mathcal{S}_{C_{\texttt{Ch}_i}}$ is updated to include $M_{\texttt{Id}_i}$, and $\mathcal{C}_{ID}$, $\mathcal{S}_{ID}$, and $\mathcal{N}$ are updated as described previously.

After assigning the first mixnodes to each chain, LONA proceeds to assign the subsequent $h-1$ mixnodes for each chain. For each k-th hop assignment ($2 \le k \le h$), where each chain receives its k-th mixnode, we first reset $\mathcal{C}_{ID} = \{1, 2, \ldots, d \cdot W\}$, consider the latency vectors $\mathcal{S}_{L_j}$ ($1 \le j \le N$), and use the $(k-1)$-th element of $\mathcal{S}_P$, namely $r_{d \cdot W + k - 1}$, for random prioritization in assigning the k-th hops to chains. The assignment of subsequent nodes, however, is guided by the second goal—proximity to previously selected mixnodes. The k-th hop assignment includes $d \cdot W$ rounds of mixnode assignments. In round i, the chain identifier is computed as:

$$\texttt{Ch}_i = F_{ID}\left(\left[\mathcal{H}(r_{d \cdot W + k - 1} \| {g_1}^i \| g_2) \mod |\mathcal{C}_{ID}|\right], \mathcal{C}_{ID}\right).$$

Let $M_{\texttt{Ch}_i^{k-1}}$ be the mixnode at position $k-1$ in chain $\texttt{Ch}_i$. To identify the next mixnode in this chain, the latency vector $\mathcal{S}_{L_{\texttt{Ch}_i^{k-1}}}$ is used to find the mixnode incurring the lowest latency relative to $M_{\texttt{Ch}_i^{k-1}}$ among all available nodes in $\mathcal{N}$. This node is computed as:

$$M_{\texttt{Id}_i} = \arg\min_{M_{\texttt{Id}_j}} \left(\mathcal{S}_{L_{\texttt{Ch}_i^{k-1}}}\right) \quad \texttt{s.t.} \;\; M_{\texttt{Id}_j} \in \mathcal{N}. \tag{1}$$

After assigning $M_{\texttt{Id}_i}$ to $\texttt{Ch}_i$, the sets $\mathcal{N}$, $\mathcal{S}_{ID}$, and $\mathcal{C}_{ID}$ are updated by removing the selected mixnode and chain, while $\mathcal{S}_{C_{\texttt{Ch}_i}}$ is updated by adding $M_{\texttt{Id}_i}$. If multiple mixnodes satisfy the $\arg\min$ condition, the node with the smallest identifier is selected. This iterative process is repeated for all chains ($1 \le i \le d \cdot W$) and subsequently for all remaining hops ($2 \le k \le h$). Upon completion, the union of $\mathcal{S}_{C_i}$ for $i = 1$ to $d \cdot W$ represents the final mixnet configuration.[7]

[7]Note that Eq. (1) assigns the 2nd to h-th mixnodes of each chain based on minimum latency. In highly uneven node distributions, this may cause all mixnodes of a chain to be located in one jurisdiction. To avoid this, one can modify Eq. (1) to $M_{\texttt{Id}_i} = \arg\min_{M_{\texttt{Id}_j}} \left|\mathcal{S}_{L_{\texttt{Ch}_i^{k-1}}} - \ell'\right| \;\; \texttt{s.t.} \;\; M_{\texttt{Id}_j} \in \mathcal{N}$, where ℓ' is a small random latency offset, e.g., sampled uniformly from $[0, 50\text{ms}]$, to slightly randomize the assignment. However, as Nym nodes are globally distributed, we consider $\ell^* = 0$ in OptiMix.

LONA Security. A mixnet structure arranged using LONA is *unbiasable* against an adversary controlling a set of f mixnodes prior to the execution of LONA. As a result, adversaries cannot exploit the mixnet arrangement to deanonymize client connections. Unbiasability is formally defined in Appendix E of the full version [33], with the corresponding security proof provided in Theorem 4 of the same. Intuitively, unbiasability is achieved through the pool of uniformly random values that LONA uses to select the initial mixnode in each chain. This ensures that the positions of the first nodes are unbiasable. Subsequent mixnodes are assigned based on proximity to previously selected nodes, so their positions—being dependent on the first node—also remain unbiasable.

LONA Efficiency. To run LONA, each participant (mixnode) must first engage in the VerLoc latency measurement protocol, which requires $\mathcal{O}(N^2)$ operations, followed by sorting its latency vectors with $\mathcal{O}(N \log N)$ complexity. Participants must also generate a pool of randomness, which requires $\mathcal{O}(N^2)$ operations. During the node assignment phase, for each round where the k-th hop ($1 \le k \le h$) is assigned, $\mathcal{O}(d \cdot W)$ operations are required. Since there are h such rounds, the full assignment phase costs $\mathcal{O}(N)$ overall. Therefore, the total computational complexity of LONA is $\mathcal{O}(N^2)$. It is important to note that the mixnet structure does not change frequently (e.g., it may be updated once every few hours), enabling LONA to be executed in an offline phase.[8]

B. Latency-Optimized Routing

Traditionally, in mixnets, routes are constructed by selecting mixnodes uniformly at random. When this strategy is used in a butterfly mixnet (Fig. 1), it results in the selection of one chain from each wing uniformly at random. However, this can lead to routes where the selected chains (or nodes) are geographically distant from one another, significantly increasing inter-chain latency and thereby contributing to high propagation delay (ℓ_{P}). In this section, we aim to reduce such latency. When combined with the minimized intra-chain latency achieved through LONA, this enables low-latency mixnet messaging. To this end, suppose a node i at position p_0 wants to route to a node j at position p_1. For example, p_0 may represent the last node in a chain from the first wing, and p_1 a node in the set $\mathcal{S}_{p_1}$, representing all first nodes in the chains of the second wing. Traditionally, any $j \in \mathcal{S}_{p_1}$ is selected with equal probability: $\frac{1}{|\mathcal{S}_{p_1}|}$. In contrast, strategic routing biases the selection toward nodes with lower link latency. Specifically, the probability of selecting a node j from $\mathcal{S}_{p_1}$, denoted f_{ij}, is made proportional to the inverse of the imposed latency ℓ_{ij}, i.e., $f_{ij} \propto \frac{1}{\ell_{ij}}$. This means that lower-latency links are more likely to be chosen.

While strategic routing improves performance, it may also make message routes more predictable to a GPA, potentially compromising anonymity. Moreover, it may concentrate traffic through nodes in specific jurisdictions, raising privacy concerns if those nodes are vulnerable to coercion or compromise.

[8]The full LONA protocol is described in Fig. 11 and Fig. 12 in Appendix A.

To mitigate such risks, strategic routing must preserve a degree of randomness. We quantify this randomness using a heuristic metric called the *Low-Latency Tradeoff* (LLT) metric, defined as: $\texttt{LLT} = \frac{\max(f_{ij})}{\min(f_{ij})}$. This metric captures how much more likely the lowest-latency node is to be selected compared to the node with the highest latency. When routing is uniform, $\texttt{LLT} = 1$. Conversely, when routing is fully biased toward the lowest-latency node, $\texttt{LLT} \to \infty$. An optimal latency-aware routing strategy should therefore aim to bound LLT.

Simple-Strategic Routing (SSR). As a first step toward low-latency routing, we define a simple approach for strategic routing shown in Eq. (2), tying f_{ij} to the inverse of latency raised to the power of $(1-\tau)$, where $0 \leq \tau \leq 1$ moderates the level of bias. When $\tau = 0$, the routing is fully biased toward the inverse of latency, while as τ increases, the routing becomes more uniform—becoming uniformly random over $\mathcal{S}_{p_1}$ when $\tau = 1$. SSR has a bounded LLT metric given by: $\texttt{LLT}_{\text{SSR}} = \left(\frac{\texttt{Max}(\ell_{ij})}{\texttt{Min}(\ell_{ij})}\right)^{1-\tau}$, which reaches a maximum of $\frac{\texttt{Max}(\ell_{ij})}{\texttt{Min}(\ell_{ij})}$ when $\tau = 0$.

$$f_{ij} = \frac{\left(\frac{1}{\ell_{ij}}\right)^{1-\tau}}{\sum_{n\in\mathcal{S}_{p_1}}\left(\frac{1}{\ell_{in}}\right)^{1-\tau}}. \tag{2}$$

LARMix Routing (LAR). The $\texttt{LLT}_{\text{SSR}}$ is restricted to $1 \leq \texttt{LLT}_{\text{SSR}} \leq \frac{\texttt{Max}(\ell_{ij})}{\texttt{Min}(\ell_{ij})}$, which limits SSR's ability to further bias routing toward low-latency paths. To increase bias, we adopt a formula first introduced by LARMix [34] and described in Eq. (3), extending SSR by setting $f_{ij} \propto \left(\frac{1}{e}\right)^{\mathcal{I}_{ij}\cdot\frac{1-\tau}{\tau}}$. Here, $\mathcal{I}_{ij}$ ranks the proximity of node j to node i, starting at 0 for the closest and up to $|\mathcal{S}_{p_1}| - 1$ for the farthest. When $\tau = 0$, the fastest link is always selected, and $\tau = 1$ yields uniform routing. LAR's corresponding LLT is: $\texttt{LLT}_{\text{LAR}} = \left(e^{(|\mathcal{S}_{p_1}|-1)\cdot\frac{1-\tau}{\tau}}\right)\cdot\texttt{LLT}_{\text{SSR}}$, which shows that, compared to SSR, LAR can fully bias routing toward the fastest link, as $\texttt{LLT}_{\text{LAR}} \to \infty$ when $\tau \to 0$.

$$f_{ij} = \frac{\left(\frac{1}{e}\right)^{\mathcal{I}_{ij}\cdot\frac{1-\tau}{\tau}}\cdot\left(\frac{1}{\ell_{ij}}\right)^{1-\tau}}{\sum_{n\in\mathcal{S}_{p_1}}\left(\frac{1}{e}\right)^{\mathcal{I}_{in}\cdot\frac{1-\tau}{\tau}}\cdot\left(\frac{1}{\ell_{in}}\right)^{1-\tau}}. \tag{3}$$

Geometric Weighted Routing (GWR). While LAR improves the flexibility of SSR, it introduces extreme bias toward the closest node, especially for small τ, potentially reducing anonymity. To address this, we propose GWR, defined in Eq. (4). GWR moderates the preference for the closest node using a geometric decay factor: $f_{ij} \propto \left(\frac{1}{2}\right)^{\mathcal{I}_{ij}(1-\tau)^2}$. This leads to $\texttt{LLT}_{\text{GWR}} = \left(2^{(|\mathcal{S}_{p_1}|-1)\cdot(1-\tau^2)}\right)\cdot\texttt{LLT}_{\text{SSR}}$, showing that GWR offers greater flexibility than SSR while bounding the maximum bias. Specially, when $\tau = 0$, the upper bound becomes: $\texttt{LLT}^{\max}_{\text{GWR}} = 2^{|\mathcal{S}_{p_1}|-1}\cdot\frac{\texttt{Max}(\ell_{ij})}{\texttt{Min}(\ell_{ij})}$. At $\tau = 1$, GWR behaves identically to SSR. Overall, GWR reduces the anonymity loss observed in LAR while maintaining greater flexibility than SSR.

$$f_{ij} = \frac{\left(\frac{1}{2}\right)^{\mathcal{I}_{ij}(1-\tau)^2}\cdot\left(\frac{1}{\ell_{ij}}\right)^{1-\tau}}{\sum_{n\in\mathcal{S}_{p_1}}\left(\frac{1}{2}\right)^{\mathcal{I}_{in}(1-\tau)^2}\cdot\left(\frac{1}{\ell_{in}}\right)^{1-\tau}}. \tag{4}$$

Group Prioritizing Routing (GPR). Differing from previous methods, we introduce GPR, which partitions the set $\mathcal{S}_{p_1}$ into N_g groups based on latency. The latency range Δ is computed as: $\Delta = \frac{\texttt{Max}-\texttt{Min}}{N_g}$, where Max and Min are the maximum and minimum latencies imposed by nodes in $\mathcal{S}_{p_1}$. Each group $\mathcal{G}_j$ consists of nodes whose link latencies fall within the interval $[\texttt{Min} + \Delta\cdot(j-1), \texttt{Min} + \Delta\cdot j]$. Thus, $\mathcal{G}_1$ contains the lowest-latency nodes, while $\mathcal{G}_{N_g}$ contains the highest-latency nodes. The selection probability for a node in group $\mathcal{G}_j$ is given by Eq. (5), where $0 < \alpha < 1$ is determined by solving: $\sum_{j=1}^{N_g}|\mathcal{G}_j|\alpha^j = 1$, which can be efficiently solved using standard numerical methods such as Newton-Raphson or bisection.

