PARSAN-Mix: Packet-Aware Routing and Shuffling with Additional Noise for Latency Optimization in Mix Networks (Extended Version)

Mahdi Rahimi[1] [0009−0003−0223−9082]

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

Abstract. Mix networks (mix-nets) offer strong anonymity by routing client packets through intermediary hops, where they are shuffled with other packets to obscure their origins from a global adversary monitoring all communication exchanges. However, this anonymity is achieved at the expense of increased end-to-end latency, as packets traverse multiple hops (incurring routing delays) and experience additional delays at each hop for shuffling purposes. Consequently, the overall latency for delivering a message—comprising multiple packets—is determined by the packet with the highest combined routing and shuffling delay, which can significantly degrade the client experience, particularly in latency-sensitive applications.

To address this issue, our work **first** derives the theoretical statistics of the total latency experienced by a message, revealing a clear correlation between latency and the number of packets. **Second**, we propose two approaches to reduce this total latency. First, we present a method to adjust the shuffling delays at each hop, offsetting potential anonymity loss by integrating client-generated noise, backed by differential privacy guarantees. Next, we introduce packet-aware routing techniques, offering two novel methods that prioritize messages with more packets, forwarding them through faster links. However, this may cause certain nodes to be overloaded with disproportionate traffic. To solve this, we **third** introduce an efficient load-balancing algorithm to redistribute traffic without compromising the packet-aware nature of the routing. **Finally**, through comprehensive analytical and simulation experiments, we validate our theoretical latency bounds and evaluate the efficacy of our latency management strategies. The results confirm both methods substantially reduce latency with minimal impact on anonymity, while the strategic routing method remains robust against advanced adversarial attacks.

Keywords: Anonymity · Mix-nets · Latency.

1 Introduction

Anonymous communication, which conceals the identity of communication endpoints within a network, has been designed in various ways over the past three

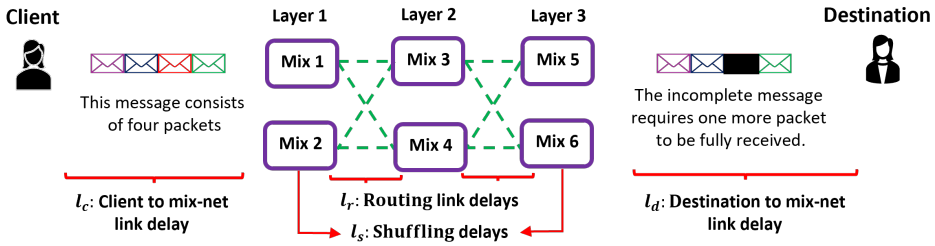


Fig. 1: Overview of mixnet communication process.

decades [29]. Among these methods, one of the most widely used systems is the Tor overlay network [12], which serves over two million daily users globally. In Tor, a client anonymizes their connection by selecting three intermediary relays—designated as the Guard, Middle, and Exit relays—to form a circuit through which traffic is routed to the destination. In this scenario, each relay forwards the client's packets to the next relay, possessing knowledge only of its immediate predecessor and successor. This configuration ensures that the client's communication remains anonymous. However, this anonymity can be compromised if an adversary controls both the Guard and Exit relays simultaneously or gains access to the Autonomous Systems (ASes) that handle traffic between the client and the Guard relay, as well as between the Exit relay and the destination, enabling correlation attacks to deanonymize the client's connections [2,14,15,31].

To address the weaknesses of Tor and achieve a higher degree of anonymity, even in the presence of a Global Passive Adversary (GPA) that monitors all communication among entities [11], the concept of the mix network (mix-net) has been explored in numerous studies [7, 19, 21, 32]. Similar to Tor, a mix-net is an overlay network designed to enhance anonymity by routing client packets through multiple intermediary nodes, known as mix-nodes, a concept first introduced by Chaum [4], before forwarding them to their final destination. Unlike Tor, each intermediary node (mix-node) in a mix-net shuffles the traffic flow it receives, making the input packets unlinkable to the output [7]. This process guarantees that as long as at least one mix-node along the packet's path operates honestly, correlating input and output traffic becomes significantly challenging [6], effectively thwarting both Tor's adversaries and the GPA.

The shuffling of traffic, facilitated by mix-nodes as the core component of mix-nets, can be implemented using various methods. The first approach is *threshold mixes* [4], where packets are accumulated until a preset threshold is reached before forwarding. The second method uses *pool mixes* [9], which require both a threshold number of packets and a time limit before transmission. The third method, *stop-and-go mixes* [16], introduces random delays to incoming traffic based on an exponential distribution $\mathsf{Exp}(\mu)$, where the average delay for each packet is $\frac{1}{\mu}$. This stop-and-go approach provides flexibility over average delay control while leveraging the memoryless property of the exponential distribution

to enhance anonymity, making it a preferred choice in deployed mix-net like NYM[1] [7].

In addition to these methods, mix-net topologies can also be structured in various ways to strengthen anonymity [8]. One common configuration is the *cascade* topology, where mix-nodes are arranged into cascades, each consisting of L intermediary hops, and clients select one of these cascades for packet routing [3, 19, 32]. Another configuration, *Free Routes*, allows clients to randomly select L distinct nodes from all available mix-nodes to form a packet path [20]. A third configuration is the *stratified* topology, where mix-nodes are organized into L distinct layers, with a packet path created by selecting one mix-node from each layer. This stratified approach offers more diverse routing paths compared to the cascade topology and is more adaptable for handling cover traffic[2] compared to Free Routes, as implemented in systems like [7, 21]. In this paper, we focus on the stratified topology of mix-nets combined with stop-and-go mixes, commonly referred to as Loopix mix-nets [21]. Fig. 1 illustrates an example of such a mix-net with $L = 3$ layers, each containing two mix-nodes.

Problem Statement. Mix-nets, irrespective of their topology or the specific mix-nodes used, are prone to high end-to-end latency due to the requirement of routing packets through multiple intermediary hops, with additional delays imposed at each hop for shuffling. Unlike systems such as Tor, where all packets of a message travel through the same circuit (path), mix-nets handle each packet independently in terms of routing and shuffling. This independent process means that the overall end-to-end latency of a message—comprising multiple packets—is dictated by the packet that experiences the longest delay [7].

To clarify, consider the example depicted in Fig. 1, where a client sends a message divided into four packets through a mix-net. In this case, the destination must wait for all packets to arrive before the message can be fully reconstructed. The delays experienced by each packet can be further broken down into four key components: l_C, the delay between the client and the mix-net; l_r, the delay for packets traveling between mix-nodes; l_d, the delay for forwarding packets from the mix-net to the destination; and finally, l_s, the delay introduced by mix-nodes for shuffling.[3]

Design Goals. Given the high latency problem in mix-nets, our primary goal in this paper is to provide clients with the ability to manage the end-to-end latency of their messages while using the Loopix mix-net. Specifically, we assume the Loopix mix-net consists of L layers, with W mix-nodes per layer. When a client sends a message divided into n packets, they aim to achieve low latency for this message. To facilitate this, we propose methods that allow clients to adjust either the shuffling delay (l_s) or the routing delays (l_C, l_r and l_d), depending on the number of packets in the message. Our objective is to offer this flexibility without

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

[2] Cover (dummy) traffic refers to additional packets generated by clients that loop back to themselves, designed to mislead a GPA and enhance anonymity.

[3] We assume that cryptographic operation latency is negligible and queuing delays are not considered, as the network is assumed to have sufficient capacity.

significantly compromising anonymity, ensuring adversaries cannot exploit these adjustments to reduce privacy. Additionally, we ensure proper load balancing throughout the mix-net to prevent traffic bottlenecks.

Related Work. Although this paper introduces the first granular latency management framework for mix-nets, earlier efforts to optimize latency were primarily focused on the Tor network. One notable example is LasTor [1], which pioneered latency optimization by strategically routing packets within Tor. LasTor proposed partitioning the network into regions, each containing a few Tor relays. In such cases, clients would select the closest region based on a routing formula and then randomly choose relays within that region. While this approach was innovative for Tor, it is less suitable for mix-nets due to differing threat models. For instance, LasTor's partitioning strategy was designed to avoid AS-level adversaries, whereas mix-nets assume the presence of a GPA, making such partitioning unnecessary. Moreover, LasTor does not address load balancing, a key requirement in mix-nets. Additionally, its routing formula, which is suitable for long-term fixed-path routing, is less effective for mix-nets, where packets are independently routed through the network.

Another notable low-latency routing proposal for Tor is CLAPS [27], which uses linear programming to optimize an objective function, such as minimizing $l_c + l_r + l_d$, while mitigating AS-level correlation attacks. Although CLAPS could theoretically be adapted for mix-nets to reduce routing delays, its high computational complexity—on the order of $O(N^6)$, where N is the number of nodes—makes it impractical for our purposes, as also noted in LARMix [26].

Within the mix-net domain, LARMix [26] stands out as a key work focused on low-latency routing tailored to stratified topologies. LARMix introduces a diversification algorithm to ensure geographically diverse nodes are present at each layer, alongside a routing formula that prioritizes proximity and low-latency paths. While LARMix is relevant, it primarily addresses single-packet messages, which limits its applicability to multi-packet message latency management.[4]

In addition to low-latency routing strategies, studies such as [21] demonstrate that managing shuffling delays (l_s)—specifically by altering the average delay governed by the exponential distribution used by mix-nodes—can reduce latency. However, like other works, these methods focus on single-packet messages and do not address how latency can be managed without compromising anonymity. In contrast, our first latency management method proposes a similar but more comprehensive approach, allowing clients to manage the latency of messages with multiple packets while maintaining anonymity using cover traffic generated by clients, supported by differential privacy guarantees.

[4] CLAM [23] provides a similar framework to LARMix but is also restricted to single-packet routing without addressing load balancing, making it less suitable for our goals.

Additionally, [24] provides a low-latency routing approach for Free Routes topology in mix-nets, while [25] introduces technical analysis of the impact of biased routing on anonymity.

Contributions. To address the high end-to-end latency associated with forwarding messages consisting of n packets through a mix-net, this paper makes the following key contributions: **First**, we theoretically explore the end-to-end delay of such messages to establish foundational principles for latency management. Specifically, we identify that routing delays (l_c, l_r, and l_d) are only loosely dependent on the shuffling delay (l_s), allowing for independent analysis of these components. Therefore, we first theoretically analyze the shuffling delay introduced by a single stop-and-go mix and then extend this analysis to a full mix-net scenario involving L layers. Our analysis covers key aspects such as average latency, variance, and the α-percentile of the shuffling delay l_s, providing a detailed latency model, which, to the best of our knowledge, is the first of its kind in this granular scenario. Furthermore, we explain how routing delays generally affect the end-to-end latency, which can be additionally combined with the impact of shuffling delays.

Second, after quantifying the high end-to-end latency theoretically, we introduce our first latency management method, Managing Shuffling Delays (MSD), which focuses on controlling l_s. In this approach, clients can adjust the parameter μ in stop-and-go mixes to reduce the average shuffling delay. However, reducing shuffling delays may compromise anonymity. To mitigate this, we allow clients to generate noise, parameterized by $0 \leq \theta \leq 1$, where lower values of θ reduce cover traffic but also lower anonymity. We also demonstrate that this method maintains privacy guarantees under an (ϵ, δ)-differential privacy framework.

Third, we propose another method for latency management called Packet-Aware Routing (PAR), which includes two routing techniques based on the number of packets n in a message. These techniques prioritize faster paths for messages containing a higher number of packets. The first technique, Threshold-Based Routing (TBR), sets a latency threshold to ensure packets avoid high-latency links. The second technique, Weighted-Prioritized Routing (WPR), assigns higher probabilities to selecting low-latency hops, while still allowing other nodes to be selected with minimal probability. Although these routing strategies help reduce latency, they may lead to some mix-nodes receiving disproportionate traffic, creating bottlenecks. To address this, although we considered using the load-balancing technique from the state-of-the-art LARMix [26], we found their method inefficient, imposing a computational complexity of $O(N^4)$, where $N = L \cdot W$. To provide an efficient solution, we propose a novel load-balancing algorithm with a computational complexity of $O(N^2)$, ensuring even traffic distribution without compromising packet-aware routing characteristics.

Finally, we implemented a prototype of our proposed methods in *Python*, consisting of approximately 5000 LOC. This implementation includes both our theoretical bounds and proposed latency management strategies and utilizes the discrete event simulator *SimPy* [22] to simulate a Loopix-like mix-net. In our experiments, link delays between nodes were modeled based on the RIPE Atlas dataset [30], using RTT/2 as the baseline delay.[5]

[5] See Appendix B for details on how bidirectional paths in the RIPE dataset have negligible impact on measured latency.

