Analysis and Attacks on the Reputation System of Nym

Xinmu Alexis Cao
Johns Hopkins University
xcao26@jhu.edu

Matthew Green
Johns Hopkins University
mgreen@cs.jhu.edu

Abstract

Nym is a reputation- and incentive-enhanced anonymous communications network that utilizes staking, performance monitoring, and rewards to encourage high-quality contributions. In this work, we analyze the reputation mechanism used in Nym's Mixnet and NymVPN service. Using a combination of source code analysis, data collection from Nym mainnet, and network simulations with a custom simulator, we demonstrate active attacks that may allow a moderately resourced adversary to gain control of a fraction of Nym Mixnet's *active set*. This condition may enable connection de-anonymization attacks. In particular, we show that the mechanism Nym uses to measure node performance is vulnerable to a form of "framing" attack that allows a small number of low-stake nodes to damage the score of high-reputation active nodes. We then consider and discuss various mitigations. This work highlights the challenge of nodes' reliability measurement in reputation-enhanced networks, where the entry of low-reputation nodes is required for network survivability but also grants attackers a platform to launch attacks against the network.

Keywords

anonymous communication, Nym, Tor, VPN, privacy, networking

1 Introduction

Digital surveillance has become an embedded part of everyday life, posing a constant threat to individual privacy. Robust infrastructure that prioritizes user privacy is essential in mitigating the risks of network monitoring. Anonymous communication networks, such as Tor, make up an important component of this infrastructure. Tor is primarily run by volunteers, who contribute both their internet bandwidth and time without compensation for their efforts. While this approach works to a certain extent, the need for unpaid node operators may limit the number of contributors to the resulting network, which in turn may facilitate de-anonymization attacks [2, 23] by well-resourced adversaries.

1.0.1 Incentivized networks: Nym. One solution to this problem is to build *incentivized* networks that combine financial remuneration and other elements such as participant staking and reputation. An example of this design is Nym, which builds a decentralized and incentivized mixnet designed for both high- or low-latency anonymous communication. Users connect to the network via dedicated Nym client software, including a commercial VPN application called NymVPN [46]. Once connected, network traffic is relayed through up to five nodes: an entry gateway, three (optional) internal *mix nodes*, and an exit gateway.

Nym employs a novel cryptocurrency-based incentive system that compensates operators for running nodes, while also limiting traffic and payments to nodes with high reputation [14]. To join the network, each node operator is required to *stake* a quantity of tokens, a barrier designed to incentivize node quality and increase the cost of Sybil attacks. Third parties can increase this amount by *delegating* their own tokens to nodes of their choice.[1] In return, well-staked nodes with strong network performance are periodically selected into an *active set* advertised to clients, who can use these nodes to form circuits (or *paths*.) Nodes in the active set are rewarded according to their staking level and measured performance, which is intended to encourage reliable behavior.

The Nym network currently includes more than 550 nodes, distributed across 64 countries [48] with approximately 3 billion packets sent through the network every day [45]. In 2025, the Nym project launched a commercial VPN service built on top of the Nym network. While it is difficult to measure the total number of users, the Android application currently has 13,000 estimated downloads, and this is only one of the five supported platforms [1].

1.0.2 This work. While Nym has received academic analyses in the past [4, 12, 15, 22, 24, 33], most of the prior work focuses on specific components of the system, such as Nym's traffic anonymity or architecture [15, 24, 33]. In this work, we focus on a different area of the Nym system: the reputation-based algorithms that select Nym nodes for use in routing traffic. To do this, we carefully analyze the full Nym system and conduct an analysis of the current live Nym network. Based on this information, we next evaluate the feasibility of implementing various attacks on Nym's performance scoring and node selection algorithms. Our findings are based on documentation, source code analysis of the current Nym software, surveys and limited experiments conducted on the network, as well as simulations using a custom network simulator. Our analysis revealed two types of results: (1) several critical differences between the early design outlined in the technical white paper and Nym's current design, leading to increased network centralization, and (2) the existence of practical attacks on Nym's staking and node scoring reputation system. In this paper, we explore the impact of both types of attack and show that they may have significant impacts on the privacy of Nym users.

1.0.3 Scientific relevance. Nym is an important case study, if only because it is deployed and has growing usage. However, Nym's design and outcomes are also interesting from a purely scientific perspective: it is the first widely-deployed network to implement more than two decades of academic analysis of *reputation-enhanced* and incentivized mix networks [10, 11, 14, 16, 18, 19, 28]. The broad

[1] Criteria other than a node's performance may attract stake delegations, *e.g.,* an operator's past behavior or social reputation in the community.

Proceedings on Privacy Enhancing Technologies 2026(2), 277–298

https://doi.org/10.56553/popets-2026-0048

hypothesis of this research is that reputation and incentives can enhance anonymity networks, both in terms of overall performance and attack mitigation [16]. However, these early works also identify a number of non-intuitive attacks [19] that can be leveraged against reputation-enhanced mixing systems, some of which may be relevant to systems like Nym. Hence, one goal of our research is to evaluate whether attacks can be leveraged in practice, and what mitigations may exist in Nym's design to contain them.

In addition to known concerns, Nym's design adds novel challenges by incorporating the additional element of *decentralization*. Nym nodes are dynamic and can join and leave the network without the permission or control of a centralized organization: this requires a mechanism through which new nodes can join the network with only modest entry barriers, so that they can replace older nodes that are lost to technical failures and operator churn. At the same time, the entry of these nodes poses a threat: each new node offers a platform for attackers to harm existing high-reputation nodes within the network. Moreover, in contrast to a centrally-managed network like Tor, Nym's design could make it more challenging to apply corrective actions enforced by a centralized operator.[2] Navigating this tradeoff requires some non-intuitive compromises: a second goal of this research is to analyze the success of the system's design choices, and to propose alternatives.

1.0.4 Our contributions. In this work we provide the following contributions:

Design, implementation and network analysis. In the course of analyzing Nym's design, we identified a number of areas where early design and current implementation differ, leading Nym to have more points of centralization than we expected. For example, we observed that nodes' layer assignment after active set selection is performed by the Nym project itself rather than using the verifiable sampling process described in the Nym white paper [13] as the verifiable sampling process is still under development. This raises the possibility that a compromised server could place malicious nodes on each layer in attempt for fully compromised paths linking sender and receiver. We also identify some differences in the way that the NymVPN client selects servers, which could increase the vulnerability of this client as compared to the standard Nym mix network client. In addition, we conducted a one-month observational study of the network that evaluated current network's composition in topology, staking, performance scores, as well as the structure of Nym's active set.

Attacks on the Nym reputation system. Based on the data from the previous investigations, our main contribution is to explore active attacks on Nym's reputation system. Using our network data, we first considered the financial cost of a "baseline" stake attack in which a resourceful attacker attempts to control a fraction of Nym's active set by heavily staking a set of nodes. Based on observations and simulations conducted using a custom network simulator, we estimate that controlling 20% of the entire active set (with a goal of using this dominance to conduct de-anonymization attacks of fully compromised paths) would require as much as 43,460,000 NYM, representing $1,738,400 USD. To our knowledge, this is the first public work to simulate the cost of abusing Nym's staking mechanism. With this first result in mind, we next developed and evaluated a novel attack on Nym's *performance scoring* mechanism: this attack allows an attacker to seriously degrade the performance scores of existing active set nodes by selectively dropping packets and "framing" other nodes. We show that this attack, which is related to Dingledine and Syverson's "creeping death attack" [19], reduces the cost of dominating the active set by over 99% compared to the baseline attack. This reveals a serious weakness in the current process for measuring Nym nodes' reliability, and motivates improvements in this system.

Mitigations. Finally, we discuss a set of mitigations to the concentration concerns as well as stake and performance scoring attacks above. This includes an evaluation of some performance scoring design changes that were proposed to us by the Nym project after we disclosed our results.

Additional findings. Although it is not part of the main contribution of our work, we also performed network analysis to measure the network's *decentralization* by identifying clusters of nodes that are (overtly or non-overtly) likely to be controlled by the same operator. Our analysis in Appendix D quantifies the fact that substantial portions of the gateway and active sets may be controlled by specific DAOs, and that the network is reliant on a small number of hosting providers. It also revealed possible groups of co-operated nodes that are not explicitly identified as being run by the same operator.

1.0.5 Responsible disclosure and ethics. We disclosed our attacks to members of the Nym project prior to submission, and the Nym team acknowledged these attacks, provided feedback, and changed their development to try to address these attacks. We intend to continue working with the project to evaluate their newer proposals, including a decentralized network monitor [15]. The bulk of the experiments in this work were conducted in simulation; however, we also conducted several minimally invasive experiments on our own Nym node, connected to Nym's mainnet. We took care to ensure that this node would not route traffic and that any impact would be minimal. See Appendix F for details.

1.0.6 Outline of this work. In §2 we provide a brief overview of Nym, and review known attacks on Nym-like anonymity networks. In §3 we present an analysis of Nym's active set structure. In §4 we outline attacks on Nym's reputation system and in §5 we simulate these attacks. Finally, we discuss countermeasures in §6.

2 Overview of Nym

The Nym network comprises two different systems: (1) a mix network that relays traffic from a Nym client to some destination server, and (2) a decentralized blockchain called "Nyx" that assists in managing node identity, staking, and compensating node operators. In this section, we present a brief technical overview of the Nym design. This overview is primarily based on online documents from Nym [43] which are actively maintained and represent the most up-to-date information, an analysis of the Nym node source

[2] As we note in later sections, Nym (in its current realization) still retains several elements of centralization. Here we make the *optimistic* assumption that Nym's long-term implementation will eventually reflect the decentralization goals outlined in the technical white paper [13].

code [44], and the official Nym Whitepaper published in 2021 [13].[3] All monetary figures are based on the NYM token price as of May 30, 2025.

2.1 Nodes and Validators

The Nym network comprises two types of entities: (1) permissionless Nym *nodes* responsible for mixing and relaying encrypted packets, and (2) permissioned validators responsible for maintaining the Nyx blockchain, executing various node selection processes, and issuing privacy-preserving credentials. In the following, we will primarily discuss Nym's standard "*mixnet*" mode. However, Nym operates a NymVPN service that supports a fast Wireguard mode with different characteristics; we will identify sections that discuss the latter.

2.1.1 Nym nodes. Nym is designed to relay communications from a client to a destination via a *path* that comprises up to five separate relay nodes. Within each path, nodes take on specialized roles. A standard mixnet path comprises a single *entry gateway* node, three *mix nodes*, and an *exit gateway* node that routes traffic to its intended destination (see Figure 1.) Traffic flow is discussed in detail in §2.2. While all three roles can be realized by the same node software, the node operator must configure each node to take on only one of the specific roles (entry/mixnode/exit) at node startup.[4] To create or (or "bond") a new node, the operator must first register the node's public keys and metadata on the Nyx blockchain. This requires the operator to provide a minimal "stake" of at least 100 NYM tokens (approximately $4 USD), which can be recovered when the node is deactivated (or "unbonded".) Once bonded, each node can be identified by a public key, as well as a unique integer *node ID* that is assigned by the network.[5] At the start of each 60-minute epoch, a subset of the available nodes is chosen to make up a fixed-sized *active set* that (at the time of writing) comprises 50 entry gateways, 120 mixnodes, and 70 exit gateways.[6] Only nodes in the active set will be selected by compliant Nym clients to route mixnet traffic, and active set nodes are compensated in NYM tokens [14].

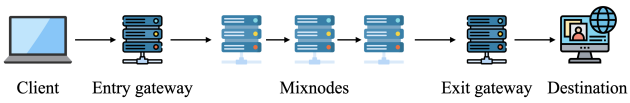

Figure 1: A typical communication path in the Nym mixnet. Note that NymVPN "fast mode" omits the mixnodes.

[3]While in many cases we granted the benefit of the doubt to Nym's documentation, in several instances we observed discrepancies between the documentation and implementation in the open source code [44]. We also relied on feedback from members of the Nym project. We will flag each such case.

[4]Nodes configured as mix nodes should not be chosen as entry/exit gateways. Nodes configured as exit gateways can be selected to route traffic as entry gateways, but not the reverse.

[5]The concept of node IDs was first introduced in Nym version 1.1.0 (2022-11-09) [40]. The IDs are assigned chronologically based on bonding order.

[6]Nym is on track to implement an Improvement Proposal in the near future: this will change the active set to 80 entry gateways, 60 mix nodes, and 100 exit gateways [30].