The corresponding GPR's LLT metric is: $\texttt{LLT}_{\text{GPR}} = \alpha^{(1-N_g)(1-\tau)}$. When $N_g = 1$ or $\tau = 1$, we get $\texttt{LLT}_{\text{GPR}} = 1$, indicating uniform routing. As τ decreases, $\texttt{LLT}_{\text{GPR}}$ increases up to a maximum of α^{1-N_g}, which remains bounded since α is non-negligible and $N_g \leq |\mathcal{S}_{p_1}|$. Hence, GPR achieves greater flexibility than SSR and stronger anonymity guarantees than LAR due to its tunable parameters N_g and τ.

$$f_{i\mathcal{G}_j} = \frac{\alpha^{j\cdot(1-\tau)}}{\sum_{\mathcal{G}_n\in\mathcal{S}_{p_1}}|\mathcal{G}_n|\alpha^{n\cdot(1-\tau)}}. \tag{5}$$

C. Balancing Load Distributions

After applying strategic routing for node selection, some mixnodes (or chains) may receive disproportionately higher traffic loads compared to what they would under a uniform routing policy. If left unaddressed, this imbalance can cause queuing delays and offer adversaries an opportunity to exploit the system—e.g., by deploying high-capacity mixnodes to attract and deanonymize traffic. To mitigate this issue, we propose an efficient *Load Balancing Algorithm (LBA)* (Fig. 10), which adjusts the load received by mixnodes at position p_1, without harming the low-latency characteristics of the routing.To formalize this, we define Γ as the routing matrix from position p_0 to position p_1, where $\Gamma = [f_{ij}]$. Thus, each row of Γ represents the routing distribution from a node $i \in \mathcal{S}_{p_0}$ over $\mathcal{S}_{p_1}$. Note that the summation of each row is 1 (probability distribution), but the summation of column j, representing the load received by node j at position p_1, is not necessarily 1. Columns with sums greater than 1 indicate overloaded nodes, while sums less than 1 identify underloaded nodes. Ideally, all columns should sum to 1 for a perfectly balanced distribution.

To achieve load balancing, the LBA algorithm begins by scaling each imbalanced column of Γ by the inverse of its total sum, ensuring all columns sum to 1. Next, all rows of the updated Γ are scaled by the inverse of their total sums to represent valid probability distributions. While this adjustment may cause some columns to become imbalanced

again, the severity of the imbalance decreases. This process is repeated iteratively until both rows and columns sum to 1, resulting in a balanced routing matrix Γ_b (see a toy example in Appendix C). Theorem 1 in Appendix C shows that LBA converges in at most $\mathcal{O}\left(\log_{10}\left(\frac{1}{\epsilon_e}\right)\right)$ iterations, where ϵ_e denotes the required balancing precision. The total computational complexity of LBA is $\mathcal{O}\left(N^2 \log_{10}\left(\frac{1}{\epsilon_e}\right)\right)$. For instance, if the desired precision is at most 10^{-5}, we have $\log_{10}\left(\frac{1}{\epsilon_e}\right) = 5$; this leads to an overall computational complexity of $\mathcal{O}(N^2)$. Moreover, compared to the state-of-the-art balancing algorithm in LARMix [34], which requires $\mathcal{O}(kN^4)$ with $k = \mathcal{O}(N)$, LBA is significantly more efficient. Our experimental evaluation (Appendix C) further demonstrates that LBA is not only faster but also better at preserving the latency-aware characteristics of strategic routing.

For practical deployment of LBA, in client-to-mixnet routing, each client discloses its preferred routing distribution,[9] while for inter-wing routing, the matrix Γ can be constructed using publicly available latency data and routing strategies published on the broadcast channel. Using this data, all mixnodes independently compute the balanced routing matrix Γ_b and broadcast their result. Since a majority of mixnodes are assumed to be honest, consensus on the correct Γ_b is guaranteed.

III. Evaluation

This section evaluates schemes proposed in OptiMix under a GPA assumption. To do so, we begin by defining evaluation metrics, describing the experimental setup, and finally presenting the evaluation results.

Evaluation Metrics. OptiMix employs two types of evaluation metrics: *analytical metrics* and *simulation metrics*. In the analytical approach, we quantify the effect of OptiMix on both propagation latency ($\ell_{\mathtt{P}}$) and anonymity, while excluding the impact of the mixing process. In this case, the average $\ell_{\mathtt{P}}$ is computed as the weighted average of latencies across all paths, where the weights correspond to the probabilities of path selection. Anonymity, on the other hand, is measured using the entropy of routing, denoted $\mathsf{H}(r)$, and calculated via Shannon entropy [37]. This metric, originally introduced in LARMix [34, page 7], captures the uncertainty that a GPA has in determining the exit mixnode of a message, given its entry mixnode to the mixnet.

Contrastingly, simulation metrics capture the combined effect of the protocols in OptiMix and the mixing process to quantify both total anonymity and end-to-end latency ($\ell_{\mathtt{e2e}}$). To this end, we deploy a discrete event simulator implemented using *Simpy* [26] in *Python*, designed to simulate a butterfly mixnet with clients generating messages that traverse all hops along their paths and experience both link latency and mixing delay imposed by each mixnode. In this case, $\ell_{\mathtt{e2e}}$ is computed by measuring the time each message takes from when it is forwarded by a client to when it exits the final mixnode. Anonymity, nonetheless, is measured by tagging a subset of messages prior to entering the mixnet, considering a GPA observing all communication exchanges. We then compute the probability of each outgoing message at the exit of the mixnet being one of the initially tagged messages, following approaches similar to [5], [24]. The entropy of the probability distribution over all outgoing messages for each tagged message, denoted by $\mathsf{H}(m)$, serves as the anonymity metric.[10]

Experimental Setup. In our evaluation, we consider three mixnet topologies: a butterfly topology with parameters ($W = 2$, $h = 3$, $d = 80$); a cascade topology ($W = 1$, $h = 3$, $d = 80$); and a stratified topology ($W = 3$, $h = 1$, $d = 80$). We set the average mixing delay per mixnode to follow an exponential distribution with a mean of 50ms, which aligns with the deployed Nym mixnet. During simulation, we generate an average of 20,000 messages per second entering the mixnet, based on a Poisson distribution. Furthermore, to collect peer-to-peer latency data, we employ the Verloc framework [20] through the Nym network. Specifically, we query each mixnode on port `1790` and incorporate the measured latencies into our evaluation to model inter-node delays across the mixnet. To ensure statistical reliability, each experiment is repeated over at least 400 independent mixnet instantiations, reducing the likelihood of anomalies or faulty results.

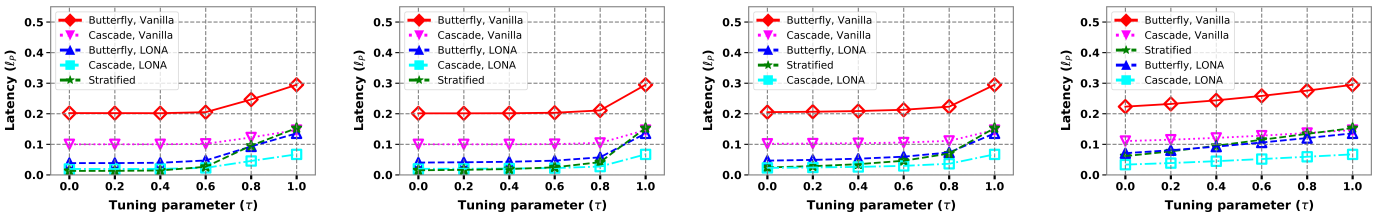
Average Propagation Latency. We begin presenting the evaluation results by assessing the average propagation latency $\ell_{\mathtt{P}}$ across the butterfly topology and its instantiations for cascade and stratified mixnets. Specifically, we compare both vanilla mixnet arrangements and LONA for the butterfly and cascade topologies. For the stratified topology, we consider only LONA, as both LONA and the vanilla configuration yield nearly identical results, assuming that node assignment in the vanilla setting is also based on uniformly random values. The results are shown in Fig. 2, where we evaluate $\ell_{\mathtt{P}}$ across all routing strategies as a function of the tuning parameter τ.

Fig. 2 reveals that across all strategies and mixnet structures, reducing the value of τ leads to decreased latency, as lower τ emphasizes the selection of low-latency links, thereby minimizing overall delay. More specifically, Fig. 2 demonstrates that for both butterfly and cascade topologies, constructing the mixnet using LONA, rather than the vanilla approach, significantly reduces average propagation latency across all values of τ and all routing strategies. Notably, when $\tau = 1$ (uniform routing) in the cascade setting, latency under vanilla reaches approximately 150 ms, while LONA reduces it to about 80 ms. In the butterfly topology, although latency increases overall due to the additional number of hops, LONA still brings it down to around 125 ms, a 58% reduction, highlighting its effectiveness in optimizing intra-chain latency.

Additionally, Fig. 2 suggests that LAR and GWR strategies follow similar trends, reducing latency to approximately

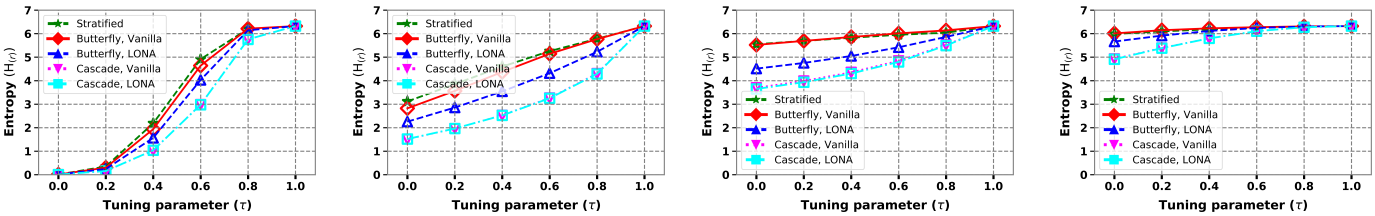
[9] This disclosure does not compromise client privacy, as a GPA can already infer routing preferences through traffic observation.

[10] We later extend this metric to incorporate the impact of a mixnode adversary. Note, however, that one could alternatively use methods such as [10], [31] for anonymity analysis. These approaches, however, rely on assumptions regarding the distribution of clients and their destinations.