Our evaluation consists of two components: analytical and simulation-based. In the analytical experiments, we configured various mix-net instances and, to quantify anonymity, we measured the predictability of client paths to the mix-net exit using Shannon entropy [28].[6] We also measured the average routing delay ($l_c + l_r + l_d$) and the average shuffling delay (l_s) based on our theoretical analysis. Additionally, the simulation-based evaluation, which also aimed to quantify anonymity, focused on assessing the Shannon entropy of packets targeted by a GPA, quantifying the difficulty for adversaries to correlate outgoing packets with their respective inputs. We also measured the end-to-end delay $l_{e2e} = l_c + l_r + l_s + l_d$, representing the time taken for the slowest packet in a client's message to reach its destination.

Our evaluations confirmed the accuracy of our theoretical latency bounds, showing a significant increase in latency as n (the number of packets) increases. Moreover, we demonstrated that setting the noise parameter $\theta = 0.4$ in MSD can reduce end-to-end latency by at least 50% without significantly compromising anonymity. Conversely, applying TBR or WPR resulted in at least a 20% latency reduction while compromising less than one bit of anonymity. Additionally, our load-balancing algorithm effectively ensured even traffic distribution across mix-nodes, preserving the packet-aware routing characteristics.

Finally, we assessed the potential adversarial gains with the PAR methods. We examined scenarios where adversaries control a subset of mix-nodes and collaborate with a GPA to deanonymize client communications. We explored both random and greedy corruption scenarios, where adversaries strategically control mix-nodes to maximize their advantage, measured by the fraction of fully corrupted paths (FCP). Our results reveal a balanced trade-off between low latency and anonymity, with the probability of full path corruption remaining negligible in realistic scenarios.

2 Methodology

In this section, we begin by laying the groundwork with a theoretical analysis of the latency encountered when forwarding n packets that make up a message. Based on these insights, we further introduce two latency management methods designed to enhance the practicality of mix-nets.

2.1 Theoretical Analysis

This section examines the total latency affected by multi-hop routing and intentional shuffling delays on the delivery of n packets that constitute a message to its destination. In this context, the total end-to-end delay, l_{e2e}, consists of both routing delays ($l_c + l_r + l_d$) and shuffling delays (l_s), with l_{e2e} ultimately determined by the packet with the longest overall delay. Specifically, routing delays are governed by the probability distribution that dictates the forwarding of

[6] k bits of Shannon entropy imply 2^k possible packet exit nodes.

packets from one node to the next. To clarify, consider a stratified mix-net with L layers, each containing W mix-nodes. Let layer 0 represent the clients and layer $L+1$ represent the destinations. We define r_{ij}^k as the probability of routing a packet from node i in layer $k-1$ to node j in layer k, where $1 \leq i, j \leq W$ and $1 \leq k \leq L+1$. In this setup, the routing strategy is captured by the routing matrix $\mathcal{R}^k = \left[r_{ij}^k\right]$, which specifies the probabilities of selecting paths between layers, leading to the overall latency caused by routing.

On the other hand, stop-and-go mixes introduce random shuffling delays for each packet, modeled using an exponential distribution $\mathsf{Exp}(\mu)$. These delays are sampled independently for each packet, ensuring that routing strategies and shuffling delays operate independently. This suggests that the routing matrix $\mathcal{R}^k$ and, consequently, routing delays are independent of the shuffling delays l_s. Given this separation, we can assess the effects of both independently on l_{e2e}, before examining their combined impact.

Shuffling Delay Effects for a Single Mix-Node. We begin by analyzing the effect of shuffling delays when a client sends n packets to a single mix-node to deliver a message. In this scenario, our analysis focuses on the average latency, variance, and the α percentile of latency induced by the mixing process within the mix-node. We first start with the average latency. To analyze this, we assume that the client dispatches all packets simultaneously to the mix-node.[7] Upon their arrival at the mix-node, each packet i, where $1 \leq i \leq n$, is subjected to a random delay X_i, following an exponential distribution $\mathsf{Exp}(\mu)$, before being flushed from the mix-node.

To formalize the event of the last packet exiting the mix-node out of n packets, we define the event Y as the completion time for flushing all n packets from the mix-node. We then derive the probability of all messages being flushed out of the mix-node before a specific time t as $\mathbb{P}(Y \leq t) = \prod_{i=1}^{n} \mathbb{P}(X_i \leq t)$. For an exponential distribution X with parameter μ, the cumulative distribution function (CDF) is expressed as $\mathbb{P}(X \leq t) = 1 - e^{-\mu t}$, enabling us to articulate the probability $\mathbb{P}(Y \leq t)$ as depicted in Eq. (1).

Eq. (1) represents the CDF of the random variable Y. To derive the average latency (or $\mathbb{E}[Y]$), it is often more insightful to work with the probability density function (PDF), which elucidates the distribution characteristics of Y. Transitioning from the CDF to the PDF requires differentiation with respect to t, as elaborated in Eq. (2).

$$\mathbb{P}(Y \leq t) = \left(1 - e^{-\mu t}\right)^n. \tag{1}$$

$$f_Y(t) = \frac{d}{dt}\mathbb{P}(Y \leq t) = \mu n e^{-\mu t}\left(1 - e^{-\mu t}\right)^{n-1}. \tag{2}$$

Utilizing the PDF $f_Y(t)$, we proceed with the computation of the average latency by integrating over time as outlined in Eq (3) to (6). Eq. (4) utilizes the binomial theorem for the expansion, while Eq. (5) incorporates integration by parts for terms of the form $\int te^{at}\,dt$, with the solution shown in Eq. (6).

[7] This assumption simplifies the analysis and helps derive a lower bound for the latency.

$$\mathbb{E}[Y] = \int_0^\infty t f_Y(t)\,\mathrm{d}t = \int_0^\infty \mu n t e^{-\mu t}\left(1-e^{-\mu t}\right)^{n-1}\mathrm{d}t, \tag{3}$$

$$= \int_0^\infty \mu n t e^{-\mu t}\left(\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^{n-i-1}e^{-\mu(n-1-i)t}\right)\mathrm{d}t, \tag{4}$$

$$= -\mu n\left(\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^{n-i}\int_0^\infty t e^{-\mu(n-i)t}\mathrm{d}t\right),$$

$$= -\mu n\left(\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^{n-i}\left[\frac{te^{-\mu(n-i)t}}{-\mu(n-i)} - \frac{e^{-\mu(n-i)t}}{(\mu(n-i))^2}\Big|_0^\infty\right]\right), \tag{5}$$

$$= -\frac{n}{\mu}\left(\sum_{i=0}^{n-1}\binom{n-1}{i}\frac{(-1)^{n-i}}{(n-i)^2}\right). \tag{6}$$

Eq. (6) precisely represents the average shuffling latency (l_s). Using Theorem 1, we simplify Eq. (6) to express l_s as $\frac{H_n}{\mu}$. This implies that average latency experienced by the destination is inversely proportional to μ, the parameter denoting the rate of the exponential distribution. Interestingly, as the number of packets n increases, the latency proportionally escalates. For instance, with $n = 1$, the average latency is directly equivalent to the mean of the exponential distribution. As n increases to 2, the latency rises to $\frac{3}{2}$ times the exponential mean, showcasing a consistent trend of increasing latency with more packets.

Theorem 1. *The average latency ($\mathbb{E}(Y)$) derived in Eq. (6) for n packets delivery, considering H_n as the harmonic number up to n, is given by:*

$$\mathbb{E}[Y] = \frac{H_n}{\mu}.$$

Proof. *See Appendix C.*

To further analyze the variance of latency for n packets departing from the mix-node, we adopt a systematic approach similar to that used for determining $\mathbb{E}[Y]$. Initially, the calculation of $\mathbb{E}[Y^2]$ is undertaken, as elaborated in theorem 2. Once $\mathbb{E}[Y^2]$ is computed, the variance $\mathrm{Var}(Y)$ can be derived using the formula: $\mathrm{Var}[Y] = \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2$.

Theorem 2. *The second moment of latency $\mathbb{E}(Y^2)$ for n packets is given by:*

$$\mathbb{E}[Y^2] = \frac{2}{\mu^2}\left(\sum_{i=1}^{n}\binom{n}{i}\frac{(-1)^i}{i^2}\right).$$

Proof. *See Appendix C.*

In some scenarios, our focus shifts from controlling the average or variance of latency to managing the α percentile of latency. The α percentile, denoted by t_α,

implies that in α percent of instances, the latency experienced will not surpass t_α. This notion prompts the question: What is the value of t_α in the equation $\mathbb{P}(Y \leq t_\alpha) = \alpha$? Referring to Eq.(1), we represent this as $(1 - e^{-\mu t_\alpha})^n = \alpha$. Simplifying further, we find $1 - e^{-\mu t_\alpha} = \alpha^{\frac{1}{n}}$, leading to $e^{-\mu t_\alpha} = 1 - \alpha^{\frac{1}{n}}$. Thus, we deduce t_α as shown in Eq. (7):

$$t_\alpha = \frac{\log\left(\frac{1}{1-\alpha^{\frac{1}{n}}}\right)}{\mu}. \tag{7}$$

In Eq. (7), given the bounds $0 \leq \alpha \leq 1$, we observe that $0 \leq \alpha^{\frac{1}{n}} \leq 1$. As the number of packets n increases, $\frac{1}{n}$ decreases, causing $\alpha^{\frac{1}{n}}$ to rise. This increase in $\alpha^{\frac{1}{n}}$ means $1 - \alpha^{\frac{1}{n}}$ moves closer to zero, thus enlarging the value of $\log\left(\frac{1}{1-\alpha^{\frac{1}{n}}}\right)$, and consequently, increasing t_α. This behavior is expected as more packets typically imply greater latency. Moreover, the inverse relationship between t_α and μ suggests that enhancing the average mixing delay ($\frac{1}{\mu}$) of a mixnode proportionately raises the α percentile. This relationship is critical for understanding how adjustments in mixing delay can influence the overall latency distribution, especially in scenarios demanding strict latency controls.

Shuffling Delay Effects for a Mix-Net. Moreover, we extend our analysis to scenarios where a message must traverse through L mix-node hops in a mix-net. In this case, each packet, indexed from 1 to n, experiences a random delay X_i, representing the total time taken for each packet to pass through the mix-net, from entry to exit.

Since there are L mix-nodes, each introducing a delay modeled by an exponential distribution $\mathsf{Exp}(\mu)$, X_i aggregates L independent and identically distributed exponential variables, forming an Erlang distribution. This Erlang distribution is characterized by parameters L and μ, and its probability density function (PDF) is given by:

$$f_{X_i}(t) = \frac{t^{L-1}\mu^L e^{-\mu t}}{(L-1)!}.$$

This distribution allows us to model the total waiting time required for all n packets to be received at the destination. The probability of interest is $\mathbb{P}(Y = \max(X_1, X_2, \ldots, X_n) \leq t)$, where Y denotes the maximum time among all packets exiting the mix-net. This can be expressed as the product of the probabilities for each packet:

$$\mathbb{P}(Y \leq t) = \prod_{i=1}^{n} \mathbb{P}(X_i \leq t).$$

Using the CDF of the Erlang distribution, $\frac{\gamma(L,\mu t)}{(L-1)!}$, where $\gamma(L, \mu t)$ is the lower incomplete gamma function, we express the CDF of Y as:

$$\mathbb{P}(Y \leq t) = \left(\frac{\gamma(L, \mu t)}{(L-1)!}\right)^n. \tag{8}$$

Here, $\gamma(L, \mu t) = \int_0^{\mu t} x^{L-1} e^{-x} \, \mathrm{d}x$, and its derivative with respect to t is $\frac{\mathrm{d}}{\mathrm{d}t}\gamma(L, \mu t) = \mu^L t^{L-1} e^{-\mu t}$, which leads to the PDF of Y:

$$f_Y(y) = n\mu^L \frac{t^{L-1} e^{-\mu t}}{(L-1)!} \left[\frac{\gamma(L, \mu t)}{(L-1)!}\right]^{n-1}.$$

Next, the expectation $\mathbb{E}(Y)$ can be computed using Eq. (9).

$$\mathbb{E}(Y) = \int_0^\infty t f_Y(t) \, \mathrm{d}t = \int_0^\infty n\mu^L \frac{t^L e^{-\mu t}}{(L-1)!} \left[\frac{\gamma(L, \mu t)}{(L-1)!}\right]^{n-1} \mathrm{d}t. \tag{9}$$

Further analysis in Appendix C demonstrates that as L and n increase, the average shuffling delay l_s also increases, while increasing μ decreases the overall shuffling latency.