2.1.2 Nyx validators. Validators are responsible for maintaining the Nyx blockchain, executing various node selection processes, and issuing privacy-preserving credentials. The blockchain uses a proof-of-stake based consensus based on the Tendermint algorithm [27], and uses smart contracts to execute several Nym-critical functions. Validators also operate a public "NymAPI" that other participants can use to query network information [38]. Portions of the NymAPI are centralized and run by the Nym project; these portions handle tasks such as active set assignment. Unlike Nym nodes that use a permissionless design where anyone can operate a node, validators are currently permissioned and participation is restricted to a select set of validators approved by the Nym project.[7]

2.1.3 Active Set Selection. Nym's reputation system is based on two main components: the financial stake of a node (including stake delegated by third parties) and the measured network performance of that node. These values are used in currently deployed Nym network to periodically select active set nodes that will route traffic.

Staking. A node's total stake is composed of two parts: (1) "self-bonded" stake that a node operator invested into their own node, with a minimum of 100 NYM; (2) "delegated" stake attached to a node by third parties that do not directly operate nodes, but can invest NYM tokens into nodes of their choice and in return obtain a share of network rewards from those nodes. The rewards received by the node operator and delegates are compounded with their stake. Nym implements a *stake saturation level*, measured at 1,034,081 NYM as of March 2025, that caps the selection benefit that stake affords to any single node.[8]

Network monitor for performance scores. The performance of every Nym node is routinely measured by a centralized *network monitor* operated by the Nym project. As we discuss below, Nym has developed multiple algorithms for network scoring, but all share the same common elements: at 15-minute intervals, the monitor emulates a set of Nym clients to construct "loop" paths comprising three mixnodes and a gateway serving as both entry and exit gateway, such that each path ends at the test client. The monitor then transmits test packets via these paths, and measures the number of packets that are successfully received. The main difference between each of Nym's measurement algorithms is in (1) the way that test paths are constructed, and (2) how blame is assigned when a packet is dropped. To date, Nym has developed two deployed versions of the monitor:

(1) **NMv1**. This monitor was developed and deployed in 2022, and is currently used for active set selection. For every 15-minute testing round, this monitor works in two phases: (1) in a *pre-testing* phase, the monitor uses weighted random sampling based on nodes' pre-existing performance scores to select six *"candidate"* paths with four nodes each (the exit and entry gateway are the same node.) It then "pre-tests" those candidate paths by rapidly transmitting 1,000

[7]There are 26 validators as of August 2025.

[8]The stake saturation is determined according to the circulating supply, the supply scaling factor, and the size of the rewarded set. Absent network configuration changes, and the increase of circulating supply is gradual. As of August 6, 2025, it was 1,040,817 NYM; we used the March 2025 number in our experiments. It is worth noting that a proposed Nym Improvement Proposal suggests changing the stake saturation level to 250k NYM by adjusting the supply scaling factor [30].

probe packets via each path. If 100% of the probe packets are received, the path and nodes on that path are marked as *validated*. This process repeats for a maximum of 30 attempts until three validated paths have been selected. If the monitor fails to select even one validated path, that testing round is skipped. If it is successful, the monitor moves into a second *testing phase*: here it first randomly assigns each bonded mixnode to a layer (1, 2 or 3), and then it systematically constructs paths to test each bonded node in the system as follows: first it begins with a validated path, and then the network monitor substitutes the bonded node into the appropriate position in the path (either one mix node slot, or the gateway position.) This process is repeated for every validated path such that each bonded node is placed on three distinct test paths. Each substituted node on a validated path is marked as "test node".[9] The monitor then relays three packets through each path, and scores the test node based on the fraction of packets received vs. expected. In this design, sole blame for any relay failure during the testing phase is placed on the test node, reflecting the assumption that the rest of validated nodes on the path are "known good" following the pre-testing phase.

(2) **NMv2**. This monitor was developed and deployed in 2024, but the score it produces is *not* currently used in active set selection. As of the time of writing,[10] work on this later design and NMv1 has been halted due to the attacks in this paper, and implementation has started on the decentralized design of network monitor [15]. Here the monitor simply selects many random four-node paths from the set of all bonded nodes (where the exit and entry gateway are the same node in each path) and sends encrypted packets to itself every 15 minutes. If the monitor receives the packet, all nodes on the path receive a *complete path count*; otherwise, they all receive an *incomplete path count*. A node's performance score for every 15-minute test period is the ratio of complete paths counts to the total complete and incomplete paths counts. In this approach, blame for any relay failure is therefore distributed equally across all nodes in a given path.

In both approaches, this testing process is repeated every 15 minutes, resulting in a performance score in the range $[0, 1]$ for that period. These scores are then averaged over time: a node's performance score is computed as the average of all the per-test-period scores it has received over the past 24 hours.[11]

Active set selection process. After collecting nodes' stake from Nyx's smart contracts as well as performance scores from the network monitor, a component of NymAPI that is centrally run by the Nym project calculates the *active set selection weight* for each node. This value is calculated based on a *stake ratio S* which represents the node's total stake divided by the stake saturation level (and is capped at 1), and a *performance score P* in the range [0,1].[12] These are combined using the following equation:

$$\mathbf{W}_{\text{selection}} = S \cdot (P)^{20} \tag{1}$$

The centralized NymAPI then performs a weight-based sampling process to place nodes into the gateway and mixnode active sets, and assign mixnodes to the three *layers*. We discuss the security implications of this centralized set selection in Section 2.3. Validators receive the selection transaction from NymAPI and include it in the blockchain to make it effective. After each epoch, all active set nodes are rewarded with NYM tokens funded from a "mixmining" pool reserved at token launch.[13] We describe a future reward mechanism proposed by Nym in Appendix B.

2.2 Traffic Flow

After selecting a communication path, Nym mixnet clients send each packet to some destination address using the layered Sphinx encrypted packet format [9]. To enable responses from a destination node, Nym clients transmit single-use reply blocks (SURBs) which are pre-computed Sphinx packet headers encoding a route that ends in the gateway address of the client that created the SURB.

2.2.1 Selecting paths and fixing gateways. Nym mixnet clients sample paths from the nodes in the active set. To facilitate long-lived (e.g., TCP) connections, the client will first sample a distinct entry and exit gateway to act as "mailboxes" that remain fixed over many packets. For every packet sent, the Nym client then samples three internal mix nodes according to their *layer* (*i.e.,* layer one nodes are used for the first mix hop, and so on.) Clients continuously send encrypted packets following a Poisson process with constant rate, even when no real data is available to send, in which case *cover traffic* is sent to the client through a path that ends in the client's entry gateway (*loop* route.)

2.3 Nym Project as a Trusted Third Party

In this work, we are mainly interested in the resilience of Nym to broad attacks on the reputation system. However, we must also note the possibility of narrower attacks aimed at compromising the centralized servers that operate portions of Nym's processes: these include the centralized network monitor and a portion of the NymAPI that controls the active set selection process. The risk here is that an attacker who compromises these servers will have near-total control over the active set, with all the security implications that this implies. For the purposes of this work, we view these as temporary features, and note that the Nym project has already developed plans to address these vulnerabilities: for example, the Nym technical white paper describes a decentralized process for active set selection, using verifiable randomness beacons [13]. Recent

[9] Critically, this testing process also involves testing nodes in the validated paths, meaning that a "validated" node can find itself occupying up to two different positions in the path simultaneously: one as a "known good" relay, and one as a test node.

[10] December, 2025.

[11] We verified this behavior by analyzing source code and through limited network experiments (Appendix F.) We then confirmed the details with Nym scientists. We also learned that Nym is evaluating designs for a new network monitor, which we discuss in §6.

[12] A node's performance score P is computed as the product of a configuration score and a "routing score" measured by the network monitor. The configuration score is itself based on various node parameters (including recency of the node's software version); in our observations, this is 1 for nearly all nodes in the network, and we will omit the details here. In practice, the performance score P is essentially equal to the score measured by the network monitor.

[13] This pool contains a bootstrapping reserve of locked funds initialized at 250 million NYM. The pool emits approximately 5,000 NYM per epoch, of which around 2,000 NYM are distributed among active set nodes, with undistributed rewards remaining in the pool.

work by Nym scientists proposes a decentralized implementation of the network monitor [15].

2.4 NymVPN

NymVPN is an application built on top of Nym's network. It can be configured to run in "mixnet" mode using Nym's standard five-hop paths, or "fast/Wireguard" mode where Wireguard packets are routed through only two nodes: an entry and exit gateway. Critically, the NymVPN client's route selection algorithm differs from Nym's standard approach in both modes, such that it completely disregards the contents of the gateway active set and allows any correctly configured gateway nodes with a performance score above 0.5 to be included in the list of available servers [37]. Moreover, users must select either a region (or a specific server) for each of their entry and exit gateways. When the user selects a region for these gateways (which represent the only nodes used in "fast" mode), the VPN randomly samples a gateway within the region for the user (or two distinct gateways if the region is the same for exit and entry.)[14] This raises the possibility that an attacker might insert many colluding entry/exit gateway nodes into the NymVPN gateway set at a minimal staking cost of 100 NYM ($4 USD) each; this could allow an attacker to inexpensively dominate a large fraction of the NymVPN gateway set (globally or within specific regions), and potentially control the full path of some users who choose Wireguard mode. We illustrate this in Figure 5 (located in Appendix C) where six out of seven servers in Switzerland region are run by a single decentralized autonomous organization (DAO).

3 Analyzing Nym's Active Set Composition

Although Nym is architecturally designed to be a decentralized system, in practice, it is possible for any network to experience patterns of centralization that could undermine network anonymity. In Nym, only nodes in the active set are responsible for routing real user traffic. Consequently, the concentration of routing power within a small number of node clusters in the active set could pose a significant privacy risk as user traffic may be routed through a path where all five nodes are controlled by the same entity.

Recall that Nym uses a reputation system based on stake and performance score to select nodes into the active set. Thus, to investigate the extent of decentralization in Nym's active set and how each factor of the reputation system determines the likelihood of a node being selected, we recorded the contents of the active set nodes as well as their stake and performance scores once every hour for a 30-day period from June to July of 2025. We used the SpectreDAO Nym Network Explorer [48] as the most complete source for network information.[15] This system provides an API that distributes real-time information about Nym nodes, performance scores, stake, and current active set.

Overall, throughout the time of data collection, we identified 562 bonded nodes on the network, spreading across 64 countries and 107 recognized hosting providers. Our network analysis shows that 8 countries and 5 hosting providers collectively hold 50% of the total nodes. In terms of total stake-weight, the network currently has approximately 175 million NYM staked across all bonded nodes.

Specifically for active sets, over the 30-day period, 453 out of 562 total nodes on the network (approximately 80%) were selected into the 240-node active set at least once. Although no single node is present in every active set during our observation period, out of 435 nodes that are selected for at least once, approximately 33% of them are present in the active set for more than 70% of epochs, demonstrating that there is still a strong tendency for a core group of nodes to be repeatedly selected.

We discussed in §2.1.3 that a node's performance score is prioritized in active set selection weight as it is raised to the 20^{th} power. When we examine the performance score of every node that has been selected into the active set for at least once over the 30-day period, we find that, as expected, only nodes with high performance scores are selected. However, when we examine their stake distribution, nodes with a total stake exceeding 100,000 NYM dominate the active set, while those with lower stakes are rarely selected despite representing a moderate portion of the network. Thus, high stake-weight appears to be an implicit requirement to join the active set, indicating that packet routing power is highly concentrated. Upon further investigation, we found that most highly-staked nodes belong to large node clusters, which are primarily run by DAOs that have substantial funding and social reputation for operating as business services. We provide a further analysis on identifying node clusters and evaluating active set compositions in Appendix D, and discuss mitigations to concentration concerns in §6.0.1.

4 Attacks on Nym's Reputation System

Previous literature has proposed using reputation systems in mix networks to measure *mix* reliability, including specifying a set of behaviors that characterize a "good" or "bad" mix, or creating economic incentives for nodes to behave reliably [16]. Nym combines both of these approaches by taking into account a node's performance score and its stake when selecting the active set. This raises the possibility that a malicious party could bias either measure in order to elect its own nodes into the active set.

4.0.1 Overall goal: de-anonymization. In the following sections we discuss two different attacks on Nym, both of which propose different methods for the attacker to gain control of a significant fraction of Nym's active set. The means for obtaining control of this fraction differ between the two approaches, but the ultimate goal of the attacker remains the same: the attacker waits for a target client to traverse the network to some destination via a path that contains attacker nodes in specific positions. At this point, the attacker can use its privileged network position to link the client and the destination. More concretely, all of our attacks will consider two possible objectives:

(1) **A***A attacks.** In a first objective, the attacker might hope to control (at least) the entry and exit gateways in a path. For services that operate similarly to NymVPN's "fast mode" (using only gateways), this completely de-anonymizes connections. For five-node connections in the Nym mixnet, this scenario serves as a necessary stepping stone to a full de-anonymization attack, as we discuss below.