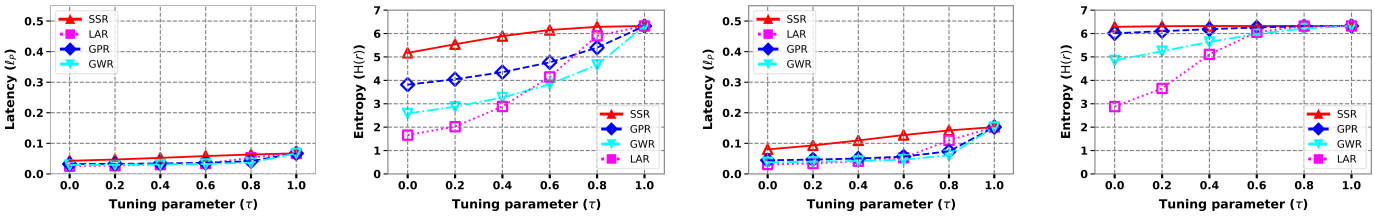
(a) LAR (b) GWR (c) GPR, $N_g = 20$ (d) SSR

Fig. 2: Average propagation latency ℓ_{P} (sec) under different mixnet arrangements.



(a) LAR (b) GWR (c) GPR, $N_g = 20$ (d) SSR

Fig. 3: Average entropy $\mathsf{H}(r)$ (bits) under different mixnet arrangements.



(a) Latency, LBA, cascade (b) Entropy, LBA, cascade (c) Latency, LBA, stratified (d) Entropy, LBA, stratified

Fig. 4: Latency and entropy after applying LBA algorithm for both cascade and stratified topologies.

15 ms (for cascade and stratified) and 40 ms (for butterfly) when $\tau = 0$ under LONA, representing an 88% reduction. Notably, latency stabilizes near the optimal level for these settings when $\tau \leq 0.6$, indicating that substantial latency improvements can still be achieved even with moderate levels of randomness. This flexibility enables routing paths to span different jurisdictions (enhancing privacy) while maintaining geographically short links to reduce delay. GWR, however, shows a more gradual increase in latency compared to LAR, which begins to increase sharply around $\tau = 0.6$. GPR with $N_g = 20$ exhibits similar behavior to both LAR and GWR, but with slightly higher latency. Under LONA and $\tau = 0$, GPR reduces latency to approximately 20 ms (cascade and stratified) and 45 ms (butterfly), with its latency curve increasing more slowly than that of GWR. In contrast, SSR provides the least latency reduction, showing a near-linear increase in latency as τ rises—achieving, at best, 40 ms (cascade/stratified) and 80 ms (butterfly) under LONA. This demonstrates that SSR falls short in latency optimization. Moreover, the stratified topology shows higher latency under SSR, almost equivalent to butterfly, while under other strategies and for $\tau \leq 0.6$, stratified behaves similarly to cascade. This highlights a further limitation of SSR in reducing latency.

Average Entropy (Anonymity). We further assess anonymity across OptiMix strategies using the entropy of routing, $\mathsf{H}(r)$, as shown in Fig. 3. As illustrated, increasing τ generally raises entropy across all settings, since higher τ introduces more randomness into routing, thereby increasing uncertainty. Specifically, in both cascade and butterfly settings, when LONA is applied, entropy slightly decreases across all routing strategies and values of τ compared to the vanilla approach. This is because LONA forms chains in such a way that some provide significantly low-latency for specific clients, causing routing strategies to become more skewed, thereby reducing entropy. Furthermore, butterfly topologies tend to yield higher entropy than cascades due to the increased number of hops along the message path. In contrast, stratified topologies, despite involving fewer hops, offer the highest entropy overall. This is because in stratified designs, each hop involves an independent selection from a full layer of mixnodes. In cascades and butterfly topologies, however, selecting the first node in a chain implicitly determines the remaining nodes in that chain, thereby reducing overall entropy.

Additionally, across all settings, LAR consistently results in the lowest entropy among all routing strategies, particularly when $\tau \leq 0.6$, with a more dramatic drop when $\tau \leq 0.2$,

where entropy approaches zero, indicating minimal anonymity. This limitation is mitigated in GWR, which ensures at least 1.6 to 3 bits of entropy for cascade and stratified topologies, respectively. On the other hand, GPR consistently maintains high entropy across all τ values due to its group-based routing design, which introduces randomness even among geographically close nodes, yielding more than 2 bits of entropy even in the worst case. Finally, SSR achieves the highest entropy overall, consistently guaranteeing at least 5 bits across all configurations, despite its limitations in latency optimization.

Load Balancing Analysis. We now evaluate the impact of applying load balancing on both latency and anonymity across all routing strategies and topologies, as shown in Fig. 4. Specifically, Fig. 4a presents $\ell_{\mathtt{P}}$ after applying LBA in a cascade topology. Compared to the results in Fig. 2, latency increases slightly across all values of τ due to the balancing process redistributing traffic toward more distant nodes, resulting in longer traversal paths. Nonetheless, latency remains below 90 ms for all strategies. Notably, GWR and GPR maintain the lowest latency after applying LBA. LAR performs comparably to GWR and GPR when $\tau \leq 0.8$, but for larger τ, its behavior converges to that of SSR, which consistently shows the least latency reduction. On average, LBA adds approximately 8 ms to the latency for GWR, GPR, and LAR. Additionally, Fig. 4b shows the routing entropy $\mathsf{H}(r)$ after applying LBA. Compared to Fig. 3, entropy increases for all strategies. This is because LBA introduces additional randomness by redistributing traffic, making routes less predictable. Notably, for LAR, entropy no longer drops to zero when $\tau \leq 0.2$. GWR and GPR also maintain higher entropy when $\tau \leq 0.6$, with entropy gradually increasing as τ grows. Eventually, LAR converges toward SSR, which consistently achieves the highest entropy across all settings.

For stratified topologies, latency results are shown in Fig. 4c. Here, the effect of LBA is more pronounced, as stratified topologies—unlike cascades—introduce new randomness for each hop selection, which under LBA results in further randomness in routing. This also leads, as shown in Fig. 4d, to increased entropy. Specifically, entropy of GPR nearly matches the entropy levels of SSR, while GWR consistently provides over 5 bits of entropy. Even LAR, which previously exhibited poor anonymity, now achieves at least 3 bits of entropy, an improvement of at least 1 bit compared to its cascade performance.

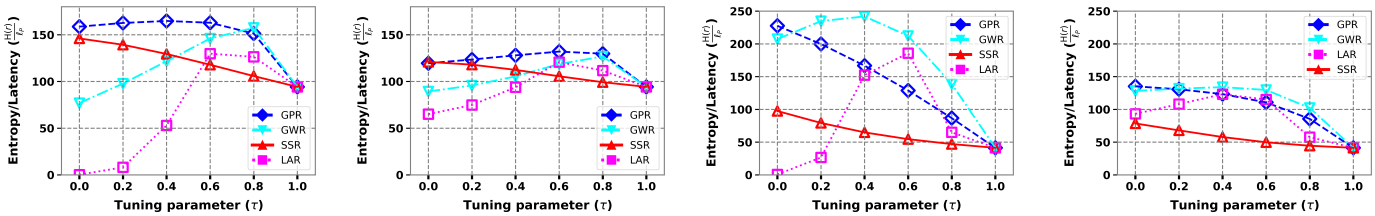
Trade-offs. To fairly compare strategic approaches, we further evaluate the ratio of Entropy/Latency, $\frac{\mathsf{H}(r)}{\ell_{\mathtt{P}}}$, which serves as a balanced performance metric—maximized when both entropy is high and latency is low. We present results for both cascade and stratified topologies constructed with LONA, with and without applying LBA, as shown in Fig. 5. Specifically, Fig. 5a shows this ratio for cascade mixnets without applying LBA. The results indicate that GPR with $N_g = 20$ consistently achieves the best trade-off, peaking at $\tau = 0.6$ with a ratio close to 170. SSR performs well when $\tau \leq 0.4$, due to its high entropy, but GWR surpasses SSR for $\tau > 0.4$, reaching a maximum of 165 at $\tau \geq 0.8$. LAR performs poorly when $\tau \leq 0.6$ but improves notably for higher τ, achieving a peak ratio of 125. Fig. 5b, on the other hand, presents the same ratio for cascade topologies after applying LBA. The overall trend closely mirrors the unbalanced case, confirming that LBA preserves the core characteristics of the routing strategies. However, since LBA redistributes traffic toward less optimal nodes, latency increases slightly, resulting in marginal performance degradation. Still, GPR remains the top performer, peaking near 135. SSR shows slight improvement at $\tau = 0$, but it falls behind for $\tau \geq 0.6$, where GWR consistently tracks close to LAR and aligns more closely with GPR's curve.

Additionally, Fig. 5c shows the Entropy/Latency trade-off for stratified topologies without applying LBA. In this case, we observe relatively higher gains compared to the cascade topology. For $\tau \geq 0.1$, GWR achieves the best performance, reaching a maximum ratio of 250, while GPR peaks around 240. LAR and SSR perform worse overall, except for $0.4 \leq \tau \leq 0.7$, where LAR briefly outperforms GPR, though it still lags behind GWR. Finally, when LBA is applied to stratified mixnets (Fig. 5d), performance slightly degrades due to increased latency from load balancing. Nonetheless, the trend remains consistent, with GWR and GPR maintaining the best Entropy/Latency trade-offs, both reaching ratios close to 145.

Simulation Results. We now present the simulation results of the mixnet using a LONA-structured stratified topology under all proposed routing strategies.[11] Specifically, in Fig. 6, the boxplots represent the interquartile range (25th to 75th percentiles), whiskers denote the 10th to 90th percentiles, and points outside these ranges are considered outliers. Furthermore, the latency distribution of $\ell_{\mathtt{e2e}}$ across all routing strategies is shown in Fig. 6a. The observed trends closely match our analytical findings, revealing that increasing τ results in higher average latency. Both LAR and GWR exhibit very similar behavior for $\tau \leq 0.8$, after which LAR shows a sharp increase in latency. GPR closely follows GWR but with slightly higher latency, while SSR consistently yields the highest latency overall. Interestingly, when routing incorporates higher randomness (i.e., larger τ), the spread in latency decreases, indicating fewer outliers. In contrast, for low τ, the latency spans from very low (for most messages) to very high (for a few messages), as distant nodes may still be selected with very low probability, leading to increased variance.

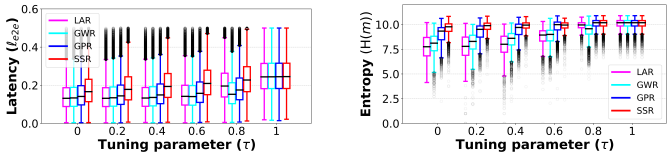
Additionally, message anonymity results are shown in Fig. 6b. As expected, increasing τ generally improves entropy across all strategies. SSR achieves the highest overall entropy, followed closely by GPR. GWR and LAR yield lower entropy, with LAR consistently showing the least. Notably, LAR exhibits outliers with zero entropy when $\tau \leq 0.4$, indicating complete traceability for some messages. In contrast, GWR and GPR mitigate this, offering consistently higher entropy distributions. Overall, GWR maintains high entropy

[11] We also performed simulations for cascade and butterfly topologies, but since trends remained consistent, we report only the stratified setting for brevity.



(a) Cascade (b) LBA, cascade (c) Stratified (d) LBA, stratified

Fig. 5: Trade-offs introduced by mixnet arrangements (LONA) and routing strategies.



(a) Latency ($\ell_{\mathtt{e2e}}$). (b) Entropy ($\mathsf{H}(m)$).

Fig. 6: Simulation results for different routing strategies in a stratified mixnet structured with LONA.

even under moderately biased routing, while GPR consistently achieves entropy levels close to the maximum.