Similarly, the variance of latency is calculated using $\mathbb{E}[Y^2(t)] - \mathbb{E}^2[Y(t)]$, utilizing the PDF of Y. Additionally, within the mix-net scenario, the α-percentile of shuffling latency latency can be derived through Eq. (10).

$$t_\alpha = \frac{\gamma_L^{-1}((L-1)!\alpha^{\frac{1}{n}})}{\mu}. \tag{10}$$

Here, γ_L^{-1} represents the inverse of the gamma function $\gamma(L, \mu t)$, yielding the value of μt_α. Similarly, Eq. (10) demonstrates that as L and n increase, the average shuffling delay l_s also increases. Conversely, increasing μ results in a decrease in latency.[8]

Routing Delays Effects. Finally, we briefly analyze the effects of routing delays on latency when forwarding n packets to deliver a message to its destination. These delays are influenced not only by the routing matrix $\mathcal{R}^k$ but also by the distribution of delays within the network. As a result, deriving a concrete analysis of these delays is challenging, as they can vary significantly from case to case. Instead, we provide insights into how routing delays affect latency and how they can be controlled. For example, when routing one packet from layer $k-1$ to layer k using $\mathcal{R}^k$, the routing dealy can be kept below an acceptable threshold with probability $1-\beta$. However, with probability β, the latency may exceed this threshold, resulting in higher delays. When routing n packets, the cumulative probability of encountering high latency becomes $1-(1-\beta)^n$, meaning that as n increases, the likelihood of experiencing high latency grows rapidly.

To manage routing delays effectively, especially given the variability of link delays, the optimal approach is to avoid distant nodes that contribute disproportionately to overall latency. Importantly, as we demonstrate in §2.3, this strategy does not significantly compromise anonymity, making it feasible to strategically route packets while avoiding high-latency links. This contrasts with shuffling delays l_s, where directly reducing these delays compromises anonymity (as discussed in §2.2).

[8] Refer to comprehensive analysis in Appendix C.

2.2 Managing Shuffling Delay (MSD)

One crucial element in mix-nets that provides anonymity, while potentially increasing latency, is the mixing process controlled by the parameter μ, which governs the exponential delay at each mix-node. Specifically, μ inversely affects the shuffling delay l_s, meaning that increasing μ allows packets to exit mix-nodes more quickly, thereby reducing l_s. With this in mind, we propose the first method to manage high latency by controlling μ. In this approach, a client wishing to reduce latency can adjust μ to $\mu' = a\mu$, where $a \geq 1$. This adjustment directly reduces latency, as derived from our theoretical bounds. However, reducing shuffling delays means packets spend less time in each mix-node, decreasing their likelihood of being mixed with other packets, which reduces anonymity.

To address this, we propose a framework to enhance privacy by generating cover traffic. We assume the incoming traffic rate produced by the client to the mix-net follows a Poisson distribution (as assumed in Loopix [21] and NYM [7]) with parameter λ. If the parameter of the exponential distribution is μ, each mix-net layer's input-to-output traffic rate can be expressed as $\frac{\lambda}{\mu}$. Since the number of shuffled messages at each mix-node follows a Poisson distribution, adjusting μ to $\mu' = a\mu$ modifies this ratio to $\frac{\lambda}{\mu'}$. Given that $a \geq 1$, this results in a reduction in the traffic mixing ratio, which diminishes anonymity.

To counterbalance this reduction in anonymity, we propose that clients increase their generated outgoing traffic rate by adding noise, adjusting λ to $\lambda' = \left[\frac{a}{(1-a)\theta+a}\right]\lambda$, where $0 \leq \theta \leq 1$ is a cost parameter for generating additional traffic. When $\theta = 0$, the cost is minimal ($\lambda' = \lambda$), and when $\theta = 1$, the cost is maximal ($\lambda' = a\lambda$), preserving the ratio of incoming to outgoing traffic despite client manipulation of μ. In this case, anonymity is not reduced. However, for intermediate values of θ, there is a trade-off between cost and anonymity, allowing clients to balance these factors.

Furthermore, adding this cover traffic in the form of a Poisson distribution provides an (ϵ, δ)-differential privacy guarantee. To illustrate this, consider a dataset $\mathcal{X}$ representing the clients' traffic distribution across mix-nodes, such as $[0, 1, 0, 0]$, indicating that the clients' traffic is forwarded only to the second mix-node out of four available in the first layer. Suppose $\mathcal{Y}(\mathcal{X})$ outputs the total rate of packets received by the mix-nodes. This dataset is more vulnerable to attacks if faster shuffling is used, as packets are less mixed. To enhance privacy, clients add cover traffic, modeled as the mechanism $\mathcal{M}$, which outputs $\mathcal{M}(\mathcal{Y}) = \mathcal{Y} + \mathcal{N}(\lambda', \mathcal{X})$, where $\mathcal{N}(\lambda', \mathcal{X})$ represents the noise generated by the clients.

Now, consider two adjacent datasets $\mathcal{X}$ and $\mathcal{X}'$, which differ by only one row. The following theorem shows that $\mathcal{M}(\mathcal{Y}(\mathcal{X}))$ provides (ϵ, δ)-differential privacy for such adjacent datasets.

Theorem 3. *A mechanism that adds Poisson noise with parameter λ' to the output $\mathcal{Y}(\mathcal{X})$ is (ϵ, δ)-differentially private with respect to the adjacent dataset*

$\mathcal{X}'$, which differs by one row in $\mathcal{X}$, with a sensitivity of 1, where:

$$\epsilon = \ln\left(1 + c\frac{\sqrt{\lambda'_i}}{\lambda'_i}\right), \quad \delta = \frac{e^{-\lambda'_i}(e\lambda'_i)^{\lambda'_i + c\sqrt{\lambda'_i}}}{(\lambda'_i + c\sqrt{\lambda'_i})^{\lambda'_i + c\sqrt{\lambda'_i}}}.$$

(Here, $c > 0$ allows trading higher ϵ for lower δ). ***Proof.*** *See Appendix F.*

In Theorem 3, λ'_i corresponds to the amount of traffic directed to mix-node i in the first layer, where $\lambda'_i = \frac{\lambda'}{W}$ when the routing is uniform. This theorem shows that δ is inversely related to λ'_i, indicating that adding more noise results in a smaller δ, thus providing stronger privacy guarantees under (ϵ, δ)-differential privacy. Additionally, the cost parameter θ can be tuned to modify λ'—higher θ values lead to more cover traffic and lower δ, translating directly into stronger privacy guarantees.

2.3 Packet-Aware Routings (PAR)

Apart from reducing l_s to decrease l_{e2e}, another potential for latency reduction lies in routing delays ($l_c + l_r + l_d$). In this section, we provide two routing algorithms that take into account the number of packets n required to transmit a message. Based on this n, the routing of packets is biased. For instance, for larger n, all packets' paths will be biased toward very low-latency links, ensuring that routing delays have minimal impact on l_{e2e}. In contrast, when n is smaller, it is acceptable to route through more diverse paths. We also note that this bias in routing, unlike reducing l_s, will not directly affect anonymity. This means that even if packets are forwarded to low-latency paths, they still enjoy sufficient anonymity.

Threshold-Based Routing (TBR). The first PAR method to prioritize low-latency mix-nodes along the packet route is TBR, described in Eq. (11). Here, r^k_{ij}, the routing probability from node i in layer $k-1$ to node j in layer k, is defined based on l^k_{ij}, their corresponding link latency. We define $\mathcal{I}^k_{ij}$ as the ranking function that shows the closeness of node j in layer k to node i in layer $k-1$, where an output of 0 indicates the closest node, and $W-1$ indicates the farthest node. In this case, if the ranking function is less than $\lceil \frac{W}{H_n} \rceil$, mix-node j can be selected with a probability inversely proportional to its latency, powered by $\frac{H_n - 1}{H_n}$. This power increases as n grows, meaning that the higher the number of packets, the higher the probability of selecting the closest nodes. The probabilities are normalized over the set of mix-nodes $\mathcal{S}_k$ within the threshold.

$$r^k_{ij} = \begin{cases} \dfrac{\left(\frac{1}{l^k_{ij}}\right)^{\frac{H_n-1}{H_n}}}{\sum_{j \in \mathcal{S}_k}\left(\frac{1}{l^k_{ij}}\right)^{\frac{H_n-1}{H_n}}}, & \text{if } \mathcal{I}^k_{ij} \leq \lceil \frac{W}{H_n} \rceil, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Weighted-Prioritized Routing (WPR). The second PAR method, WPR, unlike TBR, does not entirely exclude the farthest nodes but selects them with

negligible probability, as described in Eq. (12), showing r_{ij}^k for choosing mix-node j in layer k. Similar to TBR, WPR prioritizes mix-nodes based on the inverse of their latency to the power of $\frac{H_n-1}{H_n}$. Additionally, it introduces weights in the form of $\left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k \cdot H_n}$, which, for higher numbers of packets n, strongly favors the closest node. For such nodes, $\left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k \cdot H_n} = 1$, while for farther nodes, as H_n increases, the selection probability diminishes significantly, promoting faster nodes. In other words, WPR prioritizes low-latency links with overwhelming probability and assigns negligible probability to farther nodes.

$$r_{ij}^k = \frac{\left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k \cdot H_n} \cdot \left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}{\sum_{j=1}^{W} \left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k \cdot H_n} \cdot \left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}} \tag{12}$$

In addition, Theorem 4 explains that the KL divergence between the probability distributions derived from both TBR and WPR is upper bounded by $\left(\frac{(W-1)H_n}{2}\right)\log(H_n)$, showing that as the number of packets constituting a message increases, the distance between these two probability distributions also increases.

Theorem 4. *The KL divergence distance between the probability distribution of routing of TBR and WPR is upper bounded by* $\left(\frac{(W-1)H_n}{2}\right)\log(H_n)$.

Proof. *See Appendix D.*

Balancing the Loads. After applying the PAR routing schemes, some mix-nodes may experience higher traffic loads compared to what they would under a uniform routing policy. If left unaddressed, this imbalance can lead to delays in packet processing and may also provide adversaries with the opportunity to exploit the network by deploying high-capacity mix-nodes. Such nodes could handle large amounts of traffic, increasing the risk of deanonymization of clients.[9] To address this, we propose a load-balancing algorithm (Algorithm 1) that first measures the total traffic assigned to each column (representing the mix-node in layer k) in $\mathcal{R}^k$, the routing matrix. If a column exceeds a load of 1, it is labeled as *overloaded*. Conversely, if a column's load falls below 1, it is considered *underloaded*. Columns with exactly 1 are classified as *balanced*.

Once this categorization is completed, the algorithm treats the overloaded column entries by eliminating the excess load from them. For example, if the excess load is ℓ, it is removed from overloaded columns by subtracting it from their W' entries, with each entry receiving a reduction of $\frac{\ell}{W'}$, where W' represents the number of entries in those columns capable of reducing their load.[10] The

[9] By correlating messages through traffic analysis, similar to Guard replacement attacks in the Tor network [33].

[10] Typically, $W' = W$, unless some entries in the overloaded columns have no room for further load reduction, in which case no further reduction is possible for those entries.

algorithm then calculates the load deficit for underloaded columns. For these columns, the additional load required to achieve balance is computed as ℓ'. The redistribution process divides the load ℓ' across the entries of the underloaded columns that need additional load, proportionally to the amount of available room in the corresponding rows of each entry, ensuring that the rows still fulfill valid probability distributions. This process ensures that the load is balanced across all mix-nodes, with the balancing process completing in $O(W^2) \approx O(N^2)$, where $N = L \cdot W$. This approach is significantly faster than the $O(N^4)$ time complexity required by LARMix's load-balancing algorithm [26]. A toy example of our algorithm can be found in Appendix E.3, along with a comparison between our load-balancing method and that of LARMix.

3 Evaluation

In this section, we evaluate our theoretical analysis as well as the proposed methods for reducing latency in mix-nets.

Experimental Setup. Before delving into the experimental results, we first outline the parameters chosen for our experiments. We consider $W = 128$ mix-nodes per layer and $L = 3$ layers in the mix-net. To model the link delay between nodes in the network, we utilize the RIPE Atlas dataset [30], as its globally distributed endpoints offer a representative model of real-world latencies.

We simulate the mix-net using a discrete-event simulator within the *SimPy* environment in Python, where mix-nodes emulate the mixing process. Each mix-node uses stop-and-go mixing [17], sending out packets after a delay sampled from $\mathsf{Exp}\left(\frac{1}{20}\right)$, equivalent to an average shuffling delay of 50 ms.[11] The incoming rate of traffic λ is set to 20,000 packets per second. For all experiments, we perform at least 400 iterations, with each iteration involving a new mix-net configuration to minimize the likelihood of outliers or faulty results.

Evaluation Metrics. Our first set of metrics are analytical. To analytically evaluate latency, we compute the separate effects of shuffling delay l_s and routing delays $(l_c + l_r + l_d)$, and then combine these components to assess the overall end-to-end delay l_{e2e}. For anonymity evaluation, we use Shannon entropy [28], denoted as $\mathsf{H}(r)$, to quantify the anonymity provided by the routing strategy. This is defined as the difficulty an adversary faces when determining the exit node of a packet given its first mix-node in the path.[12]

In contrast, simulation-based metrics are derived directly from simulated scenarios. To measure latency, we record the time taken for messages to traverse the entire mix-net from entry to exit, denoted as l_{e2e}. For anonymity in simulation, we examine a scenario where a GPA monitors client communication patterns and

[11] This value is in line with real-world deployments of mix-nets like NYM [7].