[14] A user may override the random selection by the client through manually selecting an entry and an exit gateway

[15] This Network Explorer has been used officially within Nym community. The alternative is Nym Explorer, which has less features.

(2) **AAAAA attacks.** In a complete attack on the mixnet, the attacker controls all five nodes between the client and the destination. This attack is straightforward: given even one packet sent from client to destination (or vice versa), the attacker can link the pair.

Moving from **A***A** *to* **AAAAA**. A critical observation about Nym's mixnet is that, for interactive applications (such as TCP/IP sessions), the entry and exit gateways are selected and held fixed for relatively long-lived connections, while internal mix nodes are re-sampled for every packet sent. This suggests a dual strategy wherein an attacker controls a fraction of both the gateway set *and* the mixnode set. In the first phase of the attack, each new client connection creates an opportunity for the attacker to achieve the **A***A** condition. Once that condition has been attained, every packet sent by the client provides a fresh opportunity for adversarial nodes to be selected for all mixnodes on the path (we denote the additional condition as ***AAA***.) When this second condition is achieved for even one packet, this provides the attacker with the full **AAAAA** condition necessary for de-anonymization.

Therefore, evaluating these attacks requires us to consider two separate parameters: the minimum number of *connections* N_c that we expect a client to make (*i.e.,* the number of times it re-samples its gateways); and separately the minimum number of *packets* N_p that will be sent to a destination of interest within any given connection. Similar to an issue identified in prior work by Oldenburg, Acar, and Diaz in PETS'22 [32] that generating many fresh circuits enables an adversarial middle relay to discover a victim's guard node in Tor, Nym's frequent resampling of mixnode paths for every packet allows an attacker to have client traffic routed through one of their controlled paths relatively quickly. We discuss the details of selecting appropriate amount of N_c and N_p to use for simulations in §5.2.2.

4.1 Baseline Stake Attacks

Nym incorporates financial staking as part of the criteria in determining which nodes will be selected into the active set. This implies an "obvious" attack strategy in which a well-resourced attacker runs many highly-staked nodes in order to dominate the active set.[16] While this is not a novel attack idea, we consider it here to establish a cost baseline to compare against alternative attack methods.

4.1.1 Requirements. Any Nym node that is unaffected by intentional misbehavior is likely to receive a near-perfect performance score. Certainly, the overwhelming majority of nodes have near-maximum performance scores in Nym, as shown in Figure 8. Hence, the main contributor in calculating the selection probability is simply the amount of stake a node has. Based on our one-month analysis of the active set in Section 3, nodes with minimum stake are unlikely to ever be selected into the active set. Through analyzing all the nodes in 720 active sets during the data collection, our result shows that active set nodes have 425,000 NYM ($17,000 USD) in stake on average. This creates a high barrier of entry for new nodes, and the routing power concentrates on a few groups of nodes with extremely high stakes, usually run by well-funded DAOs. This can be considered a "feature": if the existing highly-staked nodes behave honestly, then the cost of inserting new adversarial nodes is extremely high. Naturally, this can also be viewed as a weakness: if the existing highly-staked nodes behave adversarially, then it is very unlikely that honest but low-resourced volunteers will be able to dilute their power.[17]

If an attacker wishes to control a large fraction of the active set, the most straightforward approach is to simply stake a large number of NYM tokens on a set of nodes they control. However, this requires substantial capital. Since Nym's *stake saturation level* places a cap on the useful stake any individual node can be given, the maximum advantage that the attacker can achieve depends on the number of nodes they operate, as compared to non-attacker-controlled nodes who are also at the stake saturation level. For instance, our simulations indicate that for an attacker to control 20% of the entire active set to enable fully compromised path de-anonymization, they would require at least 43,460,000 NYM (approximately $1,738,400 USD). We discuss these estimates and our methodology more extensively in Section 5.

4.2 Performance Scoring Attacks

One way to reduce the cost of the previous attack is to exploit the fact that each node's performance score is raised to the 20^{th} power. This means that a slightly higher relative performance score can compensate for a substantial stake difference between competing nodes. As a simplified example, a node with 100% performance but only at 50% of stake saturation level has a much greater chance of being chosen (selection weight is $1.00^{20} \cdot 0.5 = 0.5$) than a node with 90% performance but saturated stake (selection weight is $0.90^{20} \cdot 1.0 = 0.122$).

A high performance score can be achieved by simply relaying packets effectively. However, as noted previously, most nodes already possess near-perfect scores. This raises a second possibility: malicious nodes might attempt to artificially lower the scores of other nodes by *framing* other nodes.

4.2.1 Framing attacks in Nym. Nym's performance monitor designs make the network highly vulnerable to framing attacks. Recall that in all versions of Nym's network monitor, the monitor must send test packets that plausibly resemble real client packets whenever such test traffic is present (if this requirement is not adhered to, a malicious node can only relay test packets while refusing to route real traffic.) This requires the monitor to route traffic through multiple nodes that — from the perspective of a node being tested — resemble a standard Nym path. However, in doing so, the monitor takes on a new risk: an adversarial node at any other location in the path can also drop a packet, and in the process "frame" other innocent nodes. The challenge for the monitor is to identify an appropriate method to penalize the various nodes in each path when relay failures occur.

[16] Executing this attack may pose practical challenges as well: the NYM token has limited market depth, and so large purchases of NYM might result in a visible price spike, or could require the attacker to purchase or steal tokens directly from large stakeholders. On the other hand, while such attack effects may be noticeable, it is not clear what corrective actions could be taken to halt the attack.

[17] The internal operations of these DAOs remains opaque to us. In this work we assume pessimistically that all DAO nodes are potentially controlled by the same operator, but in practice this might not be true.

Blame allocation generally follows one of two paradigms. At one end of the spectrum (exemplified by Nym's NMv2 algorithm), all nodes on a test path may be assigned an equal share of blame, since the monitor has no way to determine which specific node dropped the packet. The obvious drawback of this approach is that a single minimally-bonded malicious node can cause score penalties to multiple victim nodes, at the cost of receiving only one unit of score penalty for itself. At a different point on the tradeoff space, blame can be allocated more selectively based on other criteria, such as behavior in past testing rounds. An extreme example of this approach is embodied by Nym's NMv1 algorithm, which performs a *pre-testing* phase to measure previously well performing nodes, and then assumes these nodes as "known good" helpers to be used while testing the remainder of the network. The weakness of this approach is that past node behavior is not necessarily predictive of the node's honesty during any subsequent period: a malicious node can easily alter its behavior between the pre-testing and testing phases. Specifically, if the malicious node can cause the monitor to identify it as a "known good" node during the pre-testing phase, then it can easily drop packets during the subsequent testing phase, and these failures will be entirely attributed to innocent victim nodes.

The flaw in both approaches is that a malicious node can impart a disproportionately larger *aggregate* score reduction on the entire network than it receives itself. This effect is reminiscent of the "creeping death attack" first observed by Dingledine and Syverson [19]. A second feature of Nym's scoring algorithms amplifies the impact a malicious node can have: since the network monitor incorporates both high-reputation active-set nodes and minimally-staked nodes that may have just been created into its test paths, an attacker can create essentially disposable and inexpensive "sacrifice" nodes and use them to inflict serious performance score degradation on well-staked nodes that are currently in the active set. We now outline concrete attacks on each of Nym's two network monitor architectures.

NMv1. To exploit the NMv1 architecture, an attacker must create a set of mixnodes (S) that will selectively drop packets, and a set of "attacking" gateway nodes (A) that will not drop packets. Both sets must be configured with minimal-to-moderate stake (the exact stake values for the attack vary, as we will discuss in §5.) Since in either case these nodes are not highly-staked, each node's probability of being selected into the active set is initially low, and any node that is (improbably) selected into the active set should simply refrain from participating in the attack. This ensures that for the remaining S nodes, *all* received packets will be test packets. Before launching any attack, each colluding node should relay packets correctly for a period of at least 24 hours, with the result that all will achieve a perfect (or near-perfect) performance score.

The attack requires that each S node must be able to recognize and alter its behavior between the pre-testing and testing phases. For S nodes that are selected into candidate paths during pre-testing, recognizing pre-testing is straightforward: pre-testing is marked by a rapid stream of 1,000 packets all sent via the same path (i.e., from the same predecessor node to the same successor.) All S nodes should initially assume they are in pre-testing, and thus should relay all packets correctly to maximize the probability that they will be selected into a validated path. When an S node identifies that it has relayed this stream of 1,000 packets, the node will now optimistically assume that it has been selected into a validated path and the monitor has now entered the testing phase.[18]

In the testing phase, each of the S nodes selected into a validated path will switch to a strategy of selectively dropping packets. Note that since testing covers all nodes *including* the validated nodes, each validated node will also be treated as a node-under-test during one test run: it could therefore appear at one or two different positions in the path.

Nodes' selective dropping has three cases: (1) whenever the validated S node recognizes the IP address of a fellow (non-validated) S or A node in the previous or subsequent position, it should relay a test packet, hence avoiding harm to another colluding node. (2) when the validated S node identifies that it has been placed at a different position in the path, it should relay the test packet (this indicates that itself is likely the node under test.) Finally (3) in all other cases, the validated S node simply drops the packet. Broadly, this approach avoids most cases that result in score degradation to the validated S node or any other colluding node, albeit not every possible one. For example, a fellow colluding node may be under test in a path position that is two hops away from the validated S node and hence will be harmed by a packet drop. Alternatively, the validated S node may have been selected for testing and randomly assigned to exactly the same position it normally sits in. Since the validated S node will drop the packet in these cases, an S node will occasionally suffer some score penalty. These cases comprise a minority of the full set of possible eventualities, and hence the overall penalty applied to the non-colluding victim nodes (V) will be many times greater.

It is worth noting that this attack is self-reinforcing: since candidate paths are identified by first selecting the nodes with the highest performance scores, any attack that lowers the score of the non-colluding nodes in the network will tend to improve the probability that a colluding node is chosen into a candidate path during subsequent pre-testing phases.[19]

NMv2. To exploit the NMv2 architecture, we use a slightly different approach. First, an attacker creates two sets of nodes: a set of minimally-staked sacrifice nodes (S) configured as mixnodes and a set of minimal to moderately-staked "attacking" nodes (A) (whose exact configurations depend on attack objectives as we will explain in §5.) Then, attacker controlled nodes relay packets correctly for at least 24 hours to achieve a high performance score before the attack. And finally, S nodes proceed to selectively drop packets as the following.

Recall that packet encryption should make test packets indistinguishable from real or cover traffic *in principle*. In practice, however, freshly-created nodes are not in the active set and thus receive no real or cover traffic.[20] While nodes cannot decrypt packets, they can

[18] If a node *is* selected as for a validated path, the signature of subsequent testing packets is recognizable. To bolster their confidence, nodes can access a NymAPI endpoint that indicates the identities of nodes that are selected for the validated path.

[19] Adversarial nodes may also use attempt to *prevent* the selection of any validated paths (by sabotaging the pre-testing phase), causing the testing phase to be skipped. We did not investigate the implications of this approach.

[20] We confirmed this by operating a minimally-staked node on the Nym network; the only packets we observe correlate with the cadence of performance monitor test packets.

observe the IP address of the preceding and subsequent relay nodes. If either node has an IP within the list of A nodes, the sacrifice node relays the packet as expected; otherwise, it simply drops the packet. This approach may occasionally harm some A nodes when they are in more distant positions within the path, but the attack will still cause greater damage overall to victim nodes in V than to A nodes.

Discussion. Although the attack strategies for NMv1 and NMv2 will produce a result within one epoch, the attacker must continue the attack for several hours to achieve the full effect: this is due to the fact that Nym's performance monitor averages node performance scores over a 24-epoch window. At the conclusion of this attack period, the victim nodes (set V) should exhibit dramatically lower performance scores than the attacking nodes A, which will enable each A node to have an increased probability of being selected into the active set, despite having a lower stake per node.

While we have not executed either of the two attacks described above against the Nym network for reasons of both ethics and cost, we have conducted limited experiments on the Nym node software and mainnet to verify that (1) the node software can be modified to support selective packet drops, and (2) previous/subsequent IP addresses are available to the relevant code, (3) failure to relay packets impacts node performance scores in the manner expected based on documentation and source code, and (4) that freshly-created sacrifice nodes are indeed incorporated into paths containing nodes in the active set. We discuss the results of our network experiments in detail in Appendix F.