End-to-End Latency Constraint. So far, we have seen that tuning τ provides different trade-offs between latency and entropy. However, there are settings where $\ell_{\mathtt{e2e}}$ on average should be less than a limit ℓ^*. In these settings, it is not immediately clear how to set τ to achieve maximum anonymity while meeting the latency constraint. In such settings, if we increase τ to increase the entropy of routing $\mathsf{H}(r)$ with the hope of heightening the total anonymity of the message $\mathsf{H}(m)$, we may not reach the optimal goal, since $\ell_{\mathtt{e2e}}$, which should be limited, is a combination of both $\ell_{\mathtt{P}}$ and $\ell_{\mathtt{Mix}}$; thus increasing τ, which raises the propagation delay $\ell_{\mathtt{P}}$, consequently reduces the available budget for mixing delays $\ell_{\mathtt{Mix}}$. As a result, messages are less likely to be adequately mixed, potentially lowering the overall anonymity metric $\mathsf{H}(m)$. Hence, a more appropriate strategy is needed to achieve optimal anonymity in this setting.

To investigate this, we perform experiments using LONA with our proposed routing strategies for both stratified and cascade topologies and set $\ell^* = 200$ ms. We vary τ under each routing strategy and calculate the average $\ell_{\mathtt{P}}$ for each setting. Further, to ensure that the total latency remains within ℓ^*, we then compute the total budget for the mixing delay as $\ell_{\mathtt{Mix}} = \ell^* - \ell_{\mathtt{P}}$, which consequently leads to the average mixing delay per mixnode as $\frac{1}{\lambda} = \frac{\ell_{\mathtt{Mix}}}{W \times h}$, where λ is the parameter of exponential delays applied at each mixnode. We then run simulations with these settings and record all average values for $\ell_{\mathtt{P}}$, mixing delay λ^{-1}, and entropy of routing and messages for several values of τ. The results are presented in Tab. I and Tab II for cascade and stratified topologies, respectively.

As expected, when τ increases, $\ell_{\mathtt{P}}$ increases across all strategies. Consequently, the average mixing delay λ^{-1} decreases accordingly to keep $\ell_{\mathtt{e2e}} \leq \ell^*$. On the other hand, increasing τ increases $\mathsf{H}(r)$. However, for $\mathsf{H}(m)$, as this metric is influenced by both the randomness introduced by τ and the mixing delay, we observe concave behavior: as τ increases, $\mathsf{H}(m)$ increases until $\tau = 0.8$, where it peaks and then starts decreasing beyond this point. This indicates further that neither maximizing τ nor maximizing mixing delay leads to higher anonymity. Rather, there is an optimal point for maximizing anonymity of messages. Additionally, in both stratified and cascade topologies, at the optimal τ, GPR provides the highest entropy, with GWR and SSR performing as the second highest, and LAR consistently resulting in the least entropy. Overall, the anonymity achieved with all routing strategies is high and shows different levels of trade-offs between propagation delay $\ell_{\mathtt{P}}$ and mixing delay $\ell_{\mathtt{Mix}}$ to achieve optimal anonymity, which gives a free hand to the designer to select the best approach based on the specific requirements of the network.

Further Security Considerations of OptiMix. We conclude this section by discussing an important security consideration related to the use of strategic routing. Specifically, message paths derived from routing strategies may—while optimizing for latency—increase the likelihood that all selected mixnodes are located within the same country. Such geographic concentration elevates the risk of state-level surveillance by making it easier to coerce nodes into revealing correlations between ingress and egress traffic. To mitigate this risk, we recommend two countermeasures: (1) tuning the routing parameter τ, as higher values (e.g., $\tau \geq 0.6$ across all methods) generally reduce the probability of geographic clustering; and (2) allowing clients to enforce a policy that rejects paths in which all selected mixnodes reside in the same country. In such cases, clients can simply resample a new path from the routing distribution, thereby avoiding the risk entirely.

Our analysis using the Nym dataset further quantifies this threat. Under the GPR and GWR strategies, the probability of all nodes in a path being located in the same country is approximately 0.22 and 0.30, respectively, when $\tau = 0$. This probability decreases sharply as τ increases. That said, not all values of τ necessarily carry the same level of risk in practice. This effect must be evaluated on a per-method basis, and clients may need to be restricted from selecting unsafe values. For instance, as shown in Fig. 3a, the LAR method maintains sufficient routing entropy only within the interval $0.6 \leq \tau \leq 1$. Clients can be constrained to this safe range to prevent insecure path construction.

TABLE I: Latency constraint on end-to-end latency ($\ell_{\mathtt{e2e}}$): This table presents the results for a cascade mixnet structured with the LONA protocol.

τ	**0**				**0.2**				**0.4**				**0.6**				**0.8**				**1**			
Approaches	LAR	GWR	GPR	SSR	LAR	GWR	GPR	SSR	LAR	GWR	GPR	SSR	LAR	GWR	GPR	SSR	**LAR**	**GWR**	**GPR**	**SSR**	LAR	GWR	GPR	SSR
$\ell_{\mathtt{p}}$ (ms)	16	16	18	22	16	16	18	23	18	18	19	25	21	20	21	28	**45**	**26**	**35**	**45**	66	66	66	66
λ^{-1} (ms)	61.3	61.3	60.6	59.3	61.3	61.3	60.6	59.3	60.6	60.3	60.3	58.6	59.6	60.0	59.6	57.3	**51.6**	**58.0**	**55.0**	**51.6**	44.6	44.6	44.6	44.6
$\mathsf{H}(\mathbf{r})$ (bits)	0.0	1.5	2.0	3.6	0.1	1.9	2.6	3.9	1.0	2.5	3.8	4.3	2.9	3.2	4.8	4.9	**5.7**	**4.2**	**5.5**	**5.7**	6.3	6.3	6.3	6.3
$\mathsf{H}(\mathbf{m})$ (bits)	8.0	8.2	8.7	8.1	8.0	8.2	8.8	8.2	8.2	8.4	9.0	8.3	8.5	8.6	9.2	8.5	**9.1**	**9.2**	**9.5**	**9.2**	9	9	9	9

TABLE II: Latency constraint on end-to-end latency ($\ell_{\mathtt{e2e}}$): This table presents the results for a stratified mixnet structured with the LONA protocol.

τ	**0**				**0.2**				**0.4**				**0.6**				**0.8**				**1**			
Approaches	LAR	GWR	GPR	SSR	LAR	GWR	GPR	SSR	LAR	GWR	GPR	SSR	LAR	GWR	GPR	SSR	**LAR**	**GWR**	**GPR**	**SSR**	LAR	GWR	GPR	SSR
$\ell_{\mathtt{p}}$ (ms)	11	13	23	59	11	15	27	75	13	17	33	93	25	23	44	112	**93**	**39**	**68**	**132**	151	151	151	151
λ^{-1} (ms)	63.0	62.3	59.0	47.0	63.0	61.6	57.6	41.6	62.3	61.0	55.6	35.6	58.3	59.0	52.0	29.3	**35.6**	**53.6**	**44.0**	**22.6**	16.3	16.3	16.3	16.3
$\mathsf{H}(\mathbf{r})$ (bits)	0.0	3.0	5.5	6.0	0.3	3.8	5.6	6.1	2.1	4.6	5.8	6.1	4.9	5.2	5.9	6.2	**6.2**	**5.7**	**6.0**	**6.3**	6.3	6.3	6.3	6.3
$\mathsf{H}(\mathbf{m})$ (bits)	7.3	7.6	8.7	9.1	7.3	7.8	8.9	9.2	7.5	8.1	9.1	9.2	8.3	8.4	9.3	9.2	**9.1**	**9.3**	**9.5**	**9.3**	8.9	8.9	8.9	8.9

IV. Mixnode Adversary

In this section, we extend our adversarial model by considering a *mixnode adversary*, which, unlike a GPA that passively observes all communication across the network without access to the internal operations of mixnodes, can directly map inputs to outputs at the nodes it controls. Consequently, if such an adversary compromises all mixnodes along a message's path, the anonymity of that message is entirely compromised. Conversely, the presence of even a single honest mixnode on the path is sufficient to preserve anonymity. In such situations, to maximize their chances of deanonymization, mixnode adversaries must compromise nodes across multiple wings of the mixnet. This leads to a natural way to evaluate the effectiveness of such adversaries, by measuring the probability that they control all nodes along a message path. We refer to this metric as the *Fraction of Fully Corrupted Paths* (FCP). FCP, however, provides a worst-case measurement of the adversary's advantage, as it only accounts for cases where the adversary fully compromises all nodes along a message's path—omitting scenarios in which only parts of the path are compromised. To capture these partial compromise cases, we consider the anonymity of messages using the metric $\mathsf{H}(m)$, achieved through simulations, under the assumption that any message passing through a compromised mixnode gains no additional anonymity.

Lastly, we note that strategic routing in mixnets can increase the likelihood of deanonymization, as traffic tends to concentrate on lower-latency paths. To account for this effect, we consider specific node corruption strategies that the adversary may adopt to maximize their compromise effectiveness, described as follows.

Naive Corruption. Naive corruption is the most basic approach to compromising mixnodes within the mixnet, where the adversary corrupts f mixnodes uniformly at random.

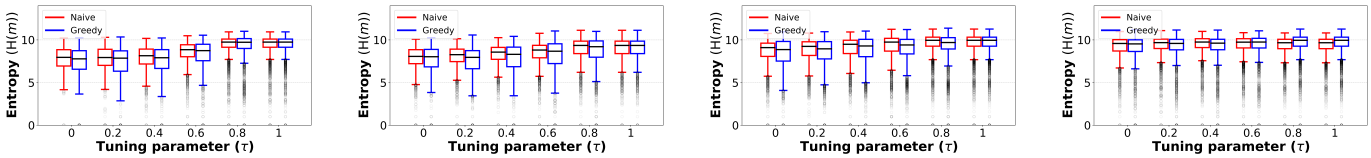
Greedy Corruption. In the Greedy corruption strategy, assuming the adversary has full knowledge of the set of mixnodes used to construct the mixnet, the adversary begins by selecting a node uniformly at random from this set. Then, the adversary selects a second node that minimizes the latency distance from the first selected node. The third node is chosen similarly, such that it minimizes latency to both previously selected nodes. This process continues iteratively until the adversary exhausts their available budget for corrupting f nodes.

Experimental Results. We begin analyzing mixnode adversaries through simulations by evaluating all routing strategies for a stratified mixnet,[12] configured with LONA, where adversaries corrupt 15% of the mixnodes under both *Naive* and *Greedy* corruption strategies. The results are shown in Fig. 7. As evident, across all routing strategies, increasing τ improves message anonymity, indicating enhanced privacy due to increased routing randomness. Additionally, across all strategies, the Naive corruption consistently results in higher message entropy compared to the Greedy model, owing to the latter's more strategic node selection. Notably, while median entropy is only slightly higher under Naive corruption, both strategies exhibit similar behavior at upper percentiles (e.g., 75%), suggesting that Greedy corruption does not significantly affect anonymity in the majority of cases.

Nevertheless, for the lower 25% tail, particularly under LAR routing, the adversary can reduce entropy by up to 2 bits, with an increased occurrence of outliers reaching zero entropy. In contrast, GWR yields a modest improvement in lower-percentile entropy, while GPR and SSR demonstrate more substantial gains. Overall, GPR and SSR offer higher message entropy with reduced variance, while LAR and GWR, although more strategically optimized, still maintain acceptable anonymity levels for the majority of messages.