[12] To derive $\mathsf{H}(r)$, first, the routing matrices for all layers $\mathcal{R}^k$ should be derived based on the routing strategies. Next, we multiply all routing matrices $\mathcal{R}^k$ across layers, resulting in a transition matrix that maps a client's packet from the first node in the first layer to the exit node (mix-node of the last layer). In this matrix, the entropy for each row is then calculated. The average entropy of all rows is considered as $\mathsf{H}(r)$.

Table 1: Key results of evaluating PARSAN-Mix.

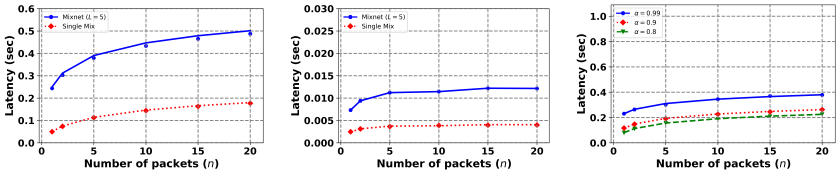
Approaches	MSD		PAR (TBR)		PAR (WPR)	
	Without noise	With noise	Imbalanced	Balanced	Imbalanced	Balanced
Latency reduction	52%	52%	20%	18%	20%	18%
Anonymity achievement	-4.1 bits	-1.9 bits	+1.7 bits	+3.5 bits	+0.3 bits	+3 bits

targets specific packets as they enter the mix-net. The GPA's goal is to correlate these incoming packets with outgoing ones, estimating the probability distribution of the targeted packets among the outgoing traffic. To quantify the level of anonymity in this case, we use the entropy of this distribution, denoted as $\mathsf{H}(p)$, which reflects the uncertainty the adversary faces in correctly mapping incoming to outgoing packets. This measure of anonymity follows the methodology established in [5, 9, 10].

Key Results. Before delving into the details of the experiments, we begin by summarizing the main results achieved. We conducted three sets of experiments. The first set validates the theoretical results for latency statistics, with findings supported by simulation methods that confirm the accuracy of our theoretical models. Additionally, we conducted two further sets of experiments focusing on the MSD and PAR methods, respectively. The key outcomes are summarized in Tab. 1.

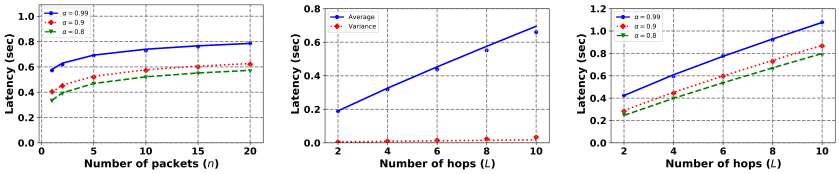
The table presents results for MSD with parameters $a = 5$, $n = 50$, both with and without cover traffic (in the former case, with $\theta = 0.4$). Specifically, MSD achieves a latency reduction of 52% compared to the vanilla (baseline mix-net) setting, with anonymity decreasing by 4.1 bits. This anonymity reduction can be mitigated to 1.9 bits by generating cover traffic. Additionally, Tab. 1 details the main results of TBR and WPR routing in PAR with $n = 50$. In both cases, applying PAR with load balancing (18%) and without load balancing (20%) reduces latency for TBR and WPR. Unlike MSD, PAR also provides a slight increase in packet entropy, $\mathsf{H}(p)$, highlighting the importance of directing more traffic to specific sets of nodes to enhance packet mixing. The results show up to a 3.5-bit and 3-bit increase in anonymity when using TBR and WPR without load balancing, and a 1.7-bit and 0.3-bit increase with load balancing, respectively, underscoring the benefits of PAR strategies.

Validating Theoretical Bounds. In this section, through Fig. 2 and Fig. 3, we validate the consistency of the theoretical results derived for shuffling delay (l_s) using both analytical and simulation experiments. It is important to note that throughout this section, and in all subsequent sections, we depict analytical results as lines and simulation results as scatter points. Fig. 2a illustrates the average latency of l_s for both a single mix-node and a mix-net scenario with $L = 5$ layers. While it may seem that the average shuffling delay of $\frac{1}{\mu} = 50$ ms should result in average latencies of 50 ms for a single mix-node and 250 ms for a mix-net (50 ms $\times$ 5), Fig. 2a shows a significant increase in average delay as n increases, reaching up to 4 times the naive expectation for a single mix-node and double for a mix-net at $n = 20$. More accurately, the latency spike from $n = 1$



(a) Average of l_s vs. n. (b) Variance of l_s vs. n. (c) α-% of l_s vs. n.

Fig. 2: Single Mix and Mix-Net Results: This figure illustrates the average and variance of latency in the case of a single mix and a mix-net scenario. It also presents the α percentile of latency for a single mix.



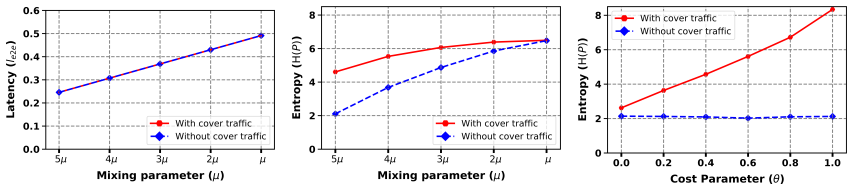
(a) α-% of l_s vs. n. (b) Statistics l_s vs. L. (c) α-% of l_s vs. L.

Fig. 3: Mix-Net Results: This figure illustrates the average and variance of latency, as well as the α percentile, for a mix-net scenario when n or L is varied.

to $n = 20$ in a single mix-node scenario is more pronounced than in a mix-net with $L = 5$ hops. However, an encouraging observation from this figure is the logarithmic nature of the expected latency increase, indicating that beyond a certain point, adding more packets does not significantly affect latency.

Fig. 2b, on the other hand, presents the variance in message latency as the number of packets changes. In both the single mix-node scenario and the mix-net scenario with $L = 5$ layers, the variance increases only slightly as the number of packets grows. This minimal variance change suggests that the overall variance is equivalent to the maximum variance among the n packets. Thus, a larger packet count naturally leads to increased variance.

Fig. 2c and Fig. 3a, additionally, depict the 80th, 90th, and 99th percentiles of latency for packets sent to a single mix-node and a mix-net, respectively. These figures demonstrate that an increase in packet count leads to a rise in the α-percentile of latency, meaning that more packets require a longer waiting time for the destination to fully receive the message. Furthermore, an increase in the α parameter elevates the percentile latency in both scenarios. Notably, the disparity at the 99th percentile underscores the necessity of planning for the longest wait times to ensure reliability, indicating outlier cases that significantly increase latency. Moreover, for $n = 1$, the 99th percentile latency for a single mix-node and a mix-net is 5 and 2.5 times the average latency seen in similar scenarios, respectively. This highlights that although stop-and-go mix mechanisms



(a) l_{e2e} vs. μ (b) $\mathsf{H}(P)$ vs. μ. (c) $\mathsf{H}(P)$ vs. θ.

Fig. 4: MSD Results: This figure shows the results of the latency optimization as well as the anonymity achieved using the MSD approach.

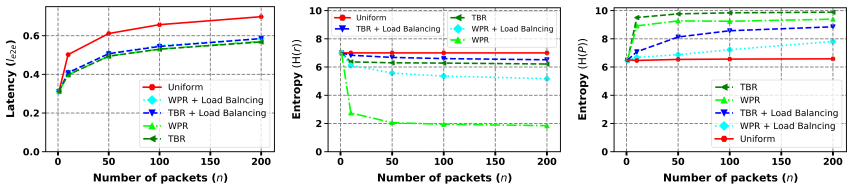
standardize an average latency, the 99th percentile can be exceedingly high—a critical design consideration to ensure manageable maximum latency.

On the other hand, to specify the effect of the number of layers L, we contemplate a setup where a client employs a mix-net for the anonymization of their communications, sending $n = 5$ packets per message. The rationale behind integrating additional hops is to enhance the intermingling of packets from different clients, thereby improving anonymity, albeit at the cost of increased latency. While an initial assumption may suggest that adding an extra hop would increase the average latency of l_s by an additional $\frac{1}{\mu}$, Fig. 3b reveals that the surge in average latency mirrors the increase seen in a single mix-node instance. This indicates that the addition of hops substantially impacts the total time required for the complete message to be received, with a marked increase in latency of l_s for each extra hop.

Fig. 3c, additionally, illustrates the 99th, 90th, and 80th percentiles of latency as the number of hops (leyers) increases, keeping $n = 5$ and $\frac{1}{\mu} = 50$ ms constant. As with earlier findings, there is a linear relationship between the hop count and the α-percentile. The significant difference between the 99th percentile and the lower percentiles highlights the importance of accounting for worst-case scenarios in system design, as the 99th percentile latency increases rapidly with additional hops. For instance, considering a 50 ms mixing delay, sending $n = 5$ packets can result in a latency exceeding 0.5 seconds when the layer count reaches $L = 4$.[13]

Managing shuffling Delay Evaluation. We further present the experimental results depicted in Fig. 4 to assess the effectiveness of the MSD framework in managing l_{e2e} by tuning the parameter μ of the $\mathsf{Exp}(\mu)$ distribution, with and without adding cover traffic. Specifically, Fig. 4a illustrates l_{e2e} derived from both analytical and simulation analyses, represented by lines and scatter points, respectively. Here, we set $\mu = 20$ and explore how latency changes by adjusting the scaling parameter $1 \leq a \leq 5$, while keeping the routing within the mix-net uniformly random. In this case, Fig. 4a suggests that increasing a leads to a reduction in latency. For instance, by adjusting a by unit increments, we observe an almost 13% reduction in average latency, corresponding to an approximate

[13] See additional results in Appendix G.



(a) l_{e2e} vs. n (b) $\mathsf{H}(r)$ vs. n. (c) $\mathsf{H}(P)$ vs. n.

Fig. 5: PAR Results: This figure illustrates the results of latency optimization and the anonymity achieved using the PAR approaches.

25 ms reduction in l_{e2e} when $a = 5$. On the other hand, when we apply cover traffic with a cost parameter $\theta = 0.4$, the latency reduction remains consistent.

Moreover, Fig. 4b illustrates the entropy of packets derived from simulations, with and without adding cover traffic. In the case where $\theta = 0.4$ and $1 \leq a \leq 5$, the figure demonstrates that without the addition of noise, entropy significantly decreases as a increases. Specifically, entropy drops by more than 4 bits, from 6.5 bits to around 2 bits when $a = 5$. However, when cover traffic is introduced with $\theta = 0.4$, this reduction in entropy is mitigated, with entropy values staying close to 5 bits even as a increases $a = 5$. This result highlights the effectiveness of adding cover traffic in maintaining higher levels of anonymity.

Finally, Fig. 4c presents the entropy of packets when $a = 5$ and adding cover traffic with the cost parameter θ ranging from 0 to 1. In this scenario, increasing θ results in higher entropy as the rate of generated noise increases, providing the highest entropy when $\theta = 1$. Interestingly, even when $\theta = 0.4$, the entropy increases significantly compared to the case without cover traffic, showing that even a moderate level of noise can effectively enhance anonymity. This illustrates a clear trade-off between the cost parameter θ and the resulting anonymity using our proposed MSD approach.

Packet-Aware Routings Evaluation. In this part of the experiments, we examine our proposed PAR schemes, where we bias the routing matrix towards low-latency paths based on the number of packets n constituting a message. Specifically, we consider both TBR and WPR methods, with and without applying the load-balancing algorithm, as well as a uniform routing baseline. The results of these experiments are shown in Fig. 5. More accurately, Fig. 5a depicts both the theoretical and simulation-based l_{e2e}, represented by lines and scatter points, respectively, as a function of the number of packets n. In line with our theoretical analysis, increasing the number of packets n increases l_{e2e}. Our PAR schemes, with or without load balancing, match the l_{e2e} of random routing at $n = 1$. However, beyond $n = 1$, the PAR schemes handle latency better by prioritizing lower-latency paths, resulting in at least a 20% latency reduction across all PAR schemes, whether load balancing is applied or not. This demonstrates the effectiveness of our load-balancing approach, which preserves the characteristics of the PAR schemes. Without applying load balancing, latency is slightly lower, but both TBR and WPR exhibit similar latency patterns.

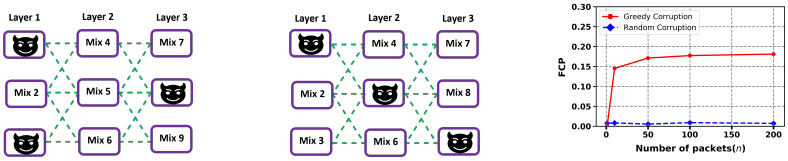
On the other hand, Fig. 5b shows the routing entropy when applying the PAR schemes or uniform routing. As expected, uniform routing provides the highest entropy of 7 bits for all values of n, indicating maximum unpredictability in routing. For all PAR schemes, with or without load balancing, the entropy decreases as n increases. This is because, as the number of packets increases, paths are biased towards lower-latency routes, making the routing more predictable and reducing entropy. Additionally, for both TBR and WPR, applying load balancing increases the overall routing entropy as it redistributes loads more evenly across the network. Interestingly, while TBR with load balancing sees a slightly higher routing entropy due to balancing the load, WPR shows a more dramatic increase in entropy after applying load balancing, rising to at least 5 bits. This suggests that WPR should be used in conjunction with load balancing to maintain both performance and anonymity.