5 Simulating Attacks

To evaluate the attacks described in the previous section, we developed a custom Nym simulator and conducted a series of experiments to evaluate the feasibility and cost of deploying each of our two attack strategies. For each attack approach, we then evaluate the difficulty of achieving different attacker objectives, within specific time and resource constraints. These objectives are (1) achieving an **A***A** positioning on the client/destination path, and (2) achieving an **AAAAA** positioning on the path.

To establish a baseline, we evaluated the stake-only attack of §4.1 in which a resourceful attacker stakes a large number of NYM tokens on its own nodes. We then simulated the performance-score framing attack described in §4.2. For both attack methods, we considered a range of specific attack parameters (*e.g.,* desired active set fraction, number of attacker nodes, etc.), and then identified specific parameters that optimize our de-anonymization success. Finally, we compared the cost of the baseline and framing attacks.

5.1 Simulation Design and Setup

5.1.1 Implementing the simulator. We built a custom simulator in 1000 lines of Python code that simulates node creation, test packet path generation, packet dropping, score and probability calculation, active set selection, and an attack orchestrator. Approximately 600 lines of additional code handle data analysis and plotting. While many of the simulators used in previous studies on mixnets have been based on the general-purpose framework *Mixim* [4], our reputation attacks can be evaluated using the much more lightweight simulation described below.

Our simulator supports simulations for NMv1, NMv2, and a new proposal given to us by Nym following our disclosure, which we will call NMv3 (see §6.0.2 for more details.[21]) At runtime, users specify the version of network monitor, the attack objective (**A***A** or **AAAAA**), and which attack to execute (stake or framing.) Our simulation first initializes a set of victim nodes V with stake and performance parameters that mirror one snapshot[22] of the Nym mainnet. We then initialize a set of attacking nodes A and a separate set of sacrifice nodes S. Each of these nodes begins with the simulation with a 0.98 initial performance score, while initial staking amounts used for these nodes depend on the attack strategy used in different attack objectives and network monitor versions, as we discuss further below. To model framing attacks, the simulator samples "test" paths through the full collection of nodes according to the specified version of the network monitor, with each test path representing a simulated test packet sent through the network. The simulator then determines whether the packet should traverse the path, by emulating the packet-dropping strategies described in §4.2. Based on these results, the simulation then recalculates node performance scores after every simulated 15-minute testing round and and computes the active set using the process described in §2 after every simulated 60-minute epoch.

The goal of our simulation is to evaluate continuous attacks that occur over multiple epochs. To accommodate score calculation algorithms where scores from previous testing rounds could affect the next, our general purpose simulator evaluates every testing round and every epoch sequentially. To emulate Nym's scoring algorithm (which averages scores over the most recent 24 epochs), we compute the most recent 24-epoch average of performance scores after every testing round before moving on to the next. (Note that if $n < 24$ we use each node's initial pre-attack score as the value for the first $24 - n$ epochs.) To remove variability, we average our results over 100 complete simulated attacks. Simulating the performance scoring attack with varying numbers of S and A nodes in our scenarios required 10 to 12 hours of wall-clock time.

5.1.2 Attack configurations. Since the baseline attack simply relies on staking and is independent of performance scoring mechanisms from network monitor, we simulated it within the NMv2 environment. For framing attack, we simulated it across NMv1, NMv2 and NMv3. In both the baseline and framing attacks, we simulated all combinations of the following stake amount and S, A node counts. For stake values in NYM, we tested [100, 1,000, 10,000, 100,000, and 1,000,000] as 100 is the minimum amount of NYM required for each Nym node and 1,000,000 was used as an approximation of the stake saturation amount 1,034,081. For node counts, in baseline attack, we tested every combination of S and A nodes where each node group takes values from range 100 to 1000 in intervals of 50. For framing attacks, each node group takes values from range 10 to 200 with intervals of 10. We selected these smaller ranges and finer intervals due to the fact that the framing attack was successful with a much smaller number of nodes than the baseline staking attack; for example, a framing attack with 200 nodes can allow an attacker to control 100% of the active set.

[21] Note that this is not an official Nym designation, we use this term only for this work.
[22] Snapshot taken at 2025-06-10 19:01.

5.2 Simulation Results

Our simulation parameters and methodologies apply to NMv1, NMv2, NMv3. In this section we discuss the results in two officially deployed network monitors NMv1 and NMv2, and compare the findings between these two versions. We include further simulation findings of NMv2 in Appendix G and discuss simulation results of the NMv3 countermeasure in Section §6.

5.2.1 Measuring attack success. The primary goal of each attack is for the simulated attacker to elevate its A nodes to occupy a fraction of the active set. Since the active set is subdivided into gateway and mixnode sets, we use f_{gw} to denote the fraction of total adversarial gateway nodes in the gateway active set, and f_{mix} to denote the fraction of total adversarial mixnodes in the mixnode active set.

In our experiments, we will consider a range of possible values for f_{gw} and f_{mix}, and then evaluate the costs to achieve each one.

Estimating the needed fractions f_{gw}, f_{mix}. Of course, this raises a more important question: *what fraction f_{gw}, f_{mix} does the attacker need?* To answer this, we must perform a calculation that differs for our two attack objectives. In this analysis, we break down the **AAAAA** attack into its constituents: **A***A** and ***AAA***. We then evaluate how varying fractions of f_{gw}, f_{mix} will affect the expected probability that the client encounters an **A***A** or ***AAA*** path after selecting exactly one path in either scenario. From this probability, we can determine the number of unique connections (resp. packets) the client will need to make in order for this condition to occur with some arbitrary target probability $Pr(S_\gamma), \gamma \in \{\text{A***A, *AAA*, AAAAA}\}$, where $Pr(S_{AAAAA}) = Pr(S_{A***A}) \cdot Pr(S_{*AAA*})$. We can then use this information in two ways: in one direction, we can calculate the expected number of connections N_c (resp. packets N_p) that a client must create/send in order to achieve $Pr(S_\gamma)$ when controlling different fractions of the active set. In the other, we can fix $Pr(S_\gamma)$ and some reasonable bounds on N_c, N_p, then calculate the needed fractions f_{gw}, f_{mix} that an attacker must control in order to achieve success. Note that this calculation is independent of whether the attack is a stake-only or performance-scoring attack, and hence we can separate this calculation from our more detailed simulations of those attacks.

Here, we formulate the relationship between f_{gw}, f_{mix}, N_c, N_p, and $Pr(S_\gamma)$. First, we use $Pr(A_{gw})$, $Pr(A_{mix})$ to denote the average probability of having an **A***A** or ***AAA*** path, which is affected by f_{gw} or f_{mix}. Then, we calculate the number of unique *connections* N_c or *packets* N_p that a client needs to generate for one of them to route through one instance of an **A***A** or ***AAA*** path with success probability $Pr(S_\gamma)$, as determined by the following:

$$\begin{gathered} 1-(1-\Pr(A_\alpha))^{N_\beta} \geq \Pr(S_\gamma), \\ \alpha \in \{\text{gw}, \text{mix}\},\ \beta \in \{\text{c}, \text{p}\},\ \gamma \in \{\text{A***A, *AAA*}\}. \end{gathered} \tag{2}$$

We explore the relationship between all the variables in more detail in Appendix E.

5.2.2 Comparing attack costs. Although both the baseline (staking) attack and framing attack can each allow the attacker to control a fraction of the active set, the resources required for the two attacks may differ substantially. To measure this, we compare the minimum total monetary costs (which consist of non-refundable virtual private server hosting costs per node,[23] and refundable stake including the minimum bonding stake per node and additional stake invested into the node) required to achieve both of our attack objectives **A***A** and **AAAAA**.

Selecting parameters. We used the approach outlined in §5.2.1 to calculate target values f_{gw}, f_{mix} that achieve our target objectives. Here we set $Pr(S_{A***A}) = 0.5$ for **A***A** objective as this probability only requires an amount of gateway connection resampling, N_c, that we consider to be realistic to achieve within our attack time frame (i.e. one day) and with a relatively small f_{gw} value (i.e. below 0.2) to limit total adversarial nodes required. We set $Pr(S_{*AAA*}) = 0.99$ so the **AAAAA** objective has a combined $Pr(S_{AAAAA}) = 0.5 \cdot 0.99 = 0.495 \approx 0.5$.[24] We selected a high $Pr(S_{*AAA*})$ value because a high value for ***AAA*** is very achievable in a short amount of time considering that mixnodes paths, N_p, are resampled much more frequently than gateways are resampled in the **A***A**. Also, this would allow the overall $Pr(S_{AAAAA})$ to be similar to $Pr(S_{A***A})$. To make sure that N_c, N_p are within realistically achievable amount and the resulting f_{gw}, f_{mix} needed are relatively small, we set $N_c = 50$ after experimentation with clients, including NymVPN,[25] where we observed frequent reconnections in a short time period due to network connectivity issues. We set $N_p = 1000$: this value represents 20 seconds of client transmission at the maximum client transmission rate, or approximately 1.5 MiB of data transfer at the maximum theoretical Maximum Transmission Unit (MTU) that Nym can support. Given these selected parameters, inputting them into Equation 2 yields $f_{gw} = 0.119$ and $f_{mix} = 0.173$. These two fractions correspond to $0.119 \cdot 120 = 14.28$ gateway nodes in the gateway active set and $0.173 \cdot 120 = 20.76$ mixnodes in the mixnode active set. Since the number of nodes must be an integer, there needs to be 15 gateway nodes and 21 mixnodes, which results in $f_{gw} = \frac{15}{120} \approx 0.13$ and $f_{mix} = \frac{21}{120} \approx 0.18$.[26] Therefore, 0.13 and 0.18 are the fractions we will use in our costs evaluations below.

Evaluating costs. For each de-anonymization objective, we simulated every combination of S, A, *stake* within the range described in §5.1.2 and recorded the f_{gw}, f_{mix} that can be achieved with each combination under baseline attack and framing attack. First, among all S, A, *stake* combinations that can achieve the target f_{gw}, f_{mix}, we find the one combination with the lowest total cost as "optimal set". In this case, the amount of S, A nodes is unconstrained and minimizing attack cost is the sole goal. Then, for framing attack, we evaluated an additional attack *with constraints* method where we limit the total number of S, A nodes that the attacker can create when launching the performance scoring attack. Specifically, among all S, A, *stake* combinations that achieve the target fraction, we find the combination with the lowest total amount of S and A nodes that can still result in savings compared to baseline attack. Our goal in this approach is to consider a secondary attacker goal in which the attacker attempts to make the insertion of nodes less

[23] We use $20 as a fixed VPS rental cost for our 24-hour attacks.

[24] We denote $Pr(S_{AAAAA})$ value as $\tilde{0.5}$ in Table 1.

[25] We use NymVPN purely as an example of how frequently a productized client will lose connectivity, even though NymVPN uses a different node selection process.

[26] We round *up* the fractions to the next two decimal places to ensure that the attack objectives can be successfully achieved.

conspicuous to network observers by reducing the number of nodes while still allowing some savings compared to stake-only attacks.

Simulation results. We separately present our results for NMv1 and NMv2. Table 1 illustrates the results of our simulations of a 24-epoch attack on NMv1 (see Appendix G for similar results on the other monitor algorithms.) In NMv1, for the **A***A** attack objective, performance scoring attacks can save 99% over stake-only attacks, with total costs reduced from $254,400 USD to $2,800 USD. Even under the scenario with constrained attacker node counts, the performance scoring attack still produces a savings of 68%. For the **AAAAA** objective, performance score attacks save 99%, reducing the cost from $1,484,000 USD to $4,200 USD; the attack with constraints still saves 81%.

In NMv2, for the **A***A** objective, the framing attack requires even less cost and fewer nodes than the attack on NMv1. However, for the **AAAAA** attack objective, the costs and required nodes ends up being slightly higher (150 nodes NMv1 vs 220 nodes in NMv2.) The primary reason for the difference in the number of required nodes is that, in NMv1, S nodes can largely drop packets without being blamed. As a result, they may be selected into the mixnode active set, which means that no additional A mixnodes are required to achieve the f_{mix} goal. However, in NMv2, since S nodes always receive blame for dropped packets, the attack requires additional A nodes to control the same fraction of the active set.

The high costs observed in the stake-only attack validate Nym's observation that financial stake can indeed deter traditional Sybil attacks. However, the amounts required for even the stake-only attack still appear to be within a viable range for a well-resourced adversary such as a national security agency.[27] The dramatic cost reduction we see in evaluating performance score framing attacks could lower the barrier to the point where such attacks are within the range of moderately-funded individuals.