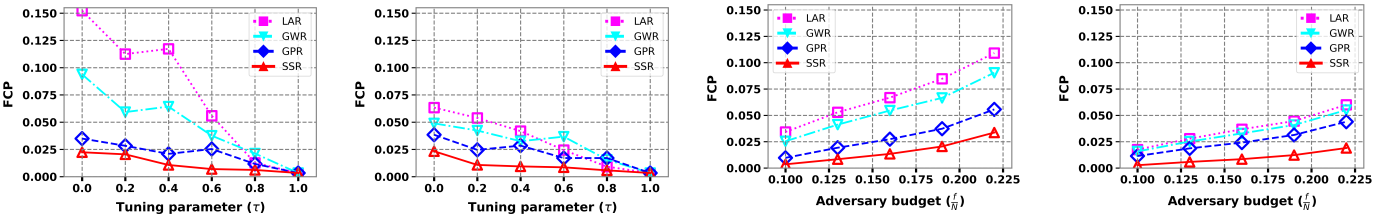
Additionally, we performed experiments under the Greedy corruption model on a stratified mixnet structured with LONA, assessing the FCP of all strategic routing methods both in isolation and in combination with LBA, while varying τ and the adversarial budget, defined as the fraction $\frac{f}{N}$. The results are shown in Fig. 8. Specifically, Fig. 8a illustrates FCP versus τ for all routing strategies when load balancing is not applied, with $\frac{f}{N} = 0.15$ fixed. In this setting, as τ increases, the overall FCP decreases. This is because higher values of τ introduce more randomness into routing, thereby lowering the probability that adversarially compromised paths are selected, and consequently reducing the FCP. Addition-

[12]Note that, for brevity, we focus on the stratified case; however, we observed similar trends across other topologies.



(a) LAR (b) GWR (c) GPR (d) SSR

Fig. 7: Simulation results under the mixnode adversary model for varying values of τ and fixed corruption ratio $\frac{f}{N} = 0.15$, using a stratified mixnet structured with LONA.



(a) FCP vs. τ (b) FCP vs. τ, LBA (c) FCP vs. $\frac{f}{N}$ (d) FCP vs. $\frac{f}{N}$, LBA

Fig. 8: Mixnode adversary results under varying values of τ and adversary budget $\frac{f}{N}$ for a stratified mixnet structured with LONA, both with and without applying LBA.

ally, LAR consistently results in the highest FCP, showing a noticeable gap from the other strategies. Conversely, both GPR and SSR demonstrate strong resilience against mixnode adversaries, thanks to the inherent randomness in their routing, keeping FCP below 0.03 across all τ values. GWR also shows FCP values between those of LAR and GPR/SSR, with FCP falling below 0.03 for $\tau \geq 0.6$.

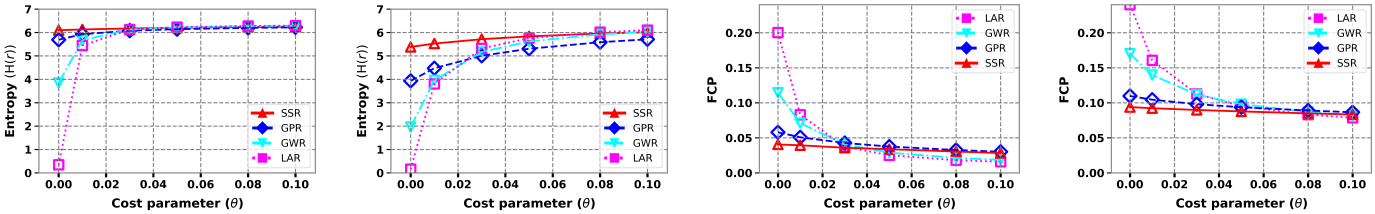
Fig. 8b, on the other hand, presents the same results when routing distributions are balanced using LBA. In this case, we observe that the trends generally align with those of the previous experiment. Additionally, LBA effectively contributes to reducing FCP, particularly for both LAR and GWR, to the extent that their FCP values are nearly halved. This occurs because including LBA rebalances the traffic distribution, which brings higher randomness, thereby reducing the probability of selecting the paths corrupted by the adversary and thus lowering the FCP. Additionally, Fig. 8c and Fig. 8d illustrate the FCP under varying adversarial budgets while setting $\tau = 0.6$, with and without applying LBA, respectively. As shown in both cases, increasing the adversarial budget consistently leads to higher FCP, as a greater number of compromised mixnodes enables the adversary to corrupt more end-to-end paths. Nevertheless, this effect is notably mitigated by LBA, due to its capacity to enhance the randomness in routing and thus reduce the adversary's overall effectiveness.

V. Cover Routing Generation (CRG)

As seen so far, introducing strategic routing inevitably renders a trade-off between latency and anonymity. In this section, we explore whether clients can experience low-latency communication within the mixnet while minimizing anonymity loss. To this end, we propose the *Cover Routing Generation (CRG)* method. At a high level, CRG shifts part of the latency–anonymity trade-off to an additional computation process performed by the client. More precisely, CRG enables a client, intending to strategically route her messages using a specific strategy with a chosen τ, to generate additional *cover routings*, consisting of dummy messages with no actual destination that eventually return to the client. These cover messages are deliberately routed through paths that are not initially favored by the real traffic. For example, consider a cascade mixnet with $d = 3$ chains, and a client who routes real messages through the first, second, and third chains with probabilities $\frac{2}{3}$, $\frac{1}{3}$, and 0, respectively. Under CRG, the cover traffic is routed in a complementary manner, sent through the first, second, and third chains with probabilities 0, $\frac{1}{3}$, and $\frac{2}{3}$, respectively. This ensures that, when real and cover messages are aggregated, the overall routing distribution appears uniform to a GPA, while real messages still benefit from low-latency paths.

Formalizing the Context. To formally explain CRG, assume a client generates real messages following a Poisson distribution with rate μ. In this case, the probability of sending a message to chain i in the first wing is denoted as f_i. Consequently, the message arrival at chain i follows a Poisson process with rate $f_i\mu$. Let $\mu' = \mu \cdot \max(f_i)$ denote the maximum rate directed to any chain. To mask this non-uniformity, the client generates cover traffic following a Poisson distribution with rate $\lambda' = \theta\lambda$, where λ represents the total volume of cover traffic needed to flatten the routing distribution: $\lambda = \sum_{i=1}^{d}(\mu' - \mu f_i)$.[13] Here, $\theta \in [0, 1]$ is a tunable cost parameter. Specifically, $\theta = 1$ achieves perfectly uniform routing, while smaller values yield partial obfuscation at a reduced cost.

[13]Note that adding traffic at a rate of $\mu' - \mu f_i$ to each chain i ensures that all chains receive the same amount of traffic, thereby making the routing distribution uniform.



(a) $\mathsf{H}(r)$ vs. θ, Stratified (b) $\mathsf{H}(r)$ vs. θ, Cascade (c) FCP vs. θ, Stratified (d) FCP vs. θ, Cascade

Fig. 9: Effect of CRG noise on both the entropy of routing $\mathsf{H}(r)$ and the FCP, evaluated using stratified and cascade mixnet topologies structured with LONA. Here, $\tau = 0.2$ and $\frac{f}{N} = 0.2$.

Following this, cover traffic with total rate λ' is distributed across the chains in such a way that each chain i receives dummy traffic at rate $\lambda_i = \theta(\mu' - \mu f_i)$. The resulting routing probability, combining both real and cover messages, for chain i is given by $\frac{\theta(\mu'-\mu f_i)+\mu f_i}{\lambda'+\mu}$, which ensures that the total observed traffic is uniformly distributed across all chains when $\theta = 1$. Additionally, for chains in the wings beyond the first, real messages follow the preferred strategic routing defined by the client, while cover routing is distributed correspondingly to counterbalance the distribution across those chains. See the toy example in Appendix D for further clarification. This approach ensures that the resulting cumulative routing distribution after applying CRG remains, to some extent, close to uniform—especially in the case where $\theta = 1$—while the client continues to benefit from latency-efficient routing for actual messages. Finally, we note that our proposed CRG method satisfies (ϵ, δ)-differential privacy. The formal statement and proof of this result are provided in Appendix D of the extended version [33]. In brief, our analysis shows that the parameter θ is inversely related to both ϵ and δ, indicating that stronger privacy (i.e., higher anonymity guarantees) can be achieved by increasing θ.

Experimental Results. To evaluate the CRG method, we performed experiments using both stratified and cascade mixnets, each structured with LONA and incorporating all routing strategies. Specifically, we measured both the entropy of routing and the FCP (evaluated under Greedy corruption), while setting $\tau = 0.2$ and adversarial budget $\frac{f}{N} = 0.2$. The results are shown in Fig. 9, while varying the cost parameter θ. Fig. 9a and Fig. 9b present the entropy of routing as a function of θ for the stratified and cascade topologies, respectively. As expected, when $\theta = 0$, the entropy values, particularly for LAR and GWR, remain low. However, even a small amount of cover routing (e.g., $\theta \approx 0.01$) results in a sharp increase in entropy in both topologies. This improvement becomes more pronounced as θ increases further, reaching maximum entropy at around $\theta \approx 0.1$. These findings suggest that even modest values of θ in CRG can effectively enhance anonymity.

Additionally, Fig. 9c and Fig. 9d illustrate the FCP when applying CRG as a function of θ for the stratified and cascade topologies, respectively. At first glance, we observe that FCP in the cascade topology is generally about twice as high as in the stratified topology. This is expected, as cascade routing exhibits reduced randomness, since the entire path is determined upon selecting the first node, unlike in the stratified topology, where each hop is selected independently. Moreover, applying strategic routing without any cover routing (i.e., $\theta = 0$)—particularly under LAR and GWR—can lead to a considerable increase in FCP. However, incorporating cover routing causes FCP to drop sharply, even with small values of θ. Specifically, at $\theta \approx 0.1$, FCP remains below 0.05 and 0.1 for the stratified and cascade topologies, respectively. This demonstrates the effectiveness of CRG in mitigating adversarial advantage while preserving the benefits of strategic routing.

VI. Discussion

In this section, we explore the key results of OptiMix, specifically focusing on anonymity, latency, and their corresponding trade-offs, by evaluating the most optimal routing strategies (GWR and GPR). We conduct our analysis on both a medium-sized network using data from the deployed Nym mixnet ($d = 80$) and a larger network derived from the RIPE dataset [40] ($d = 200$). The evaluation includes both stratified and cascade topologies, constructed using LONA, with and without applying LBA. Additionally, we compare OptiMix against both LARMix [34] and LAMP [35], which are only applicable to stratified topologies. The results are summarized in Tab. III. The upper part of the table features schemes that support load balancing, while the lower part presents those that do not. Notably, all reported values for OptiMix are based on the optimal choice of τ, as derived in § III. For LARMix and LAMP, we adopt the optimal strategies specified in their respective original works.

In this context, the key results of OptiMix suggest that for both balanced and unbalanced cases, GWR tends to yield slightly lower latency than GPR, though at the cost of reduced entropy of routing $\mathsf{H}(r)$, which is also evident under mixnode adversary settings, where GPR performs better due to its more randomized routing strategy. Additionally, thanks to the LONA protocol, cascade topologies offer better reduced latency compared to stratified topologies, though often at a higher cost to anonymity.

We also observe that increasing the size of the network (using RIPE) introduces nodes with more diverse latency characteristics, i.e., the larger network includes both very low-latency and high-latency nodes. As a result, latency can often

TABLE III: Key results for OptiMix using RIPE and Nym datasets, and its comparison with LARMix [34] and LAMP [35].