Lastly, Fig. 5c presents the entropy of packets ($\mathsf{H}(p)$) derived from simulations for all routing approaches. Interestingly, this case reveals a different behavior compared to routing entropy. In uniform routing, as n increases, packet entropy remains unchanged, showing the lowest entropy across all scenarios. This result, although seemingly counterintuitive, can be attributed to the fact that packet entropy depends on how many packets a targeted packet (monitored by the GPA) can be mixed with in the mix-nodes. With uniform routing, packets are mixed with a consistent portion of traffic, regardless of how many packets constitute the messages, thus maintaining the same entropy for all values of n. This entropy heavily depends on the $\frac{\lambda}{W}$ portion of traffic, where λ is the rate of incoming messages.

However, when routing is biased towards faster paths using either TBR or WPR, the portion of traffic a packet can be mixed with depends on the total number of packets in the messages. The higher the number of packets, the greater the chance for a single packet to be mixed with more packets, as the majority of packets are routed through fewer paths. Therefore, beyond $n = 1$, both TBR and WPR exhibit a dramatic increase in packet entropy. While TBR and WPR show lower routing entropy, their actual anonymity offered to packets, determined by how it is mixed with other packets in the network, is higher. PAR schemes improve packet entropy, reaching close to 10 bits for TBR and 9 bits for WPR, demonstrating the effectiveness of this approach in improving anonymity. Even after applying load balancing, packet entropy decreases slightly (by about 1 or 2 bits) but still remains higher than that of uniform routing. In summary, by using PAR schemes, we can not only reduce latency but also achieve higher anonymity for individual packets without overloading mix-nodes when employing load-balancing algorithms.

4 Mix-Node Adversary

In this section, we extend the adversarial analysis of mix-nets by considering a more potent adversary capable of compromising specific mix-nodes, referred to as a *mix-node adversary*. Unlike the GPA, which can only observe all communi-



(a) Random Corruption. (b) Greedy Corruption. (c) FCP vs. n.

Fig. 6: Adversarial Analysis: This figure explains two cases of mix-node corruption strategies an adversary might employ, along with the corresponding evaluation results.

cation links across the network to launch correlation attacks, this adversary can control some mix-nodes, allowing them to map the input to the output of those mix-nodes, leading to full deanonymization of the compromised nodes. However, the adversary's impact is limited as long as we maintain uniform routing within the mix-net. The situation becomes particularly critical when using strategic routing, such as PAR, as the adversary can more effectively compromise mix-nodes. Specifically, in this case, the adversary's effectiveness is measured by the Fraction of Fully Compromised Paths (FCP), which represents the portion of paths from the first to the last layer of the mix-net that are entirely corrupted. To investigate this adversarial model, we begin by detailing the strategies employed by the adversary to compromise the mix-net, followed by an evaluation of how our PAR proposal performs under such conditions.

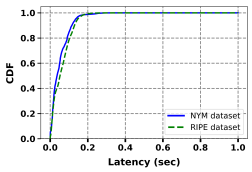
4.1 Adversarial Strategies

Random Corruption. Random corruption is the most straightforward method for compromising mix-nodes. In this approach, the adversary controls f mix-nodes, which are selected at random. The effectiveness of this strategy varies depending on which nodes are compromised, potentially resulting in either high or low FCP. Fig. 6a provides an example of this strategy, demonstrating that random selection leads to an almost negligible FCP.

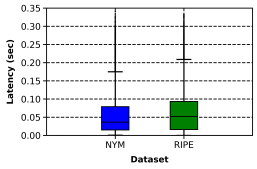
Greedy Corruption. Greedy corruption, on the other hand, is a more advanced strategy where the adversary examines the most probable paths within the mix-net and strategically compromises mix-nodes along those paths until the allotted budget of compromised nodes (f) is exhausted. This approach often acts as a theoretical upper bound on the adversary's power, as it requires prior knowledge of the mix-net's configuration to identify the most critical nodes to compromise. Fig. 6b shows that greedy corruption can considerably increase FCP compared to random corruption.

4.2 Experimental Results of Mix-node Adversary

To explore how PAR methods manage high latency in the presence of adversarial corruption, we apply both random and greedy corruption models, where



(a) CDF of datasets.



(b) Statistics of datasets.

Strategy	Initialization	Propagations
Vanilla	$O(1)$	$O(LKn)$
MSD	$O(1)$	$O(LK'n)$
PAR	$O(N \log N)$	$O(LKn)$
PAR*	$O(N^2)$	$O(LKn)$

(c) Scalability analysis of strategies. (PAR*:PAR+ Load balancing.)

Fig. 7: Datasets and complexity analysis: Figures 7a and 7b illustrate the differences between the NYM and RIPE datasets, while Tab. 7c outlines the complexity of our optimization methods.

the TBR approach is employed with $f = \frac{N}{5}$, and other parameters are kept consistent with those mentioned in § 3. In Fig. 6c, we plot the FCP against the number of packets n. The results show that greedy corruption, particularly beyond $n = 1$, consistently results in higher FCP than random corruption. As n increases, FCP from greedy corruption continues to rise, reaching up to 20% of fully compromised paths. In contrast, random corruption maintains a much lower FCP, around 2% across most values of n, without a clear trend. For some values of n, FCP slightly increases, while for others, it slightly decreases. This fluctuation is a result of randomly corrupting the mix-nodes, where, in some cases, the adversary successfully compromises mix-nodes that handle high-traffic paths, leading to a full path compromise. In other cases, the adversary corrupts nodes receiving less traffic, resulting in a smaller impact. These results suggest that while PAR schemes like TBR help reduce latency, they introduce a trade-off between the adversary's strength and the overall anonymity maintained by the system.

5 Discussion

This section provides additional insights into the dataset used to evaluate our approaches, outlines computational complexity bounds for our schemes, and offers further clarifications on assumptions and potential advantages related to adversarial strategies.

Real-World Dataset. In § 3, we used the RIPE dataset to model the latency distribution of nodes in the mix-net, as it provides globally diverse latency measurements and serves as a suitable data source for general evaluation. However, a model based on latency measurements from a deployed network would be preferable. To this end, we considered the NYM network and derived a dataset featuring latency measurements among nodes using the Verloc protocol [18], which each mixnode periodically runs to measure latency to every other mixnode.

After analyzing the derived dataset, we identified limitations in the number of available measurements. We realized that the scale at which we could evaluate our results using the NYM dataset would be much smaller than that of

the RIPE dataset (with RIPE being approximately four times larger); therefore, we opted to use the RIPE dataset for our evaluations. Nevertheless, in this section, we demonstrate that the RIPE dataset provides a reasonable approximation of latency in the NYM network (a real-world mix-net). Specifically, we derived the latency distribution of nodes as a cumulative distribution function (CDF) for both the RIPE and NYM datasets, shown in Fig. 7a. As depicted, the two datasets exhibit very similar latency distributions, with the NYM dataset showing a slightly lower overall latency.[14] Fig. 7b provides a more detailed statistical comparison of the two datasets, indicating that the average latency in the RIPE dataset is approximately 20 ms higher than in the NYM dataset, with the 75th percentile being about 25 ms higher. The RIPE dataset also shows slightly higher variance. Overall, these two distributions are quite similar, supporting the reliability and relevance of using the RIPE dataset for latency evaluation in our approach.

Scalability Analysis. In this section, we detail how our algorithms scale with network size (parameters W and L) and the number of messages sent by a client. We summarize these complexities in Tab. 7c. Specifically, we first consider the vanilla setting (the baseline mix-net), where, for initialization with a given mix-net structure, each client may need to perform a constant number of operations. When a client intends to send k messages, each consisting of n packets, they need to select L hops (mix-nodes) for each packet, resulting in an overall complexity of $O(LKn)$.[15] When using our MSD approach, the client—similar to the vanilla setting—needs to perform the same initialization operations. However, MSD relies on generating cover traffic to create additional messages, as discussed in § 2.2, leading to a complexity of $O(LK'n)$, where $K' = \left\lceil \frac{a}{(1-a)\theta+a} \right\rceil K$. This expression shows how the increased message generation in MSD contributes to additional complexity.

Furthermore, both the TBR and WPR approaches of the PAR strategy impose higher initialization complexities compared to the vanilla and MSD settings. Specifically, to apply PAR, assuming we have access to latency measurements, each layer requires sorting latencies with a complexity of $O(W \log(W))$ per layer. This results in an approximate overall complexity of $L \cdot O(W \log(W)) \approx O(N \log(N))$. Additionally, deriving the routing distribution in both TBR and WPR involves performing linear operations in terms of W for each layer, leading to a complexity of $O(N)$. Therefore, the total initialization complexity using PAR methods is $O(N \log(N))$. We may also need to apply load balancing to prevent certain mix-nodes from being overloaded. Based on the insights provided in Appendix E, this load balancing has a complexity of $O(N^2)$, resulting in an overall complexity of $O(N^2)$. Meanwhile, the complexity of message generation in PAR remains similar to that in the vanilla setting, unlike MSD. This further highlights that clients' resource expenditure may be allocated either toward ini-

[14] This lower latency can be attributed to the smaller number of peer-to-peer latency samples in NYM.

[15] This complexity corresponds to the number of messages sent by the client in each communication epoch, as seen in the NYM network [7].

tialization (as in PAR) or toward increased message generation (as in MSD) for latency optimizations.

Adversarial Advantages. In this section, we outline assumptions regarding adversarial behavior in both the MSD and PAR strategies, as well as the potential advantages an adversary might gain from these approaches. We begin with the MSD approach (§ 2.2), where we assume that additional cover traffic generated by clients can compensate for lower shuffling delays at each mix-node. Notably, we assume that an adversary (either a GPA or mix-node adversary) cannot distinguish cover traffic from real traffic, a constraint we applied in our evaluation (§ 3). This is largely ensured by the shuffling performed at each layer, which effectively mixes real and cover traffic. Ideally, clients would generate this cover traffic themselves to fully satisfy this assumption. However, in practice, we may need to offset the computational load from clients by using auxiliary entities (such as gateways in NYM [7]), which can manage cover traffic generation. Although an adversary may initially distinguish cover traffic generated by these entities, the distinction decreases as the traffic advances through multiple mixing layers.

In contrast, with PAR methods, we consider two adversarial models. The first is the GPA, which observes only the communication links connecting mix-nodes. In this case, a GPA might strategically focus on paths that handle a higher proportion of traffic due to faster links for messages with larger n values, thereby enabling more effective correlation attacks. This effect can be measured using routing entropy ($\mathsf{H}(r)$), as shown in Fig. 5b, or by packet entropy ($\mathsf{H}(p)$), as shown in Fig. 5c, describing a reasonable trade-off between security and efficiency. However, the effectiveness of this type of adversary is generally limited, and applying load balancing can substantially reduce their influence.

In § 4, we introduce a second adversarial model in which an adversary corrupts mix-nodes within the mix-net with the goal of fully compromising paths. In this model, the adversary randomly introduces some mix-nodes into the mix-net to achieve this purpose. As shown in Fig. 6c, this strategy does not lead to a significant FCP rate.[16]

This raises the question of how an adversary could leverage PAR to increase the likelihood of compromising paths. Ideally, an adversary would attempt to corrupt mix-nodes that are close to each other in terms of latency.[17] In practice, however, this is challenging, as the mix-net structure is built cooperatively by distributed entities in an adversarial-resistant manner [7], using verifiable randomness sources, as in NYM. This design makes it unlikely that an adversary can predict the exact positions of mix-nodes within the mix-net layers to corrupt the most advantageous ones. Nevertheless, in Fig. 6c, we demonstrate

[16] An α% FCP indicates that α% of paths are fully compromised, meaning each packet has an α% chance of being de-anonymized by the adversary.

[17] In NYM, adversaries could exist globally and attempt to introduce corrupted nodes into the mix-net, provided they meet certain prerequisites, such as sufficient bandwidth capacity to participate in the mix-net [7]. However, there is a limit on the number of nodes a single entity can introduce.

the effectiveness of an adversary who, by chance, is able to strategically place corrupted nodes in advantageous positions to compromise paths, compared to random corruption. However, the likelihood of this occurring remains close to zero, reinforcing the resilience of our approach against such adversaries.[18]

6 Conclusion

In this work, we addressed the high-latency problem in mix-nets caused by both overlay routing through intermediary nodes and shuffling delays at each node with high granularity to ensure anonymity when delivering a message consisting of n packets. We first theoretically analyzed the latency caused by shuffling delays for both a single mix-node and a full mix-net scenario, as well as providing insights into routing delays. Based on this, we introduced MSD, a framework that preserves anonymity with differential privacy guarantees while reducing shuffling delay at mix-nodes, controlled by a cost parameter θ to balance between privacy and performance. Additionally, we proposed PAR, which includes two schemes: TBR and WPR, focused on reducing routing delays by prioritizing faster links as the number of packets in the messages increases. We also introduced a load-balancing algorithm to prevent overloading when using PAR, which proved to be more effective and efficient than the state-of-the-art LARMix. Lastly, we provided detailed analytical and simulation experiments showing that our methods effectively reduced latency without significantly compromising anonymity, even under advanced adversarial settings, such as mix-node adversary attacks when using PAR. We believe this work will help improve mix-net designs to better manage latency while maintaining anonymity, specifically with the granularity required for applications involving messages consisting of more than one packet.