5.2.3 Varying target f_{gw}. Thus far, we've focused on comparing costs between different attack methods for a fixed fraction of the active set. Next, we focus on de-anonymization objective **A***A** to investigate how different fractions of attacking nodes in the gateway active set would affect the total cost of the attack in NMv1 and NMv2. In Figure 2, we compare the minimum costs required for each f_{gw} in the baseline attack against the same target in the performance scoring attack. The graph shows that across different f_{gw} values, costs for our performance scoring attack in both NMv1 and NMv2 remain consistently and substantially lower compared to the baseline, even if an attacker has more ambitious goals of controlling a much higher f_{gw} in the active set.

5.2.4 Varying target epochs. All of our attack simulations have considered only the result at the *conclusion* of a 24-epoch attack run. This setting gives the attacker maximum advantage, since the performance monitor averages per-epoch scores over a 24-hour period. However, this does not mean that all 24 epochs are necessary to achieve a substantial result. To investigate the actual time required, we next investigate how the fraction of A nodes in the active set changes as a multi-epoch attack proceeds. Specifically, we simulated the performance scoring attack on both NMv1 and

[27] It is also worth noting that most of this cost is in *stake*, which an attacker can reclaim even if they behave badly (subject to network changes by the Nym Project.)

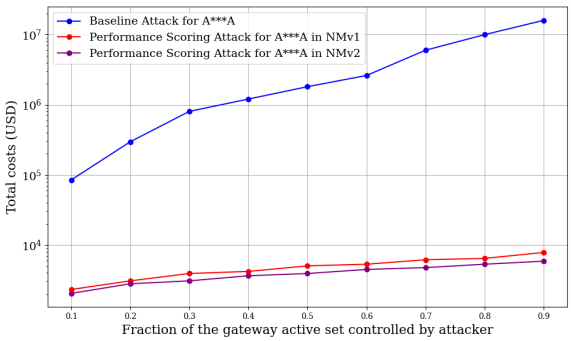

Figure 2: Minimum total costs to achieve a broad range of f_{gw} (fraction of the gateway active set controlled by attacker) values for the A*A objective across 100 simulations. Both baseline and framing attacks on NMv1, NMv2 are shown. Note that y-axis is on log based scale.**

NMv2 with the **A***A** objective over 24 epochs, while measuring changes in f_{gw} at the conclusion of each epoch. Here for NMv1, we use an attack configuration in which $|V| = 562$, $|A| = 30$ and the stake for each A node is 1000 NYM; we change only $|S|$, considering 60, 80, and 100 nodes. We set the S values based on our optimal set results in Table 1. The results on NMv1 are shown in Figure 3, and results on NMv2 are shown in Figure 15 in Appendix G. In both cases, we make two observations: (1) f_{gw} increases slowly during the first several epochs, while rising rapidly until it reaches (or remains in some cases) at the maximum value where all A nodes are selected into the active set; (2) given the same number of A nodes, the rate of which f_{gw} increases is influenced by different attack settings, such as the number of S nodes. Comparing NMv1 and NMv2, the attack progresses more quickly in NMv2 than in NMv1, and this could due to the fact that an S node in NMv2 can harm three other nodes on one path while it can only harm one node on each path in NMv1.

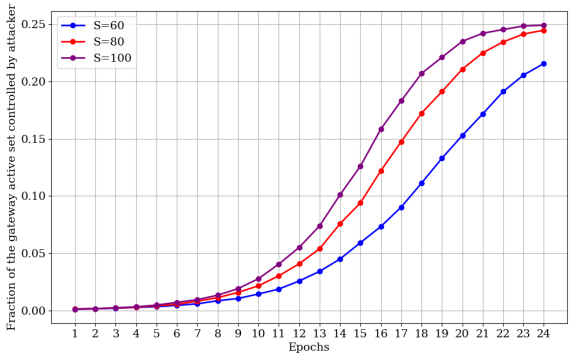

Figure 3: Simulation of the framing attack on NMv1 showing f_{gw} increasing over multiple epochs. Attack results are averaged across 1000 simulations. Here $|V| = 562$, $|A| = 30$ and each A node is staked at 1000 NYM, while $|S| \in \{60, 80, 100\}$. (Note that f_{gw} cannot exceed 0.25.)

Table 1: Two attack objectives, each with three example 24-epoch attack scenarios on NMv1. N_c is the minimum number of connections that will be made by the target client (resulting in selection of fresh gateway pairs.) N_p refers to the minimum number of packets that the target client is expected to send through each connection (if applicable.) f_{gw} and f_{mix} indicate the target fractions of attacker-controlled gateways and mixnodes in their respective active sets. $Pr(S_Y)$ refers to the attacker's probability of success for an A*A or AAAAA path. $|S|$, $|A|$ represent the size of the attacker's node sets. Costs include the USD cost of both stake and node hosting costs (shown as refundable, non-refundable.) Stake refers to the stake of each node (shown as stake per S node, A node.) We evaluate three attacks: our baseline "stake-only" attack; the performance scoring attack with unrestrained nodes amount that results in the lowest attack cost; and a variant that constrains the number of total nodes to the lowest amount that can still result in savings.**

Attack Parameters						Results					
Goal	N_c	f_{gw}	N_p	f_{mix}	$Pr(S_Y)$	**Method**	**Cost ($)**	**Savings**	$\|S\|$	$\|A\|$	**Stake (NYM)**
A***A	50	0.13	n/a	n/a	0.5	stake	254,400 (242,400, 12,000)	n/a	0	600	(0, 10,000)
						framing	2,800 (800, 2,000)	99%	80	20	(100, 100)
						w/ constraints	81,040 (80,240, 800)	68%	20	20	(100, 100,000)
AAAAA	50	0.13	1000	0.18	$\tilde{0.5}$	stake	1,484,000 (1,414,000, 70,000)	n/a	0	3500	(0, 10,000)
						framing	4,200 (1,200, 3,000)	99%	130	20	(100, 100)
						w/ constraints	282,960 (281,160, 1,800)	81%	70	20	(100,000, 1,000)

6 Countermeasures

Our analysis of Nym's reputation system reveals several issues: the concentration of routing power among a few well-staked node clusters as we outlined in §3, and the vulnerabilities in the design choices for Nym's staking and performance monitoring mechanism as we discussed in §4 and §5. Many of these issues can be mitigated, and in this section we discuss the ramifications of those design decision changes.

6.0.1 Mitigating staking concentration. To lower the barrier of entry for new nodes and address centralization issues, Nym created an Improvement Proposal in June 2025 suggesting a reduced stake saturation level of 250,000 NYM [30], with the change scheduled to take place on September 2, 2025.[28] This change is well motivated but presents two major implications. First, it could encourage existing large DAOs to run additional nodes by reallocating stake that is no longer required from previously fully staked nodes. Thus, the concentration concern would still persist along with higher likelihoods of collusion in which user traffic is routed through nodes from the same cluster. Second, this could potentially lower the costs for both our baseline attack and framing attack: to obtain at least the same active set selection weight as victim nodes, adversarial nodes with at least equal (if not higher) performance scores require lower stake per node.

6.0.2 Mitigating framing. Framing attacks are the most powerful and cost-effective attacks we considered. There are several relatively simple changes, including some already under consideration by Nym, that may help to address the specific attacks we identify; however, we argue that durable fixes may be more challenging.

Alternative blame attribution. The ideal approach to mitigating framing is to identify a means of attributing blame *only* to adversarial nodes. Unfortunately, this appears to be challenging when multiple possibly-adversarial nodes may exist within a testing path. In the absence of perfect blame attribution, an obvious alternative is to use additional information, such as past node behavior, to attribute blame. As we have demonstrated in our analysis of NMv1, this approach (taken to extremes) carries risks: it permits an adaptive attacker to alter their strategy between testing rounds, gaining even more power to harm victim nodes in the process. However, this does not mean that NMv1 is the only strategy that could be applied. After disclosing our initial results to members of the Nym project, we learned that Nym is considering alternative scoring approaches to attributing blame to nodes on a failed test path, and one approach has been implemented in production code [42] but this has not been deployed to the network monitor. In this work, we will refer to this countermeasure as "NMv3".

The NMv3 approach mirrors NMv2's strategy of testing random 4-node paths, but adds additional logic. Here, each node is assigned a *fail count* which records the number of consecutively failed test packets for each node; the count resets to zero once the node is detected on a complete test path. Nodes are penalized when they receive a fail count of more than two: all such nodes receive an incomplete path score as in NMv2.[29]

The danger with this strategy is that, *under the assumption that an attacker can identify test packets*,[30] this attacker may be able to exploit the predictable nature of the score calculation, which allows an attacker to control the pattern of packet dropping to avoid being the only node that gets blamed on a failed path, and in the process to frame other nodes. Specifically, the attack will proceed as follows. Recall that newly-bonded sacrifice nodes are never expected to be in the active set so all packets received by such a node can be reasonably assumed to be test packets. Since the scoring approach only attribute blame to nodes with more than two fail counts, an

[28] This change has been fully implemented in Nym as of November, 2025.

[29] If no node has more than two fail counts, all nodes on the path receive one incomplete score as before.

[30] This may seem like a strong assumption, however it will easily be true in two cases: (1) for nodes not in the active set, all traffic comprises test packets, and (2) gateway nodes can always see the IP address of the source/destination, which means that detecting test traffic simply requires an attacker to learn the IP addresses of the network monitor.

attacking node can simply drop two consecutive packets and relay the third packet, ensuring that its fail count will always be reset to zero before continuing to drop packets. For victim nodes however, if they already been marked with two consecutive failed packets and their next test packet path happens to include an attacking node with a fail count less than two, the attacking node will drop the packet, but only the victim node will be penalized. Even in the worst case where the attacking node receives an incomplete path count because there's no identifiable "guilty" node, all of the nodes in the path will be penalized identically to the attacking node. Thus, for every packet the attacker drops, all nodes on the path receive equal blame or *only* the victim nodes with more than two fail counts receive the blame. This allows the attacker to inflict more aggregate damage to the set of victim nodes than to itself.

To validate these findings, we simulated the attack using our simulator *under the assumption that the remaining details of the network monitor remain unchanged from the most recent network monitor (NMv2) design.* We refer to our attacker nodes as S nodes and grant them only minimal stake. We also deploy a second set of well-staked A nodes (gateways in the **A***A** objective, and both mixnodes and gateways in the **AAAAA** objective.) Our simulation results are shown in Figure 4; here we compare the countermeasure (NMv3) framing attack costs to achieve a wide range of f_{gw} for the **A***A** objective against baseline attack, and framing attacks on NMv1, NMv2. Our simulation results demonstrate that although the framing attack on NMv3 costs slightly more than NMv1 or NMv2, the attack is still very effective. We show additional results comparing NMv3 with NMv1 and NMv2 for a fixed goal in Table 5 in Appendix G.

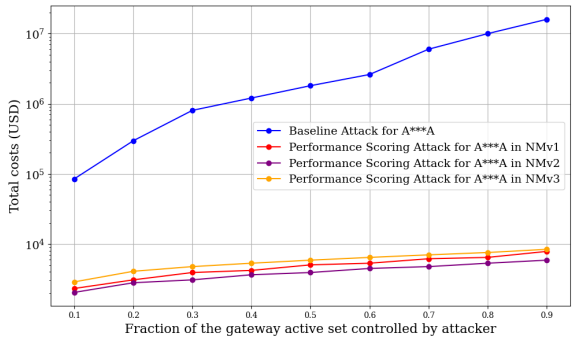

Figure 4: Minimum total costs to achieve a broad range of f_{gw} (fraction of the gateway active set controlled by attacker) values for the A*A objective across 100 simulations. Both baseline and framing attacks on NMv1, NMv2, NMv3 are shown. Note that y-axis is on log based scale.**

Separating high- and low-reputation nodes. Another obvious problem with Nym's current performance monitoring is that a performance monitor must evaluate existing (high-reputation) nodes for continued good behavior, while also measuring the performance of potential new (low-reputation) entrants. Nym's current design combines the measurement of both types of nodes within test paths, exposing high-reputation nodes to attacks by low-reputation nodes.

After disclosing our results to members of the Nym project, we learned that Nym is considering a new version of the performance monitor that involves running two monitors to separate the testing for active set nodes and non-active set nodes. For active set nodes, the first network monitor will follow the current approach, but using paths comprising *only* active set nodes. For non-active-set nodes, the second network monitor sends test packets directly to each node, which guarantees that any dropped packets are attributed solely to that node.