Approaches \Metrics	**Entropy, $\mathsf{H}(r)$ (bits)**				**Latency, $\ell_{\mathbf{P}}$ (ms)**				**Entropy/Latency $\frac{\mathsf{H}(r)}{\ell_{\mathbf{P}}}$ (bits/sec)**				**Anonymity, $\mathsf{H}(m)$ (bits)**				**FCP**			
Topology	**Cascade**		**Stratified**		**Cascade**		**Stratified**		**Cascade**		**Stratified**		**Cascade**		**Stratified**		**Cascade**		**Stratified**	
Dataset	RIPE	Nym	RIPE	Nym	RIPE	Nym	RIPE	Nym	RIPE	Nym	RIPE	Nym	RIPE	Nym	RIPE	Nym	RIPE	Nym	RIPE	Nym
Vanilla	7.6	6.3	7.6	6.3	183	155	183	155	42	41	42	41	10.2	10.1	10.4	10.6	0.02	0.02	0.02	0.02
LARMix [34]	N/A	N/A	6.5	5.2	N/A	N/A	110	106.5	N/A	N/A	59	49	N/A	N/A	10.3	10.2	N/A	N/A	0.05	0.06
OptiMix, GWR & LBA	3.2	4	6	5.6	21	28	25	34	152	143	240	165	10.2	9.8	10.2	9.9	0.12	0.12	0.05	0.04
OptiMix, GPR & LBA	5.7	4.8	7.5	6.3	28	30	39	34	204	160	193	185	10.2	10.1	10.2	10.2	0.07	0.09	0.02	0.02
LAMP [35]	N/A	N/A	3.6	2.8	N/A	N/A	18	20	N/A	N/A	200	140	N/A	N/A	9	8.8	N/A	N/A	0.15	0.16
OptiMix, GWR	2.7	3.4	3.7	4.7	14	25	14	17	193	136	264	276	10.1	8.2	10	8.5	0.18	0.13	0.1	0.07
OptiMix, GPR	5.5	4.8	6.9	5.8	22	25	25	27	250	192	276	215	10.2	9	10	9.6	0.1	0.09	0.03	0.03

be reduced more significantly in strategic routing schemes like GWR compared to GPR. However, the opposite can also occur, as routing methods with more randomness (like GPR) may cause more selection of suboptimal nodes in larger networks, which can lead to slightly increased latency compared to Nym. Overall, the trade-offs and latency reductions across both datasets remain consistent, showcasing the adaptability of OptiMix under different network sizes and topology settings. Additionally, compared to vanilla routing, latency across all OptiMix approaches improves by at least 78%, with Entropy/Latency trade-offs enhanced by 4–6×.

Compared to state-of-the-art baselines, OptiMix offers several notable advantages. Most importantly, it is a general framework that can be seamlessly integrated into various mixnet architectures, including those used in both LARMix and LAMP. Additionally, OptiMix introduces LONA, a protocol for arranging mixnodes in a latency-aware yet unbiasable manner, which operates efficiently and without reliance on any trusted third party. In contrast, LARMix and LAMP either do not provide any arrangement mechanisms or rely on approaches that are inefficient and depend on trusted infrastructure. Moreover, while LARMix adopts a single fixed routing strategy and LAMP uses region- or circle-based routing—often excluding a large number of nodes and thereby significantly reducing anonymity—OptiMix introduces new routing strategies, namely GPR, GWR, and SSR. These methods consistently outperform existing designs in terms of both latency and anonymity.

OptiMix also achieves better computational efficiency. For instance, LARMix, particularly its load balancing component, incurs computational complexity up to $\mathcal{O}(N^5)$. In contrast, OptiMix introduces a load balancing algorithm (LBA) with $\mathcal{O}(N^2)$ complexity, resulting in a practical $\mathcal{O}(N^3)$ reduction in runtime overhead. Considering the results in Tab. III, from a performance perspective, LARMix exhibits substantially higher latency and up to $4\times$ weaker Entropy/Latency trade-offs compared to OptiMix. Compared to LAMP, OptiMix avoids oversimplified assumptions—such as the omission of load balancing—while achieving comparable or improved latency. Under equivalent evaluation settings, OptiMix consistently delivers up to $2\times$ better Entropy/Latency trade-offs, particularly when employing the GWR strategy. Finally, OptiMix demonstrates stronger resilience to mixnode adversaries and achieves higher overall anonymity. It is also the first to introduce the CRG mechanism that mitigates anonymity degradation by injecting cover traffic—a capability not addressed in prior systems.

VII. Conclusion

In OptiMix, we provided mixnet arrangements, strategic routing, load balancing, and cover routing mechanisms to enable low-latency mixnets with general topologies, while preserving strong anonymity. Our evaluation highlighted that OptiMix significantly improves the latency/anonymity trade-offs compared to vanilla mixnets, as well as state-of-the-art solutions (LARMix and LAMP). Notably, OptiMix's protocols are efficient and do not rely on trusted parties, making it a suitable candidate for practical deployment.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and constructive suggestions. We are also grateful to Claudia Diaz for her valuable feedback during the rebuttal phase. This research was supported in part by CyberSecurity Research Flanders under reference number VOEWICS02.

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, 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.

[5] 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.

[6] I. Cascudo and B. David, "Albatross: publicly attestable batched randomness based on secret sharing," in *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26*. Springer, 2020, pp. 311–341.

[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] A. Czumaj and B. Vöcking, "Thorp shuffling, butterflies, and non-markovian couplings," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2014, pp. 344–355.
[10] G. Danezis and C. Troncoso, "Vida: How to use bayesian inference to de-anonymize persistent communications," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2009, pp. 56–72.
[11] 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.
[12] C. Diaz, "Anonymity and privacy in electronic services," *Heverlee: Katholieke Universiteit Leuven. Faculteit Ingenieurswetenschappen*, 2005.
[13] C. Diaz, H. Halpin, and A. Kiayias, "The nym network," 2021.
[14] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.
[15] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 193–206.
[16] 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.
[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] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go-mixes providing probabilistic anonymity in an open system," in *International Workshop on Information Hiding*. Springer, 1998, pp. 83–98.
[19] P. A. Knight, "The sinkhorn–knopp algorithm: convergence and applications," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 261–275, 2008.
[20] K. Kohls and C. Diaz, "{VerLoc}: Verifiable localization in decentralized systems," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2637–2654.
[21] A. Kwon, D. Lu, and S. Devadas, "Xrd: scalable messaging system with cryptographic privacy," in *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation*, 2020, pp. 759–776.
[22] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, 2019, pp. 203–226.
[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] Python, "Event discrete, process based simulation for python." https://pypi.org/project/simpy/, 2013.
[27] M. Rahimi, "CLAM: client-aware routing in mix networks," in *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2024, Baiona, Spain, June 24-26, 2024*, F. Pérez-González, P. C. Alfaro, C. Krätzer, and H. V. Zhao, Eds. ACM, 2024, pp. 199–209. [Online]. Available: https://doi.org/10.1145/3658664.3659631
[28] ——, "Larmix++: Latency-aware routing in mix networks with free routes topology," in *International Conference on Cryptology and Network Security*. Springer, 2024, pp. 187–211.
[29] ——, "Malaria: management of low-latency routing impact on mix network anonymity," in *2024 22nd International Symposium on Network Computing and Applications (NCA)*. IEEE, 2024, pp. 193–202.
[30] ——, "Dp-mix: Differentially private routing in mix networks," in *Proceedings of the 41st Annual Computer Security Applications Conference*, 2025.
[31] ——, "Mocha: Mixnet optimization considering honest client anonymity," in *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, 2025, pp. 98–107.
[32] ——, "Parsan-mix: Packet-aware routing and shuffling with additional noise for latency optimization in mix networks," in *International Conference on Applied Cryptography and Network Security*. Springer, 2025, pp. 159–188.
[33] ——, "Optimix: Scalable and distributed approaches for latency optimization in modern mixnets (extended version)," *Cryptology ePrint Archive*, 2026.
[34] M. Rahimi, P. K. Sharma, and C. Diaz, "Larmix: Latency-aware routing in mix networks," in *The Network and Distributed System Security Symposium*. Internet Society, 2024.
[35] ——, "Lamp: Lightweight approaches for latency minimization in mixnets with practical deployment considerations," in *The Network and Distributed System Security Symposium*. Internet Society, 2025.
[36] 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.
[37] C. E. Shannon, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.
[38] 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.
[39] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, no. 2, pp. 343–348, 1967.
[40] R. N. Staff, "Ripe atlas: A global internet measurement network," *Internet Protocol Journal*, vol. 18, no. 3, pp. 2–26, 2015.
[41] N. Tyagi, Y. Gilad, D. Leung, M. Zaharia, and N. Zeldovich, "Stadium: A distributed metadata-private messaging system," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 423–440.
[42] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: Scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015, pp. 137–152.
[43] 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. Algorithms and Protocols

In this section, we provide the pseudocodes of the protocols and algorithms described in OptiMix.

B. Background and Preliminaries

In this section, we briefly introduce some of the protocols used as subroutines in our approaches. We assume the existence of a broadcast channel, which can be implemented using a blockchain. The blockchain acts as a public, append-only log maintained in a decentralized manner by the nodes. Alternatively, other secure broadcast mechanisms with Byzantine fault tolerance [22] can be employed. These mechanisms ensure both integrity and availability, guaranteeing that once information is uploaded, it remains publicly accessible to all participants and cannot be altered. Considering this, we describe the subroutines used in OptiMix as follows.

Peer-to-Peer Latency Measurements. Our protocols rely on peer-to-peer latency measurements among mixnodes in a mixnet. To facilitate this, we assume the use of the VerLoc protocol [20], a decentralized approach that allows nodes to verify each other's locations and estimate link latencies. Operating under a majority-honest assumption, VerLoc yields accurate

Algorithm: Load Balancing Algorithm (LBA)

1) **Load Balancing Algorithm:**
   Input: Imbalance matrix Γ_{Im}.
   Output: Balanced matrix Γ_{Ba}.
   a) Initialize count $= 0$,
   b) **While True**:
   - **For** each column in Γ_{Im}:
     - **If** sum(column) $\neq 1$:
       - Normalize the column entries so that sum(column) = 1,
       - Increment count by 1,
   - **If** count = 0:
     - **Break**;
   - **For** each row in Γ_{Im}:
     - Normalize the row entries so that sum(row) = 1,
   - Reset count $= 0$,

   c) **Output:** $\Gamma_{\text{Ba}} = \Gamma_{\text{Im}}$.

◀ **Complexity** : $\mathcal{O}\left(N^2 \log(\frac{1}{\epsilon_e})\right)$.

Fig. 10: Algorithm: Load Balancing Algorithm (LBA).

latency and location estimates, ensuring that deviations by a minority of nodes do not affect the integrity of the results. The protocol is resilient to adversaries corrupting up to 30% of the mixnodes. If an adversary controls a larger fraction of the network, they may gain partial advantages in compromising client anonymity.

Randomness Generation. In OptiMix, uniformly random values are required to preserve the unbiasability of the mixnet arrangement. To this end, we employ the randomness generation mechanism proposed in [6], a state-of-the-art multiparty randomness generation algorithm. This method involves N parties (mixnodes) collaboratively producing t^2 random values, where $N = 2f + t$ and f represents the number of adversarial parties. In this protocol, each party i $(1 \leq i \leq N)$ shares t secrets using Shamir's secret sharing scheme with all other parties. These shares, further accompanied by NIZK proofs, are publicly verifiable. After the sharing phase, a qualified set $\mathcal{Q}$ of $f+t$ parties produces the matrix $S_{(f+t)\times t}$, with each row representing the t secrets shared by a party in $\mathcal{Q}$. The matrix S is then multiplied by a pre-agreed matrix $M_{\ell\times(t+f)}$, resulting in t^2 uniformly random secrets.

The computational complexity of this approach is $\mathcal{O}(N^2)$ for all parties, with verification costs of $\mathcal{O}(fN)$. (See § 4.3 in [6] for further details.)

C. Supplementary Material for § II-C

This section provides detailed analyses of the Load Balancing Algorithm (LBA), including a toy example, the convergence proof of LBA, and its comparison with the balancing algorithm in LARMix [34].