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

References

1. Akhoondi, M., Yu, C., Madhyastha, H.V.: Lastor: A low-latency as-aware tor client. In: 2012 IEEE Symposium on Security and Privacy. pp. 476–490. IEEE (2012)
2. Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against tor. In: Proceedings of the 2007 ACM workshop on Privacy in electronic society. pp. 11–20 (2007)
3. Chaum, D., Das, D., Javani, F., Kate, A., Krasnova, A., De Ruiter, J., Sherman, A.T.: cmix: Mixing with minimal real-time asymmetric cryptographic operations. In: Applied Cryptography and Network Security: 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings 15. pp. 557–578. Springer (2017)

[18] LARMix [26] has suggested alternative methods for compromising nodes, which have been shown to be no more effective than random corruption.

4. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2), 84–90 (1981)
5. Danezis, G.: Mix-networks with restricted routes. In: Privacy Enhancing Technologies: Third International Workshop, PET 2003, Dresden, Germany, March 26-28, 2003. Revised Papers 3. pp. 1–17. Springer (2003)
6. Diaz, C.: Anonymity and privacy in electronic services. Heverlee: Katholieke Universiteit Leuven. Faculteit Ingenieurswetenschappen (2005)
7. Diaz, C., Halpin, H., Kiayias, A.: The nym network (2021)
8. Diaz, C., Murdoch, S.J., Troncoso, C.: Impact of network topology on anonymity and overhead in low-latency anonymity networks. In: Privacy Enhancing Technologies: 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings 10. pp. 184–201. Springer (2010)
9. Diaz, C., Preneel, B.: Taxonomy of mixes and dummy traffic. In: Information Security Management, Education and Privacy: IFIP 18th World Computer Congress TC11 19th International Information Security Workshops 22–27 August 2004 Toulouse, France. pp. 217–232. Springer (2004)
10. Diaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: International Workshop on Privacy Enhancing Technologies. pp. 54–68. Springer (2002)
11. Diaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Privacy Enhancing Technologies: Second International Workshop, PET 2002 San Francisco, CA, USA, April 14–15, 2002 Revised Papers. pp. 54–68. Springer (2003)
12. Dingledine, R., Mathewson, N., Syverson, P.F., et al.: Tor: The second-generation onion router. In: USENIX security symposium. vol. 4, pp. 303–320 (2004)
13. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)
14. Evans, N.S., Dingledine, R., Grothoff, C.: A practical congestion attack on tor using long paths. In: USENIX Security Symposium. pp. 33–50 (2009)
15. Johnson, A., Wacek, C., Jansen, R., Sherr, M., Syverson, P.: Users get routed: Traffic correlation on tor by realistic adversaries. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 337–348 (2013)
16. Kesdogan, D., Egner, J., Büschkes, R.: Stop-and-go-mixes providing probabilistic anonymity in an open system. In: International Workshop on Information Hiding. pp. 83–98. Springer (1998)
17. Kesdogan, D., Egner, J., Büschkes, R.: Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In: Proceedings of Information Hiding Workshop (IH 1998). Springer-Verlag, LNCS 1525 (1998)
18. Kohls, K., Diaz, C.: {VerLoc}: Verifiable localization in decentralized systems. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 2637–2654 (2022)
19. Kwon, A., Lu, D., Devadas, S.: {XRD}: Scalable messaging system with cryptographic privacy. In: 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20). pp. 759–776 (2020)
20. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster protocol—version 2 (2003)
21. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th *USENIX* Security Symposium (*USENIX* Security 17). pp. 1199–1216 (2017)
22. Python: Event discrete, process based simulation for python. `https://pypi.org/project/simpy/` (2013)

23. Rahimi, M.: CLAM: client-aware routing in mix networks. In: Pérez-González, F., Alfaro, P.C., Krätzer, C., Zhao, H.V. (eds.) Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2024, Baiona, Spain, June 24-26, 2024. pp. 199–209. ACM (2024). https://doi.org/10.1145/3658664.3659631, https://doi.org/10.1145/3658664.3659631
24. Rahimi, M.: Larmix++: Latency-aware routing in mix networks with free routes topology. In: International Conference on Cryptology and Network Security. pp. 187–211. Springer (2024)
25. Rahimi, M.: Malaria: Management of low-latency routing impact on mix network anonymity. In: 2024 IEEE 22nd International Symposium on Network Computing and Applications (NCA). vol. 22. IEEE (2024)
26. Rahimi, M., Sharma, P.K., Diaz, C.: Larmix: Latency-aware routing in mix networks. In: The Network and Distributed System Security Symposium. Internet Society (2024)
27. Rochet, F., Wails, R., Johnson, A., Mittal, P., Pereira, O.: Claps: Client-location-aware path selection in tor. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 17–34 (2020)
28. Shannon, C.E.: Communication theory of secrecy systems. The Bell system technical journal **28**(4), 656–715 (1949)
29. Shirazi, F., Simeonovski, M., Asghar, M.R., Backes, M., Diaz, C.: A survey on routing in anonymous communication protocols. ACM Computing Surveys (CSUR) **51**(3), 1–39 (2018)
30. Staff, R.N.: Ripe atlas: A global internet measurement network. Internet Protocol Journal **18**(3), 2–26 (2015)
31. Sun, Y., Edmundson, A., Vanbever, L., Li, O., Rexford, J., Chiang, M., Mittal, P.: {RAPTOR}: Routing attacks on privacy in tor. In: 24th USENIX Security Symposium (USENIX Security 15). pp. 271–286 (2015)
32. Van Den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: Scalable private messaging resistant to traffic analysis. In: Proceedings of the 25th Symposium on Operating Systems Principles. pp. 137–152 (2015)
33. Wan, G., Johnson, A., Wails, R., Wagh, S., Mittal, P.: Guard placement attacks on path selection algorithms for tor. Proceedings on Privacy Enhancing Technologies **2019**(4) (2019)

A Key Notations

We present the full notation of variables used in this paper, as depicted in Tab. 2.

B Latency Emulation

As previously discussed, we model the link-delay latency in our network using the RIPE ATLAS dataset [30]. It is important to note that latency on the Internet can vary due to the bidirectional paths between the sender and receiver, and vice versa. To ensure the validity of our analysis, we first plotted the cumulative distribution of both sender-to-receiver and receiver-to-sender latency, as shown in Fig. 8. The figure indicates that both directions exhibit almost identical distributions, confirming that our model is accurate regardless of the direction considered. Therefore, we model the link delay between two nodes by averaging their sender-to-receiver and receiver-to-sender latency.

Table 2: List of key notations and variables.

Symbol	Meaning
U	Number of clients
W	Number of mix-node at each layer
N	Number of mix-nodes
L	Number of layers
r_{ij}^k	Probability distribution between node i at layer $k-1$ and j at layer k
f	Number of corrupted nodes
$\mathcal{R}^k$	Routing matrix between layer $k-1$ and layer k
l_c	Client to mix-net latency
l_r	Routing delays within mix-nodes
l_{e2e}	End-to-end latency
l_d	mix-net to destination latency
l_s	Shuffling delay
a	Hyper parameter in MSD
$\mathcal{I}$	Ranking function
$\mathsf{H}(r)$	Entropy of routing
$\mathsf{H}(P)$	Entropy of packets
μ	Exponential distribution parameter
λ	Poisson rate of incoming messages
l_{ij}	Propagation latency between mix-nodes i and j
θ	Cost parameter

C Proof of Theorems

In this section, we provide the proofs for the theorems referenced throughout the paper.

C.1 Average of Shuffling Delay for a Single Mix-Node Scenario

Eq. (6) provided the average delay for sending n packets to deliver a message in a single-mix-node scenario; however, it does so without presenting a simplified or closed form. To enhance comprehension, an exploration into the harmonic number H_n is required. Defined as $\sum_{i=1}^{n} \frac{1}{i}$, the harmonic number represents the summation of the reciprocals of natural numbers up to n, which plays a pivotal role in simplifying Eq. (6).

To this end, we propose the following theorem, aimed at elucidating the relationship between the harmonic number and the combinatorial coefficients:

Theorem 5. *The nth harmonic number, H_n, can be equivalently expressed as:*

$$\sum_{i=1}^{n} \binom{n}{i} \frac{(-1)^{i+1}}{i}.$$

Proof. We begin with Eq. (13), which illustrates a geometric series, and proceed with its integration. This mirrors the definition of harmonic numbers as outlined in Eq. (14). In Eq. (15), we perform a substitution with $u = 1 + z$, leading to

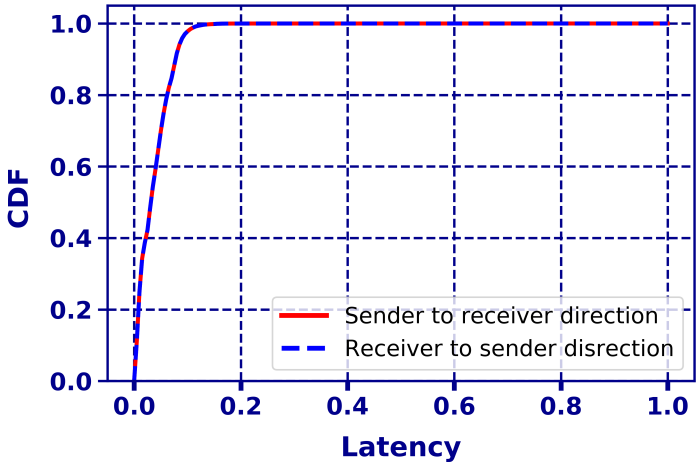


Fig. 8: CDF of RIPE dataset.

further integrations that exploit the binomial series expansion. This sequence of operations culminates in Eq. (16), thus validating the theorem and reaffirming the harmonic number's representation.

$$\frac{1-u^n}{1-u} = 1 + u + u^2 + \cdots + u^{n-1}, \tag{13}$$

$$\int_0^1 \frac{1-u^n}{1-u}\,\mathrm{d}u = \sum_{i=1}^{n} \frac{1}{i} = H_n, \tag{14}$$

$$\int_0^1 \frac{1-u^n}{1-u}\,\mathrm{d}u = \int_{-1}^{0} \frac{1-(z+1)^n}{-z}\,\mathrm{d}z, \tag{15}$$

$$= \int_{-1}^{0} \frac{-1+\sum_{i=0}^{n}\left[\binom{n}{i} z^i\right]}{z}\,\mathrm{d}z,$$

$$= \int_{-1}^{0} \left[\frac{-1}{z} + \sum_{i=0}^{n}\left[\binom{n}{i} z^{i-1}\right]\right]\,\mathrm{d}z = \sum_{i=1}^{n} \binom{n}{i} \frac{(-1)^{i+1}}{i}. \tag{16}$$

Equipped with this theorem, we can refine Eq. (6) to present Eq. (17), which shows that the average latency experienced by the destination is inversely proportional to μ, the rate parameter of the exponential distribution. Notably, as the number of packets n increases, the latency proportionally escalates. For instance, when $n = 1$, the average latency is equivalent to the mean of the exponential distribution. As n increases to 2, the latency rises to $\frac{3}{2}$ times the exponential mean, demonstrating a consistent trend of increasing latency as more packets are involved.

$$\mathbb{E}[Y] = \frac{1}{\mu}\left(\sum_{i=0}^{n-1}\binom{n}{i}\frac{(-1)^{n-i+1}}{n-i}\right),$$
$$= \frac{1}{\mu}\left(\sum_{i=1}^{n}\binom{n}{i}\frac{(-1)^{i+1}}{i}\right) = \frac{H_n}{\mu}. \tag{17}$$

C.2 Variance of Shuffling Delay for a Single Mix-Node Scenario

To further analyze the variance of latency for n packets departing from the mix-node, we adopt a systematic approach similar to that used for determining $\mathbb{E}[Y]$. Initially, the calculation of $\mathbb{E}[Y^2]$ is provided through Eq. (20). Then, for computing the variance, we use $Var(Y) = \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2$, as the variance of latency, unlike the average latency, cannot be expressed in closed form.

$$\mathbb{E}[Y^2] = \int_0^\infty t^2 f_Y(t)\,\mathrm{d}t = \int_0^\infty \mu n t^2 e^{-\mu t}\left(1-e^{-\mu t}\right)^{n-1}\mathrm{d}t, \tag{18}$$
$$= \mu n\left(\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^{n-i-1}\int_0^\infty t^2 e^{-\mu(n-i)t}\mathrm{d}t\right), \tag{19}$$
$$= \frac{2n}{\mu^2}\left(\sum_{i=0}^{n-1}\binom{n-1}{i}\frac{(-1)^{n-i}}{(n-i)^3}\right),$$
$$= \frac{2}{\mu^2}\left(\sum_{i=0}^{n-1}\binom{n}{i}\frac{(-1)^{n-i}}{(n-i)^2}\right) = \frac{2}{\mu^2}\left(\sum_{i=1}^{n}\binom{n}{i}\frac{(-1)^{i}}{i^2}\right). \tag{20}$$

C.3 Average of Latency for a Mix-Net Scenario

To calculate the average latency experienced by a message composed of n packets traversing a mixnet with L intermediary hops, we consider each packet, indexed from 1 to n, as being subject to a random delay X_i. This variable represents the time it takes for each packet to pass through the mixnet, from entry to exit. Given this setup, our primary focus is on the latency induced directly by the mixnodes. As there are L mixnodes each introducing a delay, modeled by an exponential distribution $Exp(\mu)$, X_i essentially aggregates L independent and identically distributed exponential variables, thereby following an Erlang distribution characterized by parameters L and μ, succinctly described by $f_{X_i} = \frac{t^{L-1}\mu^L e^{-\mu t}}{(L-1)!}$. This formulation prompts us to examine the total waiting time required for all n packets to be aggregated at the destination.