This approach prevents non-active set nodes from framing active set nodes, which should substantially increase the cost of framing attacks, although it may not eliminate them. The basic problem for any path-based test strategy remains: the aggregate score damage to honest (victim) nodes can be a multiple of the aggregate effect on the attacking nodes. An attacker could still launch attacks using well-staked nodes that are selected into the active set. Moreover, the nature of attacks launched from the active set is that success can beget further success: as an attacker forces victim nodes out of the active set, the stake-weight of the active set will be reduced. This may allow an attacker to elevate additional colluding nodes into the active set, which further increases the effectiveness of the attack. Any such attack strategy would be extremely complex and would depend on the intricate details of the final network monitor proposal, which we do not have at this time. Hence, we must leave the project of simulating the cost of this new attack as future work.

Finally, we discuss the impact of combining previously mentioned alternative scoring strategy with this new network monitor design. Here, only active set nodes are included in test paths, and hence any test traffic will therefore be mixed with real and cover traffic. Superficially, this makes simple modulation of dropping behavior much more difficult, since the attacker may not know which packets to drop. However, as a defense, we believe this is relatively fragile: if an attacker develops a means to distinguish test packets from real packets, then the predictable scoring behavior becomes immediately exploitable. To illustrate one possible attack vector: gateway nodes are able to directly examine the IP addresses of clients, which means that a malicious gateway must *never* be able to determine whether a "client" is actually the network monitor. We are uncertain about whether the new network monitor can exclude this possibility, but it exemplifies the fragile nature of traffic assumptions. Since we lack a final design for the new network monitor, we cannot empirically evaluate how these scoring systems will work in that setting.

Stake slashing. A critical feature of the Nym system is that nodes do not risk their stake due to malicious activity. Hence, attacking nodes experience no direct financial penalty for poor performance, other than a short-term opportunity cost.[31] Future versions of the Nym reputation system could address this problem by "slashing" the stake of misbehaving nodes. We have received similar suggestions from Nym scientists. Unfortunately, designing this countermeasure is challenging and subtle, and could have broad implications for the entire Nym network. First consider a mixnet setting without adversarial presence. Honest high-reputation nodes can have the most to lose from slashing since simple accidents or downtime

[31] Naturally, an attack that harms the network could lower the value of NYM, but this is an indirect mechanism.

from node updates could cost them large amounts of stake. Next, consider a mixnet that contains adversarial nodes. Given that stake transactions in Nym are processed rapidly (effective by the next epoch), a misbehaving node operator could reallocate their stake or unbond their node before slashing occurs to prevent any stake loss. Given specific versions of network monitor (i.e. NMv1) and any blame allocation strategy where adversarial nodes can escape the blame on a failed test packet path, stake slashing would not effectively prevent framing attacks at all and would mostly harm honest nodes. Therefore, any stake slashing strategy must be carefully designed to avoid compounding the challenges identified in this work, since an adversary who exploits the testing system may be able to permanently harm honest nodes.

Monitoring. An important countermeasure to framing attacks is to monitor the network regularly to identify large clusters of nodes that experience sudden performance score drops that occur within one or several epochs. Detailed examination of performance monitor logs might also reveal the creation of sacrifice nodes. In principle, Nym project could then act to ban the attacking nodes through *e.g.,* software updates or the publication of IP ban lists. We are not aware of any systematic effort to monitor the network for these attacks; hence, one of our main recommendations is for Nym to develop and maintain that capability. However, monitoring may not present further information beyond the fact that Nym could be under attack. A downside to banning nodes based on monitoring results is that in framing attacks, both honest nodes and adversarial nodes will experience sudden performance score drops. Thus, it is hard and arbitrary for Nym to distinguish whether the cluster of nodes that experience sudden score drops are the victim of the framing attack or the attackers.

6.0.3 Broader discussion of framing attacks. The fundamental challenge that underlies our framing attacks as well as many prior work on node reliability measurement in mixnets [11, 16, 19, 28] is the inherent difficulty in accurately determining which node is responsible for a failed packet. In this section, we draw on our analyses of NMv1, NMv2, and various countermeasures to discuss the broad tradeoffs and limitations of different approaches for identifying the node(s) responsible for failed packets, with the goal of determining if there's a general framework that can lead to the most promising results.

Here, we categorize different approaches based on these two metrics: (1) the nature of information used when determining the responsible node, and (2) the size of the set of nodes that the approach can narrow down to when attributing blame. For the first metric, we categorize blame-attribution approaches according to whether they are "current" such that they rely solely on present-time behavior of nodes during testing, or whether they are "historical" such that they rely on prior behavior (i.e. historical performance scores) to determine how to allocate blame for a testing failure. For both approaches, we must also then consider the number of nodes n in the set for blame attribution ($1 \leq n \leq 4$.)

Upon a failed test packet path, in NMv1, it is always one node that gets the blame (i.e. the "test node"), and test nodes in main testing are selected based on a node's previous testing results. In NMv2, all nodes on a path always receive blame equally, but no historical performance is used to allocate blame. In both countermeasures that we discuss in §6 (alternative blame attribution, separating active and inactive set nodes), the basis of blame remains historical as it is based on either previous consecutive fail counts or the assumption that a node has performed well enough to be selected into the active set. The set of candidate nodes for blame is dynamic in both countermeasures as it ranges from one to four.

The primary reason that framing attacks remain possible in NMv1 and NMv3 is that they rely on the assumption that historical information can reliably predict how nodes will behave on subsequent tests. While this assumption may be effective in a network environment that experiences natural packet loss, we have shown with simulation results that these assumptions can be extremely dangerous in adversarial settings, where adversaries can exploit the assumption's predictive nature by strategically adjusting their behavior to evade blame and frame honest nodes. These approaches may superficially reduce the size of the candidate set of guilty nodes, but only at the cost of excluding the actual guilty parties. Blame attribution based on current behavior appears to be a better approach, but this can only be effective if the monitor can narrow down the size of the set of potentially-guilty nodes to a small enough set that framing attacks are not effective. As we've shown with NMv2, penalizing the entire set of nodes on a failed path is extremely inefficient and vulnerable to attacks.

One lesson from our findings is that predictable behaviors by the network monitor enables relatively straightforward exploitation strategies. One recommendation is for each testing round to randomly choose a strategy (i.e. either consecutive fails, or being in the active set) when assigning blame. Given the randomness and unpredictable blame attribution, adversaries may be less likely to determine an optimal framing strategy.

7 Related Work

Mix networks first proposed by David Chaum [7] established a foundation for many subsequent anonymous communication systems [6, 17, 21, 36]. Traditional mixnets considered cascade and free-route topologies [18], which have both shown risks to anonymity [18, 50] compared to the layered topology in Nym. Earlier mixnets sent packets in batches [18], while Nym implements a stop-and-go design [8, 26]. More recent works have systematically analyzed different approaches to measure decentralization in blockchains [34]. Many previous works have considered traffic analysis and website fingerprint attacks [3, 29, 51, 52]. Although cover traffic can mitigate these attacks to some extent [24], recent works have proposed techniques that can fingerprint traffic even in the face of delays and cover traffic [35, 51]. Other work considers node reliability issues that harm or improve privacy [5, 28]. For a good overview of privacy in existing systems, see [47].

8 Conclusions

In this work we analyzed Nym and identified several design tradeoffs and potential areas for improvement. Nym's design should inspire future work that considers novel techniques for computing node performance scores that resist attacks by malicious node operators, and different routing strategies that mitigate de-anonymization attacks while maintaining performance.

References

[1] AppBrain. 2025. NymVPN: Secure VPN by Nym. Available at https://www.appbrain.com/app/nymvpn-secure-vpn-by-nym/net.nymtech.nymvpn.

[2] Liviu Arsene. 2015. De-anonymization of Tor Hidden Services With 88 Percent Certainty, Researchers Say. Available at https://www.bitdefender.com/en-us/blog/hotforsecurity/de-anonymization-of-tor-hidden-services-with-88-percent-certainty-researchers-say.

[3] Adam Back, Ulf Möller, and Anton Stiglic. 2001. Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In *Proceedings of the 4th International Workshop on Information Hiding (IHW '01)*. Springer-Verlag, Berlin, Heidelberg, 245–257.

[4] Iness Ben Guirat, Devashish Gosain, and Claudia Diaz. 2021. MiXiM: Mixnet Design Decisions and Empirical Evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society* (Virtual Event, Republic of Korea) *(WPES '21)*. Association for Computing Machinery, New York, NY, USA, 33–37. https://doi.org/10.1145/3463676.3485613

[5] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. 2007. Denial of Service or Denial of Security? How Attacks on Reliability Can Compromise Anonymity. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*. ACM, Alexandria, VA, USA, 92–102. https://doi.org/10.1145/1315245.1315258

[6] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri De Ruiter, and Alan T. Sherman. 2017. cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations. In *Applied Cryptography and Network Security*, Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi (Eds.). Springer International Publishing, Cham, 557–578.

[7] David L. Chaum. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (Feb. 1981), 84–90. https://doi.org/10.1145/358549.358563

[8] George Danezis. 2004. The traffic analysis of continuous-time mixes. In *Proceedings of the 4th International Conference on Privacy Enhancing Technologies* (Toronto, Canada) *(PET'04)*. Springer-Verlag, Berlin, Heidelberg, 35–50. https://doi.org/10.1007/11423409_3

[9] George Danezis and Ian Goldberg. 2009. Sphinx: A Compact and Provably Secure Mix Format. In *2009 30th IEEE Symposium on Security and Privacy*. 269–282. https://doi.org/10.1109/SP.2009.15

[10] Norman Danner, Danny Krizanc, and Marc Liberatore. 2009. *Detecting Denial of Service Attacks in Tor*. Springer-Verlag, Berlin, Heidelberg, 273–284. https://doi.org/10.1007/978-3-642-03549-4_17

[11] Anupam Das, Nikita Borisov, Prateek Mittal, and Matthew Caesar. 2014. Re3: relay reliability reputation for anonymity systems. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security* (Kyoto, Japan) *(ASIA CCS '14)*. Association for Computing Machinery, New York, NY, USA, 63–74. https://doi.org/10.1145/2590296.2590338

[12] Debajyoti Das, Claudia Diaz, Aggelos Kiayias, and Thomas Zacharias. 2024. Are continuous stop-and-go mixnets provably secure? *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2024, 4 (2024), 665–683. https://doi.org/10.56553/popets-2024-0136

[13] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. 2021. *The Nym Network: The Next Generation of Privacy Infrastructure*. Technical Report. Nym Technologies SA. https://nym.com/nym-whitepaper.pdf

[14] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. 2022. Reward Sharing for Mixnets. *Cryptoeconomic Systems* 2, 1 (June 2022). https://cryptoeconomicsystems.pubpub.org/pub/diaz-reward-sharing-mixnets

[15] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. 2024. Decentralized Reliability Estimation for Low Latency Mixnets. arXiv:2406.06760 [cs.CR] https://arxiv.org/abs/2406.06760

[16] Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. 2001. A Reputation System to Increase MIX-Net Reliability. In *Proceedings of the 4th International Workshop on Information Hiding (IHW '01)*. Springer-Verlag, Berlin, Heidelberg, 126–141.

[17] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: the second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13* (San Diego, CA) *(SSYM'04)*. USENIX Association, USA, 21.

[18] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. 2004. Synchronous batching: from cascades to free routes. In *Proceedings of the 4th International Conference on Privacy Enhancing Technologies* (Toronto, Canada) *(PET'04)*. Springer-Verlag, Berlin, Heidelberg, 186–206. https://doi.org/10.1007/11423409_12

[19] Roger Dingledine and Paul Syverson. 2002. Reliable MIX cascade networks through reputation. In *Proceedings of the 6th International Conference on Financial Cryptography* (Southampton, Bermuda) *(FC'02)*. Springer-Verlag, Berlin, Heidelberg, 253–268.

[20] Tor Forum. 2024. Tor Forum: Why don't we encourage Tor relays in East Asia? Available at https://forum.torproject.org/t/why-dont-we-encourage-tor-relays-in-east-asia/13674/3.

[21] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. 1996. Hiding Routing Information. In *Proceedings of the First International Workshop on Information Hiding*. Springer-Verlag, Berlin, Heidelberg, 137–150.

[22] Iness Ben Guirat and Claudia Diaz. 2022. Mixnet Optimization Methods. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2022, 3 (2022), 456–477. https://doi.org/10.56553/popets-2022-0081

[23] Alex Hern. 2016. US defence department funded Carnegie Mellon research to break Tor. Available at https://www.theguardian.com/technology/2016/feb/25/us-defence-department-funding-carnegie-mellon-research-break-tor.