Toy Example. To illustrate the LBA approach more clearly, consider a case where the routing matrix between two successive wings of a butterfly mixnet is derived from low-latency routing as follows:

$$\Gamma = \begin{bmatrix} 0.2 & 0.1 & 0.3 \\ 0.1 & 0.4 & 0.4 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}.$$

The column sums of this matrix are $[0.8, 1, 1.2]$, indicating that the second mixnode at the next hop is balanced, the third is overloaded, and the first is underloaded. To begin balancing the matrix, we multiply the columns by $\frac{1}{0.8}$, 1, and $\frac{1}{1.2}$, respectively, resulting in:

$$\Gamma = \begin{bmatrix} 0.25 & 0.1 & 0.25 \\ 0.125 & 0.4 & 0.33 \\ 0.625 & 0.5 & 0.42 \end{bmatrix}.$$

Next, we normalize the rows to ensure that each row sums to 1:

$$\Gamma = \begin{bmatrix} 0.42 & 0.16 & 0.41 \\ 0.15 & 0.46 & 0.38 \\ 0.41 & 0.33 & 0.27 \end{bmatrix}.$$

If our target precision is $\epsilon_e = 0.1$ (i.e., $\log\left(\frac{1}{\epsilon_e}\right) = 1$), we can stop here. For higher precision, we continue iterating, eventually obtaining:

$$\Gamma_b = \begin{bmatrix} 0.43 & 0.18 & 0.39 \\ 0.15 & 0.49 & 0.36 \\ 0.42 & 0.33 & 0.25 \end{bmatrix}.$$

Convergence Proof of LBA.

Theorem 1. *The load balancing algorithm (LBA) converges in at most* $\mathcal{O}\left(\log_{10}\left(\frac{1}{\epsilon_e}\right)\right)$ *steps, resulting in a computational complexity of* $\mathcal{O}\left(N^2 \log_{10}\left(\frac{1}{\epsilon_e}\right)\right)$.

Proof. Note that Γ is a square matrix of size $d \times d$. Our balancing algorithm is equivalent to the Sinkhorn algorithm [39], which states that any non-negative matrix with positive entries can be converted into a doubly stochastic matrix by alternately normalizing its rows and columns. This yields unique diagonal matrices D_1 and D_2 such that $D_1 \Gamma D_2$ is doubly stochastic.

Since Γ satisfies Sinkhorn's criteria, each iteration consists of $2d^2$ operations (row and column normalization). According to [19], convergence to a precision ϵ_e requires $\mathcal{O}(\log(\frac{1}{\epsilon_e}))$ iterations. Thus, total complexity is:

$$\mathcal{O}\left(N^2 \log\left(\frac{1}{\epsilon_e}\right)\right),$$

which proves Theorem 1. □

Comparison with LARMix Greedy Approach. The LARMix framework introduces a greedy balancing algorithm that starts by identifying underloaded nodes and reassigning loads to achieve balance (see [34], pp. 5–6). However, this

Protocol: Latency Optimized Node Arrangement (LONA) Initialization

1) **Inputs:** A set of mixnodes $\mathcal{N}$ with size $N = d \cdot W \cdot h$.
2) **Setup:** Each mixnode i generates a secret key $sk_i \in \mathbb{Z}_q$ and computes the corresponding public key $pk_i = {g_1}^{sk_i}$. Each pk_i is registered on the public ledger for $1 \leq i \leq N$. ◀ **Complexity** : $\mathcal{O}(1)$.
3) Mixnodes create the set $\mathcal{S}_{ID}$, featuring pk_i values in a random order to set mixnodes identifiers, ranging from pk_1 to pk_N. ◀ **Complexity** : $\mathcal{O}(1)$.
4) **Run VerLoc Protocol:** All mixnodes collaboratively execute the VerLoc protocol to obtain reliable latency measurements. ◀ **Complexity** : $\mathcal{O}(N^2)$.
   Each mixnode M_{pk_i} $(1 \leq i \leq N)$ posts the sorted latency set $\mathcal{S}_{L_i}$ to the public ledger. ◀ **Complexity** : $\mathcal{O}(N \log N)$.
5) **Run Randomness Generation Protocol:** Set $t = \lceil \sqrt{d \cdot W + h - 1} \rceil$ and $\mathcal{P} = \mathcal{N}$. ◀ **Complexity** : $\mathcal{O}(N^2)$.
6) **Output:** (1) $\mathcal{S}_R = \{r_i \mid 1 \leq i \leq d \cdot W + h - 1\}$, (2) $\mathcal{S}_P = \{r_i \mid d \cdot W + 1 \leq i \leq d \cdot W + h - 1\}$, (3) $\mathcal{S}_{L_i} \quad (1 \leq i \leq N)$, (4) $\mathcal{S}_{ID}$, (5) $\mathcal{N}$.

Fig. 11: Initialization of the Latency Optimized Node Arrangement (LONA).

Protocol: Latency Optimized Node Arrangement (LONA) Continued

1) **Node Assignment Algorithm:**
   Input: A set of mixnodes $\mathcal{N} = \{M_{pk_1}, M_{pk_2}, \ldots, M_{pk_N}\}$ with public keys $\{pk_i\}$, randomness $\mathcal{S}_R = \{r_i \mid 1 \leq i \leq d \cdot W + h - 1\}$, prioritization set $\mathcal{S}_P = \{r_i \in \mathcal{S}_R \mid d \cdot W + 1 \leq i \leq d \cdot W + h - 1\}$, latency vectors $\mathcal{S}_{L_i}$ for $1 \leq i \leq N$, and identifiers $\mathcal{S}_{ID}$.
   Output: A set of chains $\bigcup_{i=1}^{d \cdot W} \mathcal{S}_{C_i}$ with assigned nodes.
   a) Initialize empty sets for all chains, denoted as $\mathcal{S}_{C_i}$ for $1 \leq i \leq d \cdot W$.
   b) Define $\mathcal{C}_{ID} = \{1, 2, \ldots, d \cdot W\}$ to represent chain identifiers. Set up $F_{ID}(\cdot)$, a function mapping indices to public keys (from $\mathcal{S}_{ID}$) or chain identifiers (from $\mathcal{C}_{ID}$).
   c) For each $r_i \in \mathcal{S}_R - \mathcal{S}_P$, where $1 \leq i \leq d \cdot W$, compute:

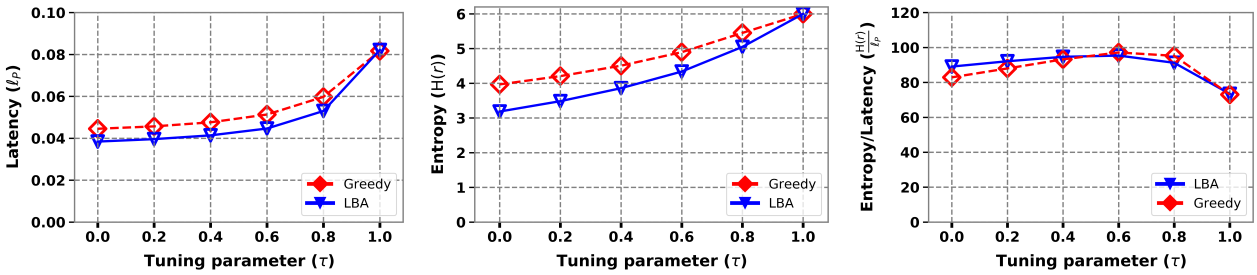
$$\mathtt{Id}_i = F_{ID}\left([\mathcal{H}(r_i \| g_1) \mod |\mathcal{S}_{ID}|], \mathcal{S}_{ID}\right), \ \mathtt{Ch}_i = F_{ID}\left([\mathcal{H}(r_i \| g_2) \mod |\mathcal{C}_{ID}|], \mathcal{C}_{ID}\right).$$

   Assign the mixnode $M_{\mathtt{Id}_i}$ to chain $\mathcal{S}_{C_{\mathtt{Ch}_i}}$ as: $\mathcal{S}_{C_{\mathtt{Ch}_i}} = \mathcal{S}_{C_{\mathtt{Ch}_i}} \cup \{M_{\mathtt{Id}_i}\}$. Update the sets $\mathcal{S}_{ID}$, $\mathcal{N}$, and $\mathcal{C}_{ID}$ by removing the selected mixnode and chain.
   d) To assign subsequent $h - 1$ nodes for each chain: Initialize *Temp* $= 0$.
   While *Temp* < 1:
   - Reset $\mathcal{C}_{ID} = \{1, 2, \ldots, d \cdot W\}$, and initialize *count* $= 0$.
     While *count* < 1:
     – For the k-th hop assignment $(2 \leq k \leq h)$, use the $(k-1)$-th element of $\mathcal{S}_P$, $r_{d \cdot W + k - 1}$, to compute:

$$\mathtt{Ch}_i = F_{ID}\left(\left[\mathcal{H}(r_{d \cdot W + k - 1} \| g_1^i \| g_2) \mod |\mathcal{C}_{ID}|\right], \mathcal{C}_{ID}\right).$$

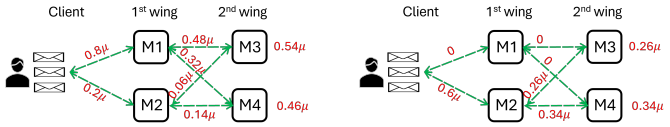
     – Using the latency vector $\mathcal{S}_{L_{\mathtt{Ch}_i^{k-1}}}$, select the next mixnode with the lowest latency relative to the previously assigned mixnode $M_{\mathtt{Ch}_i^{k-1}}$: $M_{\mathtt{Id}_i} = \arg\min_{M_{\mathtt{Id}_j}} \left(\mathcal{S}_{L_{\mathtt{Ch}_i^{k-1}}}\right)$ such that $M_{\mathtt{Id}_j} \in \mathcal{N}$.
     – Assign $M_{\mathtt{Id}_i}$ to $\mathcal{S}_{C_{\mathtt{Ch}_i}}$, and update $\mathcal{S}_{ID}$, $\mathcal{N}$, and $\mathcal{C}_{ID}$, and increment *count*$+= \frac{1}{|\mathcal{S}_R - \mathcal{S}_P|}$.
   - Increment *Temp*$+= \frac{1}{|\mathcal{S}_P|}$.
   e) **Output:** The final mixnet configuration, represented as $\bigcup_{i=1}^{d \cdot W} \mathcal{S}_{C_i}$, is obtained. ◀ **Complexity** : $\mathcal{O}(N \log N)$.

Fig. 12: Protocol: Continued Latency Optimized Node Arrangement (LONA).



(a) Latency (b) Entropy (c) Entropy/Latency

Fig. 13: Comparative analysis of LBA and Greedy LARMix balancing algorithms.



(a) Real traffic distribution (b) Dummy traffic distribution

Fig. 14: Cover Routing Generation (CRG) mechanism

method is more complex than our LBA approach. To fairly compare LBA and Greedy LARMix, we ran experiments using the general setup defined in § III. Specifically, we set $W = 1$, $h = 3$, $d = 64$, constructed the mixnet using LONA, and applied the GPR routing mechanism with $N_g = 20$. We then applied both LBA and Greedy LARMix while varying the parameter τ. In Fig. 13a, we observe the average latency ℓ_P after applying both methods. For nearly all τ values, LBA achieves up to 15% lower latency than Greedy LARMix, indicating better low-latency routing while keeping mixnode loads balanced.

However, this comes at a slight reduction in routing entropy $\mathsf{H}(\mathsf{r})$, as shown in Fig. 13b. When τ is small, LBA reduces entropy by up to 3.2 bits (0.8 bits less than Greedy LARMix). For other τ values, the gap is around 0.5 bits. This reduced entropy is not necessarily a drawback—it reflects a more efficient and latency-optimized routing. Finally, Fig. 13c shows the entropy-to-latency ratio. LBA outperforms when $\tau \leq 0.6$, while Greedy LARMix is slightly better for $\tau \geq 0.6$. Overall, LBA provides significant latency improvements and is computationally more efficient—faster by a factor of $\mathcal{O}(N^3)$—while maintaining robust anonymity guarantees.