The probability of interest, $\mathbb{P}(Y = \max(X_1, X_2, ..., X_n) \leq t)$, where Y is the maximum time among all packets to exit the mixnet, can be expressed as the product of the probabilities of each packet's exit time, $\prod_{i=1}^{n}\mathbb{P}(X_i \leq t)$. By

employing the cumulative distribution function (CDF) of the Erlang distribution, $\frac{\gamma(L,\mu t)}{(L-1)!}$, we can articulate the CDF of Y as shown in Eq. (21):

$$\mathbb{P}(Y \leq t) = \left(\frac{\gamma(L, \mu t)}{(L-1)!}\right)^n \tag{21}$$

In this equation, γ signifies the lower incomplete gamma function, defined as $\gamma(L, \mu t) = \int_0^{\mu t} x^{L-1} e^{-x}, \mathrm{d}x$. This function's derivative, $\frac{\mathrm{d}}{\mathrm{d}t}\gamma(L, \mu t)$, equates to $\mu^L t^{L-1} e^{-\mu t}$, which is instrumental in determining the probability density function (PDF) of Y, specified in Eq. (22):

$$f_Y(y) = n\mu^L \frac{t^{L-1} e^{-\mu t}}{(L-1)!} \left[\frac{\gamma(L, \mu t)}{(L-1)!}\right]^{n-1} \tag{22}$$

Consequently, the expectation $\mathbb{E}(Y)$ is derived as detailed in Eq. (23), highlighting the intricacies of calculating average latency within a mixnet framework where direct expressions are less accessible:

$$\mathbb{E}(Y) = \int_0^\infty t f_Y(t)\, \mathrm{d}t = \int_0^\infty n\mu^L \frac{t^L e^{-\mu t}}{(L-1)!} \left[\frac{\gamma(L, \mu t)}{(L-1)!}\right]^{n-1} \mathrm{d}t. \tag{23}$$

While obtaining a closed form for $\mathbb{E}[Y]$ is not feasible in this context, we employ an approximation to examine the influence of network parameters on the average latency. Initially, we hypothesize that $\gamma(L, \mu t)$ can be approximated by an upper limit, defined by the integral $\int_0^{\mu t} x^{L-1}\, \mathrm{d}x = \frac{(\mu t)^L}{L}$, based on the assumption that $\exp(-x), x \geq 0$ is always less than or equal to 1. With this approximation, Eq. (23) simplifies as follows:

$$\begin{aligned}
\mathbb{E}[Y] &= \int_0^\infty t f_Y(t)\, \mathrm{d}t = \int_0^\infty n\mu^L \frac{t^L e^{-\mu t}}{(L-1)!} \left[\frac{\gamma(L, \mu t)}{(L-1)!}\right]^{n-1} \mathrm{d}t \\
&\leq \int_0^\infty n\mu^L \frac{t^L e^{-\mu t}}{(L-1)!} \left[\frac{(\mu t)^L}{L!}\right]^{n-1} \mathrm{d}t \\
&= \int_0^\infty Lne^{-\mu t} \left[\frac{(\mu t)^L}{L!}\right]^n \mathrm{d}t = \frac{Ln\mu^{Ln}}{(L!)^n} \int_0^\infty t^{Ln} e^{-\mu t}\, \mathrm{d}t, \\
&= \frac{Ln\mu^{Ln}}{(L!)^n} \frac{(Ln)!}{\mu^{Ln+1}} = \frac{Ln(Ln)!}{(L!)^n \mu}.
\end{aligned} \tag{24}$$

Analyzing Eq. (24), we explore how altering network parameters impacts average latency. Notably, increasing the average mixing delay in mixnodes, represented as $\frac{1}{\mu}$, or decreasing μ, leads to heightened average latency. This effect is intuitive, as prolonging mixing delay at each node means messages linger longer within mixnodes, thereby increasing latency. Conversely, incrementing the packet count required for message detection from n to $n+1$ exhibits an increase in expected latency, encapsulated in Eq. (25), confirming our anticipations.

In further detail, Eq. (25) reveals an insightful aspect of how the average latency varies with an increase in the number of packets. Considering the factorial notation, where $(L(n+1))!$ represents the product sequence from $(Ln+L)$ down to $(Ln)!$, it becomes evident that the expression $\frac{(L(n+1))!}{(Ln)!}$ equates to a product sequence from $(Ln+L)$ to $(Ln+1)$, inherently exceeding $L!$. This mathematical derivation unequivocally illustrates that introducing an additional packet to the mixnet invariably raises the average latency, thereby aligning perfectly with our expectations and reinforcing the principle that an increase in packet quantity directly contributes to heightened average latency.

$$\frac{\mathbb{E}_{n+1}(Y)}{\mathbb{E}_n(Y)} = \frac{\frac{L(n+1)(L(n+1))!}{(L!)^{n+1}\mu}}{\frac{Ln(Ln)!}{(L!)^n\mu}} = \frac{n+1}{n}\frac{(L(n+1))!}{(L!)(Ln)!} > 1. \tag{25}$$

Moreover, examining the effect of incrementing the number of hops from L to $L+1$ reveals a consistent pattern of increased average latency, as depicted in Eq. (26). This increment underscores the notion that additional hops necessitate extended periods for messages to undergo adequate mixing, consequently elevating latency:

$$\frac{\mathbb{E}_{L+1}(Y)}{\mathbb{E}_L(Y)} = \frac{\frac{(L+1)n((L+1)n)!}{((L+1)!)^n\mu}}{\frac{Ln(Ln)!}{(L!)^n\mu}} = \frac{L+1}{L}\prod_{i=1}^{n}\frac{Ln+i}{L+1} > 1. \tag{26}$$

C.4 Variance and α Percentile of Latency for a Mix-Net Scenario

Given the absence of a closed-form expression for the probability density function (PDF) of the variable Y, the variance of latency is calculated using the formula $\mathbb{E}[Y^2(t)] - \mathbb{E}^2[Y(t)]$, which utilizes the PDF of Y. Additionally, within the mixnet scenario, it's pertinent to define the α-percentile of latency. This parameter indicates the maximum delay by which, for at least α percent of the cases, all packets are guaranteed to be received by the destination.

We designate t_α as the time threshold fulfilling the condition $\mathbb{P}(Y(t) \leq t_\alpha) = \alpha$. Referencing Eq. (27), the α-percentile time is determined, as shown in Eq. (28), with γ_L^{-1} representing the inverse of the gamma function $\gamma(L, \mu t)$, thereby yielding the value of μt.

Due to the monotonically increasing nature of the lower incomplete gamma function, its inverse also directly correlates t_α with the hop count L. Thus, an increase in L augments the inverse of the lower incomplete gamma function, which in turn escalates t_α. This outcome logically indicates that more hops equate to extended latency, thereby prolonging the time needed for receivers to accumulate α percent of the packets. Moreover, a rise in n inversely affects $\frac{1}{n}$, thereby enhancing $\alpha^{\frac{1}{n}}$. Given that $0 \leq \alpha \leq 1$, this implies that a greater packet count necessary for complete message reconstruction extends the waiting time for the receiver. Additionally, it illustrates that an increase in α directly influences t_α. Importantly, t_α showcases an inverse relationship with μ, thus establishing a

direct association with $\frac{1}{\mu}$. Consequently, a prolonged mixing delay incrementally extends the duration required to reliably receive the α percentage of packets.

$$\mathbb{P}(Y(t) \leq t_\alpha) = \left[\frac{\gamma(L, \mu t_\alpha)}{(L-1)!}\right]^n = \alpha. \tag{27}$$

$$\left[\frac{\gamma(L, \mu t_\alpha)}{(L-1)!}\right] = \alpha^{\frac{1}{n}}.$$

$$t_\alpha = \frac{\gamma_L^{-1}((L-1)!\alpha^{\frac{1}{n}})}{\mu}. \tag{28}$$

D Proof of KL Divergence for TBR and WPR

This section provides a detailed proof of the KL divergence between the TBR and WPR methods.

Let the probability distribution for TBR, Q, be defined as:

$$Q_{ij}^k = \begin{cases} \frac{\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}{\sum_{j\in\mathcal{S}_k}\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}, & \text{if } \mathcal{I}_{ij}^k \leq \left\lceil \frac{W}{H_n} \right\rceil, \\ 0, & \text{otherwise.} \end{cases}$$

Here:

- l_{ij}^k represents the latency from node i in layer $k-1$ to node j in layer k,
- H_n is the harmonic number (or another control parameter),
- $\mathcal{I}_{ij}^k$ is a rank function,
- W is the total number of nodes in each layer, and
- $\mathcal{S}_k$ is the set of nodes satisfying the latency condition.

On the other hand, the probability distribution for WPR, P, is given by:

$$P_{ij}^k = \frac{\left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k \cdot H_n} \cdot \left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}{\sum_{j=1}^{W}\left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k \cdot H_n} \cdot \left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}.$$

In this case, the expression $\left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k \cdot H_n}$ gives more weight to paths with a lower index $\mathcal{I}_{ij}^k$.

The **KL divergence** between two probability distributions Q (TBR) and P (WPR) is defined as:

$$D_{KL}(Q||P) = \sum_{i\in\mathcal{S}_k} Q_{ij}^k \log \frac{Q_{ij}^k}{P_{ij}^k}.$$

Substitute the expressions for Q_{ij}^k and P_{ij}^k into the equation:

$$D_{KL}(Q||P) = \sum_{i\in\mathcal{S}_k} \frac{\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}{\sum_{j\in\mathcal{S}_k}\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}} \log\left(\frac{\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}{\sum_{j\in\mathcal{S}_k}\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}} \Bigg/ \frac{\left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k\cdot H_n}\cdot\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}{\sum_{j=1}^{W}\left(\frac{1}{\sqrt{H_n}}\right)^{\mathcal{I}_{ij}^k\cdot H_n}\cdot\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}\right). \tag{29}$$

Since the denominator of WPR is less than that of TBR, we can conclude that:

$$D_{KL}(Q||P) \leq -\sum_{i\in\mathcal{S}_k} \frac{\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}}{\sum_{j\in\mathcal{S}_k}\left(\frac{1}{l_{ij}^k}\right)^{\frac{H_n-1}{H_n}}} \log\left(\frac{1}{\sqrt{H_n}}^{\mathcal{I}_{ij}^k\cdot H_n}\right).$$

This expression indicates that the divergence depends primarily on the ratio of weights assigned to each path by TBR and WPR. Furthermore, the sum of probabilities across all paths (in both Q and P) adds up to 1. Given that both methods prioritize lower-latency paths and $\mathcal{I}_{ij}^k \leq W-1$, we obtain the following bound for the KL divergence:

$$D_{KL}(P||Q) \leq \frac{(W-1)H_n}{2}\log(H_n),$$

showing that as the number of packets increases, the divergence between these two probability distributions also increases.

E Load Balancing Algorithm

In this section, we provide the pseudocode for our load-balancing algorithm alongside its complexity explanation compared to that of LARMix [26]. Additionally, we offer a toy example to demonstrate how our balancing algorithm works.