[24] Daniel Hugenroth and Alastair R. Beresford. 2023. Powering Privacy: On the Energy Demand and Feasibility of Anonymity Networks on Smartphones. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 5431–5448. https://www.usenix.org/conference/usenixsecurity23/presentation/hugenroth

[25] Ishan Karunanayake, Nadeem Ahmed, Robert Malaney, Rafiqul Islam, and Sanjay K. Jha. 2021. De-Anonymisation Attacks on Tor: A Survey. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2324–2350. https://doi.org/10.1109/COMST.2021.3093615

[26] Dogan Kesdogan, Jan Egner, and Roland Büschkes. 1998. Stop-and-Go-MIXes Providing Probabilistic Anonymity in an Open System. In *Information Hiding*, David Aucsmith (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 83–98.

[27] Jae Kwon. 2014. Tendermint: Consensus without mining. Available at https://tendermint.com/static/docs/tendermint.pdf.

[28] Hemi Leibowitz, Ania M. Piotrowska, George Danezis, and Amir Herzberg. 2019. No Right to Remain Silent: Isolating Malicious Mixes. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1841–1858. https://www.usenix.org/conference/usenixsecurity19/presentation/leibowitz

[29] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright. 2004. Timing Attacks in Low-Latency Mix Systems. In *Financial Cryptography*, Ari Juels (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 251–265.

[30] Nym Forum. 2025. NIP-2: Change stake saturation and active set to improve network decentralisation and efficiency. https://forum.nym.com/t/nip-2-change-stake-saturation-and-active-set-to-improve-network-decentralisation-and-efficiency/1400

[31] Nym Project. 2025. Nym Operator's Guide. Available at https://nym.com/docs/operators/introduction.

[32] Lennart Oldenburg, Gunes Acar, and Claudia Diaz. 2022. From "Onion Not Found" to Guard Discovery. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2022, 1 (2022), 522–543. https://doi.org/10.2478/popets-2022-0026

[33] Lennart Oldenburg, Marc Juarez, Enrique Argones Rùa, and Claudia Diaz. 2024. MixMatch: Flow Matching for Mixnet Traffic. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2024, 2 (2024), 276–294. https://doi.org/10.56553/popets-2024-0050

[34] Christina Ovezik, Dimitris Karakostas, Mary Milad, Daniel W. Woods, and Aggelos Kiayias. 2025. SoK: Measuring Blockchain Decentralization. In *Applied Cryptography and Network Security: 23rd International Conference, ACNS 2025, Munich, Germany, June 23–26, 2025, Proceedings, Part I* (Munich, Germany). Springer-Verlag, Berlin, Heidelberg, 184–214. https://doi.org/10.1007/978-3-031-95761-1_7

[35] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. 2014. Do Dummies Pay Off? Limits of Dummy Traffic Protection in Anonymous Communications. In *Privacy Enhancing Technologies*, Emiliano De Cristofaro and Steven J. Murdoch (Eds.). Springer International Publishing, 204–223.

[36] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. 2017. The Loopix Anonymity System. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1199–1216. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska

[37] Nym Project. 2025. Nym Node Setup and Run. Available at https://nym.com/docs/operators/nodes/nym-node/setup.

[38] Nym Project. 2025. NymAPI. Available at https://nym.com/docs/apis/nym-api.

[39] Alfredo Rial and Ania M Piotrowska. 2023. Compact and Divisible E-Cash with Threshold Issuance. *Proceedings on Privacy Enhancing Technologies* 4 (2023), 381–415.

[40] Nym Technologies SA. 2022. Nym Code v1.1.0. Available at https://github.com/nymtech/nym/tree/release/v1.1.0.

[41] Nym Technologies SA. 2025. Anonymous Replies with SURBs. Retrieved April 30, 2025 from https://nym.com/docs/network/traffic/anonymous-replies

[42] Nym Technologies SA. 2025. Nym alternative scoring. https://github.com/nymtech/nym/blob/drazen/reward-simulator/nym-api/src/support/storage/manager.rs#L1589

[43] Nym Technologies SA. 2025. Nym Docs. https://nym.com/docs

[44] Nym Technologies SA. 2025. Nym GitHub Repositories. https://github.com/nymtech

[45] Nym Technologies SA. 2025. NymAPI. Available at https://mainnet-node-status-api.nymtech.cc/v2/mixnodes/stats.

[46] Nym Technologies SA. 2025. NymVPN. Available at https://nym.com/features.

[47] Sajin Sasy and Ian Goldberg. 2024. SoK: Metadata-Protecting Communication Systems. In *Proceedings on Privacy Enhancing Technologies (PoPETs 2024)*, Vol. 2024. Sciendo, 509–524. https://doi.org/10.56553/popets-2024-0030
[48] SpectreDAO. 2025. Nym Network Explorer. Available at https://explorer.nym.spectredao.net.
[49] Nym Core Team. 2025. Nym's token cycle: Next step, Magura. Available at https://nym.com/blog/nyms-virtuous-token-cycle.
[50] Maximilian Weisenseel, Christoph Döpmann, and Florian Tschorsch. 2025. The Last Hop Attack: Why Loop Cover Traffic over Fixed Cascades Threatens Anonymity. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2025, 2 (2025), 382–397. https://doi.org/10.56553/popets-2025-0067
[51] Simon Wicky, Ricardo Ferreira Ribeiro, Ania M. Piotrowska, and Carmela Troncoso. 2024. Empirical evaluation of traffic fingerprinting against the Nym mixnet. Retrieved April 30, 2025 from https://nymtech.net/uploads/Nym_WFP_Paper_5_58a1105679.pdf Manuscript.
[52] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. 2004. On Flow Correlation Attacks and Countermeasures in Mix Networks. In *Proceedings of Privacy Enhancing Technologies Workshop (PET 2004)*, Vol. 3424. Springer, Berlin, Heidelberg, 207–225. https://doi.org/10.1007/11423409_13

Acknowledgements

We would like to thank Claudia Diaz and Harry Halpin for reviewing and commenting on our work. This work was supported by a Security and Privacy research award from Google. The authors used generative AI-based tools to review the text for grammatical errors and to assist in mechanical tasks related to LaTeX formatting.

A Known Attacks on Nym

Below we discuss several known attacks on Nym, which are generally also broad attacks on anonymous communication networks such as Tor. We then discuss Nym's approach to addressing these attacks. Here we consider a threat model with both passive adversaries who can observe network traffic and active adversaries who can manipulate network settings or operations.

A.1 Network Observation

The most common form of passive attack on relay networks assumes an adversary that can observe network traffic to de-anonymize traffic patterns, such as timing attacks and website fingerprinting. The most significant variable in these attacks is typically the number of points that the attacker can simultaneously monitor.

Packet timing correlations and traffic fingerprinting. Timing attacks allow a passive adversary to correlate encrypted network packets traversing two observation points [25]. A related *traffic fingerprinting* attack looks at the timing and packet counts/size to correlate traffic types and flows.

Nym mitigates these threats by using fixed-size packets, cover traffic, and random delays at each relay node. Both delays and cover traffic are sampled by the client; delays are contained in per-packet the meta-data sent to each relay [22]. While arbitrary delays can be chosen in principle, the client code specifies an average of 15 milliseconds. Nodes enforce these delays by maintaining a queue of packets to forward, ensuring that the order of incoming packets is not necessarily the same as the order of outgoing packets. Nym's approach represents an intermediate option between the batch processing strategies used in classical mixnets [7], and the zero-delay approach used by Tor.

A recent work by Oldenburg, Juarez, Argones Rùa and Diaz in PETS '24 examined the possibility of matching flows between entry and exit gateways in the Nym system [33]. In this attack, an adversary running a set of entry and exit gateways uses deep learning classifiers based on packet metadata including timestamps to match ingress and egress traffic from a set of gateways. Assuming that the adversary controls *all* the gateways of a set of clients communicating in a closed group, it can successfully deploy the attack; if any of the clients involved in the exchanged flows is sending or receiving traffic outside the monitored group, however, the attack quickly degrades. This attack analyzed unmodified "natural" traffic, and did not apply any artificial watermarks to the metadata; hence it likely represents a lower bound on what is possible if a malicious entry gateway also adds additional delays as a side channel. Although cover traffic and packet drops occurring in live Nym network reduce the effectiveness of such flow matching attacks, Nym is not immune to them.

A.2 Active Attacks

Active attacks require an adversary to actively participate or manipulate the network, including modifying traffic patterns or packet contents, and operating compromised nodes or services.

Watermarking. Watermarking attacks [25] are a form of correlation attack where an active attacker that controls the first and last node on a path manipulates packets to introduce a recognizable pattern that can survive the intermediate relay nodes. This allows an attacker to correlate the sender with the receiver and de-anonymize traffic.

Nym resists changes to the encrypted packet contents due to its use of the non-malleable Sphinx packet format [9].[32] While Sphinx rules out the possibility of watermarking attacks that alter the *content* of packets, it does not prevent an entry gateway from inducing other watermarks such as timing channels that could mark packets from a specific client. Nym's forwarding delays may obfuscate this timing data to some extent, but the flow correlation results of Oldenburg *et al.* [33] indicate that even natural traffic timing signals can remain detectable in Nym traffic, which strongly indicates that artificial signals will be as well.

Exploiting reply blocks. A variant of the watermarking attacks described above exploits *single-use reply blocks* (SURBs.) This assumes an attacker who controls a set of entry and exit gateways and wishes to identify the gateway used by a specific client [41]. Since each SURB has a limited size, a sender can generate multiple SURBs and include them in the payload if they are expecting longer responses. If a reply is longer than the total size allowed by the SURBs sent by the client, the receiver can use the reply to request the client to send more SURBs. Assume there exists a malicious service provider, R, on the receiving end of the network. When R receives a packet with SURBs from an arbitrary client, C, it utilizes those reply blocks to repeatedly request additional SURBs from C. The client, in turn, continues to respond by supplying more SURBs as requested. After gathering a large amount of SURBs, R sends back all the accumulated SURBs to C and monitors the entry gateway

[32] Sphinx defines an encryption mechanism that prevents packet-tampering by ensuring that encrypted packets each contain a MAC ("integrity tag") and are non-malleable. Since every node on the path will verify the integrity tag before relaying the packet, manipulating the content at any point on the path will cause later (honest) nodes to discard the packet.

traffic permutation in order to determine which gateway C is using. To carry out the attack, the active adversary needs to be able to run a malicious service that can repeatedly request more SURBs, and to actively monitor traffic from many entry gateways.

B Rewards Scheme

Currently, all the nodes in the active set are considered to contribute equally, and their final received rewards are proportional to their stake saturation level and measured performance. To facilitate a future usage-based reward scheme, Nym has implemented anonymous credentials based on compact e-cash with threshold issuance by the validators [39]. Specifically, clients with a valid NymVPN subscription use the associated key pair to obtain bundles of these credentials, and then "spend" them with entry gateways to demonstrate that they are legitimate users. The gateways then collect these credentials that indicate the bandwidth they will route, and get rewarded proportionally.[33]

C NymVPN Switzerland Gateways List

Here in Figure 5, we show that a user who selects Switzerland will be forced to use one of seven gateways on the list.[34] Six out of seven are run by a single decentralized autonomous organization (DAO). And even the addition of one malicious server will result in that server being selected approximately 12.5% of the time.

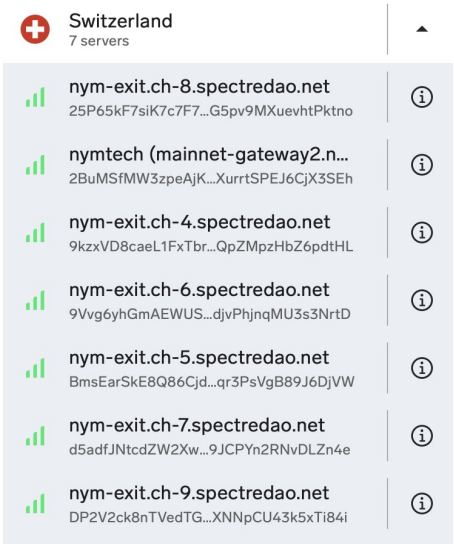

Figure 5: Complete list of NymVPN gateways available in Switzerland, as of August 4, 2025.

D Analyzing the Network Composition

The design of an anonymity network tells only part of the story. To evaluate the level of centralization in the Nym network, we conducted an analysis of the live network as it was constituted from June to July of 2025.

D.0.1 Approach. Although Nym is architecturally designed to be decentralized, in practice, it is possible for any network to experience patterns of centralization that could undermine network anonymity. Several data sources exist that can give a view of Nym's network structure. After evaluating the completeness of various sources, we identified the SpectreDAO Nym Network Explorer [48] as the most complete source for network information. This system provides an API that distributes real-time information about Nym nodes, performance scores, stake, and current active set.