D. Expanding on § V

In § V, we introduced a methodology that enables clients with additional computational resources to trade off latency against anonymity by injecting cover (dummy) traffic. The goal of this mechanism is to neutralize the statistical biases introduced by latency-aware routing. To clarify the idea, we now present a toy example that illustrates how a client can achieve uniform traffic distribution across mixnet wings. Consider a stratified mixnet with two wings ($W = 2$) and two mixnodes per wing ($d = 2$), as shown in Fig. 14. Each chain has $h = 1$ mixnode. Assume the client generates messages according to a Poisson distribution with rate μ, sending 80% (0.8μ) of the messages to the first mixnode (M_1) and 20% (0.2μ) to the second mixnode (M_2) in the first wing. This produces a biased traffic distribution at the first wing.

To restore uniformity, the client invokes CRG with $\theta = 1$, injecting 0.6μ of dummy traffic to M_2 (i.e., $(0.8 - 0.2)\mu$), resulting in both M_1 and M_2 receiving 0.8μ in total. However, this correction must also propagate to subsequent wings. Suppose M_1 routes messages to M_3 and M_4 with probabilities 0.6 and 0.4, respectively, and M_2 routes to M_3 and M_4 with probabilities 0.3 and 0.7. The real traffic received by M_3 and M_4 would then be 0.54μ and 0.46μ, respectively—again, a biased distribution.

Since M_2 also received 0.6μ of dummy traffic (Fig. 14b), the client can now redistribute this dummy traffic to balance the second wing. Specifically, 0.08μ is directed to M_4, and the remaining 0.52μ is split evenly between M_3 and M_4 (i.e., 0.26μ each). As a result, both M_3 and M_4 receive exactly 0.8μ, restoring uniformity. This procedure can be repeated recursively across all subsequent wings to ensure a uniform traffic distribution throughout the mixnet, thus mitigating the risks of anonymity leakage due to traffic analysis.

E. Artifact Appendix

1) Description & Requirements: This artifact appendix provides details on how to reproduce the results presented in OptiMix. OptiMix leverages two types of environments to evaluate its methodologies: (1) an analytical setting and (2) a simulation setting. In the analytical scenario, each technique is evaluated using datasets from Nym or RIPE, assessing the effectiveness of OptiMix methods in reducing latency while quantifying the associated anonymity loss. The methods evaluated include LONA (node adjustments), multiple routing strategies (LAR, GWR, GPR, and SSR), LBA (load balancing), and CRG (cover traffic generation). In contrast, the simulation setting—implemented in *Python* using the *SimPy* library—models the full end-to-end flow of communication in mixnets. Within the simulation, clients generate messages that are forwarded through mixnodes according to OptiMix routing strategies. Each mixnode applies mixing processes to anonymize and forward packets, with delays modeled using empirical measurements from Nym or RIPE to capture realistic inter-node transmission latency.

The artifact comprises approximately 20K lines of Python code and reproduces all results presented in the main body of the paper.[14]

How to access: The artifact repository is available via a persistent DOI: https://doi.org/10.5281/zenodo.17054156, and also on GitHub: https://github.com/OptiMixnet/OptiMix.

Hardware dependencies: Note that the artifact includes precisely the same configurations and settings presented in OptiMix. The only difference is that the number of iterations has been scaled down so that the artifact can run on standard systems with 16 GB RAM and 50 GB of disk space.

The estimated runtimes are based on executing the artifact on a system with the following specifications:

- 64-bit CPU: Intel® Core™ i7-10850H @ 2.70 GHz (1 core used)
- Physical Memory (RAM): 16 GB

Software dependencies: Additionally, the server must have Python 3.8.10 installed. Before running the experiments, it is necessary to verify that the `dependencies.txt` file is satisfied. We emphasize the importance of adhering to the specified hardware and software dependencies to ensure that the experiments complete within the expected time limits.

Benchmarks: OptiMix relies on latency and geographical data from two datasets: (i) *Nym*, which captures latency between nodes in the deployed Nym mixnet, and (ii) *RIPE*, which provides latency measurements among globally distributed nodes for broader network topologies. Both datasets are included in the repository.

2) Artifact Installation & Configuration: After setting up a system with Python version 3.8.10, one can execute the artifact by first installing the required dependencies using the script specified in `dependencies.txt`, which is provided in the repository bundled with the code. Upon installing the dependencies, running `Main.py` with the arguments explained below will generate the results, which will be saved in the `Figures` or `Tables` directories.

3) Major Claims: The major claims of OptiMix are summarized as follows:

- (C1): LATENCY PATTERN
  Fig. 2 illustrates the latency incurred in mixnets when applying different routing strategies. As the value of τ increases, latency also increases across all strategies. Overall, LAR, GWR, and GPR exhibit comparable latency, while SSR consistently incurs the highest latency.
- (C2): ENTROPY PATTERN
  Fig. 3 shows that increasing τ results in higher entropy across all routing strategies. In particular, LAR exhibits very low entropy for $\tau \leq 0.6$, while SSR consistently achieves the highest entropy overall.
- (C3): IMPACT OF LOAD BALANCING
  Fig. 4 evaluates the effect of applying the load balancing algorithm (LBA) to the routing strategies. It demonstrates that both latency and entropy increase across all values of τ for cascade and stratified topologies—compared to the results in Figs. 2 and 3.
- (C4): OVERALL PERFORMANCE
  Fig. 5 highlights that when OptiMix is applied to cascade and stratified topologies, with or without LBA, the overall performance—measured as the ratio `Entropy/Latency`—is maximized when using either GPR or GWR strategies across all values of τ.
- (C5): MIXNODE ADVERSARY
  Fig. 8 presents the fraction of fully compromised paths (FCP) under an adversary controlling a subset of mixnodes. The FCP is shown as a function of τ and the adversarial budget $\frac{f}{N}$, where f denotes the number of compromised nodes. The results indicate that increasing τ or decreasing $\frac{f}{N}$ reduces the FCP.
- (C6): COVER ROUTING
  Fig. 9 demonstrates that applying the cover routing generation (CRG) method—especially with a slight increase in the cost parameter θ—results in higher entropy and lower FCP across both stratified and cascade topologies.

4) Evaluation: This section details the set of experiments conducted to support the main claims presented in the artifact appendix. Executing these experiments produces results stored in the `Results`, `Figures`, or `Tables` directories. Additionally, we explain how to run scripts to regenerate specific results corresponding to figures or tables presented in the main body of the paper.

Experiment (E_0) [Setup] (≤ 5 min):

This experiment prepares all required datasets and auxiliary files. It must be run once before executing any other experiment.

[Preparation and Execution] Run the following command with the `Input` argument set to `0`:

```
python3 Main.py
```

[Results] The required files will be generated and stored in the `Results` folder or the root directory of the artifact.

Experiment (E_1) [Figures 2–5] (≤ 5 min):

This experiment supports claims **C1**–**C4** by generating baseline latency and entropy results.

[Preparation and Execution] Run the following command with the `Input` argument set to `1`:

```
python3 Main.py
```

[Results] The results will be saved in the `Figures` folder as: Fig. 2a–2d, Fig. 3a–3d, Fig. 4a–4d, and Fig. 5a–5d.

[Note] You may encounter the following warnings during execution, which are safe to ignore:

```
RuntimeWarning: Mean of empty slice.
  out=out, **kwargs

invalid value encountered in scalar divide
  ret = ret.dtype.type(ret / rcount)
```

[14] As the paper was accepted at the time of artifact submission, the AEC reviewed and validated all results reported in the main body of the camera-ready version.

TABLE IV: Mapping of input arguments to specific figures and tables.

Experiment	Input	Experiment	Input
Fig. 2	22	Fig. 3	33
Fig. 4	44	Fig. 5	55
Fig. 6	66	Fig. 7	77
Fig. 8	88	Fig. 9	99
Tab. I	100	Tab. II	200
Tab. III	300	—	—

Experiment (E_2) [Figure 8] (≤ 5 min):

This experiment evaluates the impact of mixnode adversaries, supporting claim **C5**.

[Preparation and Execution] Run the following command with the `Input` argument set to `2`:

```
python3 Main.py
```

[Results] The output will include Fig. 8a–8d, saved in the `Figures` folder.

Experiment (E_3) [Figure 9] (≤ 5 min):

This experiment evaluates the cover routing strategy and supports claim **C6**.

[Preparation and Execution] Run the following command with the `Input` argument set to `3`:

```
python3 Main.py
```

[Results] The results will be saved in the `Figures` folder as Fig. 9a–9d.

Experiment (E^*) [All Others] (≤ 40 min):

This experiment allows the user to generate any figure or table shown in the main body of the paper, regardless of whether it is linked to a claim.

[Preparation and Execution] Refer to Table IV below for the corresponding `Input` argument value and run:

```
python3 Main.py
```

[Results] Figures will be saved in the `Figures` folder. Tables will be saved in the `Tables` folder as PDF files. However, in cases where a LaTeX installation is not available, table data should be printed directly in the terminal.

[Note] You may encounter the following warnings while running experiments related to Fig. 2–5 or Tab. III. These warnings do not affect the correctness of the results and can be safely ignored:

```
RuntimeWarning: Mean of empty slice.
  out=out, **kwargs
```

```
invalid value encountered in scalar divide
  ret = ret.dtype.type(ret / rcount)
```

5) Parameter Settings and Execution Time: The network parameter settings are largely consistent across all experiments and match those described in the evaluation setup (see page 7 of the main body). Accordingly, `config.py` is initialized to reflect these settings precisely. In each experiment, however, a single parameter is varied. For example, in Figs. 2–7 and Tabs. I–II, the parameter τ is varied; Fig. 8 varies the adversary's budget $\frac{f}{N}$, and Fig. 9 varies the cost parameter θ.

The artifact automatically applies the corresponding configuration changes when the appropriate input argument is provided. Therefore, manual modification of `config.py` is not required to reproduce the experiments.

The only difference between the artifact and the paper lies in the number of iterations used in certain experiments, which is set to 5 in the artifact to ensure that results are generated within a reasonable timeframe (each experiment completes in under 40 minutes). Users who wish to increase the number of iterations may do so manually in `config.py`.

Users may also modify `config.py` to explore experiments not explicitly evaluated in the paper. However, such modifications require a deeper understanding of mixnets, as these are complex systems where changes to one parameter may influence others. In many cases, specific parameter combinations can be incompatible or may significantly affect the results. Moreover, some parameters are initialized based on prior work, with default values selected to enable meaningful comparisons.

Nonetheless, the following parameters in `config.py` can be safely modified within specific intervals:

- `Iterations` $= 5 \rightarrow$ Can be increased to improve accuracy; note that this change increases computational cost linearly.
- `Num_targets` $= 200 \rightarrow$ Specifies the number of target messages in simulations; can be set to any integer in $[20, 200]$.
- `run_time` $= 0.5 \rightarrow$ Defines the duration of each simulation time slot; can be set to any real value in $[0.3, 1.0]$.
- `delay1` $= 0.05 \rightarrow$ Represents the average delay imposed on each message upon entering mixnodes; can be set to any real value in $[0.03, 0.08]$.
- `e2e_delay` $\rightarrow$ Defines the end-to-end latency limit used in Tabs. I and II; can be varied between 0.16 and 0.4.

Other parameters in the configuration should not be modified without sufficient expertise. We recommend that users contact the authors directly if such changes are necessary.