E.1 Load Balancing Algorithm

E.2 Comparison with LARMix [26]

As mentioned in §2.3, after applying strategic routing within the mix-net, some nodes might receive a disproportionately high portion of the traffic. This situation increases the adversary's advantage and escalates load imbalance, potentially leading to higher queuing delays. To address this issue, load balancing

Algorithm 1 Load Balancing Algorithm

Input: Routing matrix $\mathcal{R}^k$
Initialize: Compute the load for each column in $\mathcal{R}^k$
for each column in $\mathcal{R}^k$ **do**
  if Load < 1 **then**
    Mark as *underloaded*
  else if Load > 1 **then**
    Mark as *overloaded*
  else
    Mark as *balanced*
  end if
end for
for each overloaded column i in $\mathcal{R}^k$ **do**
  $\ell \leftarrow \sum[\mathcal{R}^k[i]] - 1$
  Distribute ℓ equally across W' entries (those capable of reducing load)
end for
for each underloaded column i in $\mathcal{R}^k$ **do**
  $\ell' \leftarrow 1 - \sum[\mathcal{R}^k[i]]$
  Add ℓ' equally to the entries
end for
Output: Balanced $\mathcal{R}^k$

needs to be applied to the routing matrix of each layer $\mathcal{R}^k$. This can be achieved through either the proposed approach in this paper or by using the LARMix greedy algorithm [26].

The LARMix algorithm identifies underloaded, overloaded, and balanced mix-nodes. It then proceeds by multiplying the overloaded columns by the inverse of their sum and redistributing the excess to underloaded nodes based on the LARMix routing formula. However, this process is computationally expensive, requiring up to N^2 operations for multiplying entries and redistributing leftovers. This redistribution involves further calculations, leading to a total complexity of $\frac{k'N^4}{16}$, where k' represents the number of iterations, which may reach up to N.

In contrast, our approach requires the identification of overloaded, underloaded, and balanced columns, which can be done in $O(W)$ complexity. The redistribution of excess load also takes $O(W)$ time per column. Overall, our method has a total complexity of $O(N^2)$, significantly improving over LARMix's $O(N^4)$.

E.3 A Toy Example

To illustrate our load-balancing approach more clearly, consider a scenario where the routing matrix $\mathcal{R}^k$ between two successive layers of a mix-net is derived from low-latency routing as follows:

$$\mathcal{R}^k = \begin{bmatrix} 0 & 0.1 & 0.1 \\ 0.7 & 0.4 & 0.2 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}.$$

The summation of the columns in this matrix is $[1.2, 1, 0.8]$, indicating that:

- The second mix-node in the second hop is **balanced**,
- The third mix-node is **underloaded**, and
- The first mix-node is **overloaded**.

Step 1: Handling Overloaded Columns
We begin by addressing the overloaded first column, which has an excess load of $\ell = 0.2$. Since the first entry in this column is already at capacity (0), we subtract the excess load ℓ from the remaining entries that have room for reduction. After redistributing ℓ across the second and third entries, the new matrix becomes:

$$\mathcal{R}^k = \begin{bmatrix} 0 & 0.1 & 0.1 \\ 0.6 & 0.4 & 0.2 \\ 0.4 & 0.5 & 0.5 \end{bmatrix}.$$

Step 2: Handling Underloaded Columns
Next, we balance the underloaded third column, which has a deficit of $\ell' = 1 - 0.8 = 0.2$. First, we note that only the first and third entries of the third column have room to receive additional load to ensure that each row maintains a valid probability distribution. The first and second rows should receive 0.9 and 0.1, respectively, resulting in the following matrix:

$$\mathcal{R}^k = \begin{bmatrix} 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \\ 0.4 & 0.5 & 0.1 \end{bmatrix}.$$

This example demonstrates how our algorithm systematically redistributes the load to ensure that each mix-node in the routing matrix maintains a balanced load, preventing any node from being overwhelmed or underutilized.

F Differential Privacy Guarantee of Poisson Noise

To establish the differential privacy guarantee of the Poisson noise mechanism, we begin by considering the mechanism $\mathcal{M}$ applied to two adjacent datasets X and X', which differ by exactly one entry. Our objective is to demonstrate that this mechanism satisfies (ϵ, δ)-differential privacy. We achieve this by analyzing the ratio of probabilities that the mechanism outputs some event T under these two datasets.

We start by defining the ratio of probabilities of the mechanism's output under the two datasets as follows:

$$\frac{\Pr(\mathcal{M}(X) \in T)}{\Pr(\mathcal{M}(X') \in T)}.$$

Given that X and X' differ in exactly one entry, say at index j, this ratio simplifies to:

$$\frac{\Pr(\mathcal{M}(X) \in T)}{\Pr(\mathcal{M}(X') \in T)} = \prod_{i=1}^{d} \frac{\text{Pois}(\lambda'_i, x_i)}{\text{Pois}(\lambda'_i, x'_i)} = \frac{\text{Pois}(\lambda'_j, x_j)}{\text{Pois}(\lambda'_j, x'_j)}.$$

Here, $\text{Pois}(\lambda'_i, x_i)$ represents the Poisson probability mass function (PMF) for a given λ'_i and x_i, which is expressed as:

$$\text{Pois}(\lambda'_i, x_i) = \frac{{\lambda'_i}^{x_i} e^{-\lambda'_i}}{x_i!}.$$

Next, we consider the Poisson PMF for the adjacent dataset, where the entry at j differs by one:

$$\text{Pois}(\lambda'_j, x_j - 1) = \frac{{\lambda'_j}^{x_j - 1} e^{-\lambda'_j}}{(x_j - 1)!}.$$

The ratio of these two probabilities is then given by:

$$\frac{\text{Pois}(\lambda'_j, x_j - 1)}{\text{Pois}(\lambda'_j, x_j)} = \frac{\frac{{\lambda'_j}^{x_j - 1} e^{-\lambda'_j}}{(x_j - 1)!}}{\frac{{\lambda'_j}^{x_j} e^{-\lambda'_j}}{x_j!}} = \frac{x_j}{\lambda'_j}.$$

We analyze this ratio under two cases. First, when $x_j \leq \lambda'_j + c\sqrt{\lambda'_j}$, the ratio is bounded by:

$$\frac{\text{Pois}(\lambda'_j, x_j - 1)}{\text{Pois}(\lambda'_j, x_j)} \leq \frac{\lambda'_j + c\sqrt{\lambda'_j}}{\lambda'_j} = e^{\epsilon}.$$

In the second case, when $x_j > \lambda'_j + c\sqrt{\lambda'_j}$, the probability of such a large deviation can be bounded using a tail bound for the Poisson distribution. This probability is negligible and corresponds to the δ term in differential privacy:

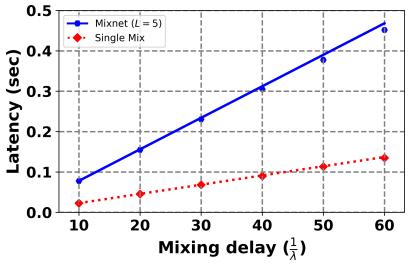
$$\Pr[X \geq x_j] \leq \frac{e^{-\lambda'_j} (e\lambda'_j)^{x_j}}{x_j^{x_j}}.$$

Finally, by combining these two cases, the overall ratio of probabilities is bounded by:

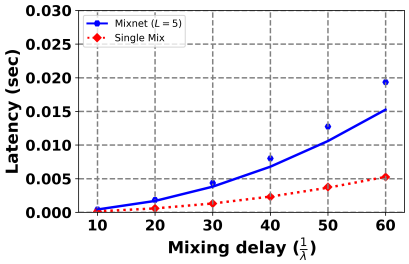
$$\frac{\Pr(\mathcal{M}(X) \in T)}{\Pr(\mathcal{M}(X') \in T)} \leq e^{\epsilon} + \delta,$$

where $e^{\epsilon} = \frac{\lambda'_j + c\sqrt{\lambda'_j}}{\lambda'_j}$ and $\delta = \frac{e^{-\lambda'_j} (e\lambda'_j)^{\lambda'_j + c\sqrt{\lambda'_j}}}{(\lambda'_j + c\sqrt{\lambda'_j})^{\lambda'_j + c\sqrt{\lambda'_j}}}$.

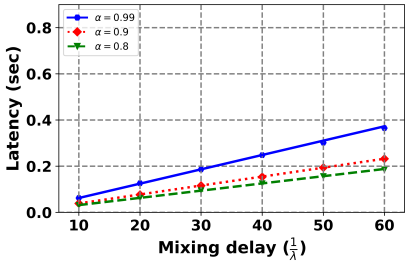
This result confirms that the Poisson noise mechanism satisfies (ϵ, δ)-differential privacy, with the parameters ϵ and δ as defined above.



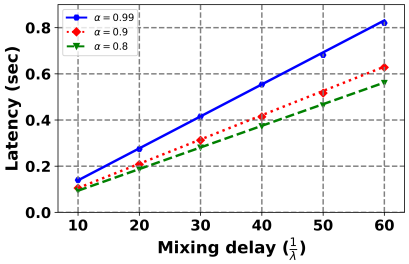
(a) Average of latency.



(b) Variance of latency.



(c) α percentile of latency for a single mix.



(d) α percentile of latency for a mixnet.

Fig. 9: Effect of mixing delay on message latency when $n = 5$ and $L = 5$.

F.1 Extension to k Rounds

This theorem can be extended to the case where the observer gathers k datasets. In this case, the privacy guarantees can be generalized as follows:

Theorem 6. *Consider an algorithm $\mathcal{M}$ providing (ϵ, δ)-differential privacy. Then, $\mathcal{M}$ provides (ϵ', δ')-differential privacy after k rounds with parameters:*

$$\epsilon' = \epsilon\sqrt{2k\ln(1/\delta)} + k\epsilon(e^{\epsilon} - 1)$$

and

$$\delta' = k\delta + \delta,$$

for any $\delta > 0$, trading higher ϵ for lower δ.

Proof. This result follows directly from Theorem 3.20 in [13].

G Expanding on Theoretical Evaluations

This subsection delves into the influence of mixing delay on the overall end-to-end latency. Through the lens of Fig. 9, a comparative analysis unfolds, elucidating

the average latency, variance, and α percentile across both single mixnode and mixnet configurations, with a specific focus on the role of the average mixing delay denoted by $\frac{1}{\lambda}$. As illustrated in Fig. 9a, a distinct linear correlation emerges, showcasing how the average latency amplifies in tandem with increases in $\frac{1}{\lambda}$, both for a singular stop-and-go mixnode with $n = 5$ and a mixnet architecture consisting of $L = 5$ hops, also at $n = 5$. This linear progression is anticipated and receives mathematical corroboration from Eq. (17), particularly for the single mix scenario, affirming a direct linkage between a mixnode's average latency and its corresponding mixing delay. As a result, any increase in the mixing delay directly scales the aggregate latency, which can also be extended to the mixnet scenario considering the approximation in Eq. (24). This underscores the need for careful adjustment of mixnode delays to effectively mitigate the potential latency burden.

Further examination in Fig. 9b reveals the variance of latency accentuating as the average mixing delay $\frac{1}{\lambda}$ advances for both the single mixnode and multi-hop scenarios. This increase in variance, while consistent across both contexts, displays a pronounced disparity in the mixnet scenario, indicating a generally higher baseline of variance. Despite this, the rate of variance growth relative to $\frac{1}{\lambda}$ adjustments remains relatively parallel in both scenarios. This deviation from the logarithmic variance increase observed with an elevated packet count n underscores the potential for heightened latency fluctuation risks, emphasizing the necessity for precision in configuring mixing delays to safeguard against undue latency variability.

The illustration of latency percentiles in Figures 9c and 9d for single mixnode and multiple hops scenarios, respectively, further demarcates the incremental nature of the α percentile in concordance with rising mixing delays, as captured by $\frac{1}{\lambda}$. This uniform linear escalation across the 80th, 90th, and 99th percentiles starkly contrasts with the more logarithmic pattern observed when comparing latency against packet numbers n, projecting a pronounced disparity, particularly at the 99th percentile. This disparity signals that, in operational settings, expectations for swift mixnet functionalities may face challenges, especially considering the heightened latency at critical percentiles. Thus, in pursuit of optimizing user experience, a comprehensive understanding and consideration of latency behavior, especially at upper percentiles, become indispensable in calibrating mixing delay parameters.

H Comparison of Different Mix-Net Topologies

As mentioned earlier, mix-nets can be implemented using various topologies, such as cascades, free routes, and stratified topologies. The topology of a mix-net significantly influences its ability to provide anonymity and manage cover traffic. Cover traffic, or dummy traffic, consists of messages added to the mix-net by clients or mix-nodes that do not have a specified destination. These messages loop back to their originator and are designed to mislead a GPA and enhance anonymity. Typically, cover traffic is generated based on the input traffic directed

to mix-nodes, ensuring that all mix-nodes handle nearly the same amount of traffic.

The cascade topology, with its restricted routing paths, facilitates greater alignment and easier management of cover traffic compared to other topologies, such as free routes and stratified configurations, which result in a more scattered distribution of traffic [8]. Additionally, because the cascade topology features restricted connections, it channels more client traffic through the same cascades, thereby offering high anonymity for client messages. However, this also makes the client message routes more deterministic, whereas stratified and free route topologies can be advantageous when more diverse routing is desired.