To investigate the extent of decentralization in the live Nym network, we collected live data from the Network Explorer, twice per day over a 30-day period. We observed nodes' naming conventions and presence patterns to identify indicators of a set of nodes being (overtly or covertly) controlled by the same operator, and evaluated the hosting providers used by all bonded nodes. Additionally, we recorded the contents of the active set once every hour during this period to evaluate set composition, stake, performance scores, and other common characteristics among those nodes.

D.0.2 Overview of the network. Throughout the time of data collection, we identified 562 bonded nodes on the network, spread across 64 countries and 107 recognized hosting providers. Our network analysis shows that 8 countries and 5 hosting providers collectively hold 50% of the total nodes. In terms of total stake-weight, the network currently has approximately 175 million NYM staked across all bonded nodes.

D.1 Node decentralization

It is extremely challenging to determine how "decentralized" a permissionless network is, since centralization may not produce visible effects. Nonetheless, there are several ways to identify clusters of nodes that may be co-operated by the same entity.

D.1.1 Identifying co-operated nodes through self-namings. Nym nodes are encouraged to set a custom *moniker*, or alias, and provide a link to their node operating business. These additional identifiers allow node operators to publicly advertise their infrastructure, extend their reputation to all their nodes, and attract stake delegations.

We analyzed the monikers and website links associated with each node to identify clusters of nodes with shared naming conventions or domain references that indicate a common operator.[35] We show the top five clusters in Table 2.[36] The largest node family, LunarDAO, operated 32 out of 562 total bonded nodes on the network at the time of data collection (approximately 5.7%.) While this may not seem significant at first glance, 31 of these nodes are running as gateways, taking up 12.8% of the entire gateway set, which indicates a much higher level of centralization and increased risks of routing traffic through entry and exit gateways from the same cluster. Several other top node clusters follow the same pattern. Together, the top five node clusters make up approximately 28% of the entire gateway set. When we focus on the contents of the *active set*, we find that this gateway percentage can rise as high as 34.3%, with LunarDAO alone holding 11.4% of the active set at some epochs. These numbers

[33] This mechanism is not yet in place.
[34] List as of August 4, 2025.
[35] For instance, `nym-exit.ch-3.spectredao.net`, `nym-exit.ch-4.spectredao.net` clearly indicate that they are both run by SpectreDAO.
[36] Data used in this analysis is recorded on July 9, 2025, and is subject to change.

Table 2: Top five node clusters by moniker. T is the total number of nodes run by an operator. P_T is the percentage of the total network. P_G is the percentage of the entire gateway set, while P_A refers to the percentage of the gateway active set at 2025-06-10 19:01:04.

Node Operator	T	P_T	P_G	P_A
lunardao	32	5.7%	12.8%	11.4%
daoariwas	18	3.2%	4.5%	10%
bwnym	17	3.0%	6.6%	10%
no trust verify	17	3.0%	0%	0%
tupinymquim	15	2.7%	4.1%	2.9%
TOTAL	99	17.6%	28%	34.3%

highlight a significant degree of potential shared control in the extant network.

D.1.2 Presence patterns. The Network Explorer queries the NymAPI, which periodically communicates with each node to monitor their presence and collect relevant data. Nodes reachable by the NymAPI will be present on the Network Explorer. Several reasons could contribute to a node not being listed in this presence data. First, if a node unbonds, the node and its node ID will disappear from the Network Explorer forever. Second, if a node experiences downtime, it will temporarily disappear from the API but may reappear after it resumes normal operation. Nodes can also become unreachable due to IP or firewall misconfiguration.[37]

During the 30-day analysis period, we analyzed node appearances and disappearances from the presence data set. We specifically focused on patterns where multiple nodes shared identical disappearance and appearance behaviors, and identified several such cases, which we illustrate in Figure 6.

Discussion. When multiple nodes disappear, and later re-appear together, this provides some evidence that they may be co-operated by the same entity.[38] Even in these cases, there remains the possibility that such synchronization could occur by chance. We examined these node clusters from appearance/disappearance patterns by manually analyzing the node identifiers and dividing these cases into three scenarios, where we interpret each as reflecting a different degree of occurring by chance unless all nodes within the cluster share the same namings.[39] Scenario (1) covers presence patterns where nodes are absent for a timestamp, indicating an absence for less than 24 hours. Scenario (2) covers presence patterns where nodes are absent for more than one continuous timestep. Scenario (3) covers presence patterns with two or more intermittent gaps. Out of these three scenarios, we believe that scenario (3) carries the lowest likelihood of being pure coincidence: the existence of repeatedly synchronized disappearance gaps suggests co-operation rather than chance. However, it remains possible that those synchronized gaps could due to unrelated nodes all operating on the same hosting providers.

To validate this methodology, we identified that several of our presence clusters shared the same moniker naming: for example, the cluster 2309, 2310, 2311, 2313, 2314, 2315, 2316 in Figure 6 are all operated by SpectreDAO, representing 7 of the 14 nodes identified with the SpectreDAO moniker. However, we also found several cases where node names do not indicate shared operation, but presence patterns indicate they may be co-operated. For instance, nodes 32 and 258 share the same presence pattern as shown in Figure 6, and are likely to be run by the same operator although they have different moniker names. Furthermore, although nodes 1398, 2057 share one presence pattern and nodes 1528, 2071 share another presence pattern, 1398 and 1528 carry similar monikers while 2057 and 2071 show the same monikers, suggesting that all four nodes could be controlled by the same entity. While our results do not indicate a high level of node co-operation, we believe that continuous presence monitoring could be a useful countermeasure against future collusion attacks.

Table 3: Top five hosting providers for all nodes.

Hosting Provider	Percentage
PQ Hosting Plus S.R.L.	25.4%
Hetzner Online GmbH	12.8%
OVH SAS	6.7%
Akamai Connected Cloud	4.1%
Hostinger International Limited	4.0%

D.1.3 Shared hosting providers. Even when nodes are run by different operators, they may still use the same hosting providers. This can potentially introduce concentration risks when hosting providers attempt to de-anonymize users by monitoring ingress and egress traffic (for example, at the direction of a national security agency.) We retrieved the Autonomous System (AS) number from each node's IP address and identified the associated hosting provider. Considering the top hosting providers for all nodes on the network (Table 3) and top providers for all gateways (Table 4), we observed a very notable dominance by a single provider, PQ Hosting Plus S.R.L., which hosts 25.4% of all nodes and 44.2% of all gateway nodes. In part we suspect this is because PQ Hosting Plus has a reputation for being relatively friendly to anonymity node operators [20], and because it offers unlimited egress bandwidth and accepts cryptocurrency as payment. Unfortunately, this concentration means that many exit and entry gateways may be exposed to risk of monitoring by a single hosting provider, or that a single provider might damage availability by banning many gateway nodes at once.

D.2 Measuring the active set

Given the importance of the active set in routing packets, we investigated whether any patterns of centralization occur in this subset of the Nym network. To collect our data, we recorded the list of nodes in the active set every hour over a 30-day period to investigate how

[37] We also acknowledge that there is also the possibility of failures at the Explorer. Although we saw no evidence of this, we cannot conclusively rule it out.

[38] In cases where multiple nodes disappear simultaneously and do not reappear, this may not be sufficient evidence to conclusively assume that they are indeed operated by the same operator, considering that such behavior could plausibly result from unrelated nodes unbonding on the same day.

[39] While we aim to provide the most plausible explanation of different patterns, this interpretation is speculative.

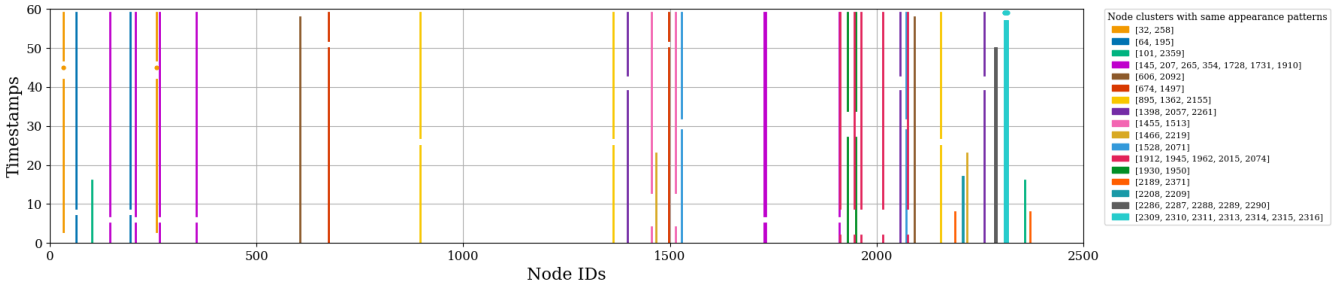

Figure 6: Nodes with disappearances over 30-day observation period that shared the same appearance patterns. A node's presence on the network is represented by a line across continuous timestamps or a dot at a specific timestamp. When a node disappears, no marker is shown. The gaps on the graph represent periods of inactivity. Lines of the same color indicate nodes that share the same presence patterns. Each node cluster has a unique color.

Table 4: Top five hosting providers for gateways.

Hosting Provider	Percentage
PQ Hosting Plus S.R.L.	44.2%
Hostinger International Limited	6.5%
DigitialOcean, LLC	3.9%
Fast Geo Hosting S.R.L.	3.9%
BrainStorm Network, Inc	3.0%

frequently individual nodes are selected, whether a same group of nodes is consistently selected, and to examine other common characteristics are among selected nodes.

D.2.1 Overall statistics. Figure 7 shows how many times each node was selected into the active set during our test period. Each column represents a single node, and bars are colored to indicate which role they were assigned to (entry gateway, layers 1-3, exit gateway.) Empty columns indicate nodes that were not selected at all.

The role of entry and exit gateways is determined by node configuration. Any mix nodes selected into the active set are supposed to be evenly distributed across three *layers*. To assess whether the assignment of mixnodes to layers is unbiased, we analyzed how frequently individual mixnodes appeared in each layer. Our findings indicate that the distribution appears uniform, indicating no significant bias toward any particular layer.

An active set always has 240 total nodes, but overall 453 out of 562 total nodes on the network, or approximately 80% of nodes, were selected into the active set at least once over the course of 30 days. Although no single node is present in every active set during our observation period, out of 435 nodes that are selected for at least once, approximately 33% of them are present in the active set for more than 70% of epochs, showing that there is still a strong tendency for a core group of nodes to be repeatedly selected.

D.2.2 Effect of stake. We mentioned in §2.1.3 that a node's probability of getting selected into the active set is determined by its performance score and stake, with performance score prioritized as it is raised to the 20^{th} power. When we examine the performance score of every node that has been selected into the active set at least once over the 30-day period, we find that, as expected, only nodes with high performance scores are selected (see Figures 8 and 9.)

However, when we examine the stake distribution for all 453 nodes that are selected into the active set for at least once over 30 days (see Figure 11) versus the stake distribution for all 562 nodes on Nym network (see Figure 10), it is clear that nodes with total stake exceeding 100,000 NYM dominate the active set, while those with lower stakes are rarely selected despite representing a moderate portion of the network. Therefore, in the current network, due to high stake saturation level, modestly-staked nodes with perfect performance scores will rarely be selected into the active set. While this should not be surprising, high stake-weight appears to be an implicit requirement to join the active set, indicating that packet routing power is highly concentrated.

Upon further investigation, we found that the majority of the highly-staked nodes are operated by the large node clusters identified in the previous section. Most of those clusters are run by decentralized autonomous organizations (DAOs) who have substantial funding and operate as a business service. Nearly all of their nodes are staked to the maximum amount, which is approximately $50,000 USD per node.

D.2.3 Correlations between Node ID and node role. In the course of our analysis, we also examined the *role* of each active set node (entry gateway, exit gateway, mix node) and compared these roles to the node ID, as shown in Figure 7. This plot shows a striking division in active set composition by node IDs, where nodes with lower IDs primarily take on mix node roles, and those with higher IDs predominantly run as gateways. This pattern was unexpected, since node role is not specifically correlated to node ID and can be updated at any time. After discussions with Nym scientists, we identified that this may be due to a historical reward pattern in which "mixmining" rewards were distributed only to mix nodes, rather than gateways until November 2024 [49]. Hence this graph simply illustrates the dramatic effect of changing reward strategies, and how this can drive many new entrants to the network.

E Calculating Network Fraction

E.0.1 Estimating the needed fractions f_{gw}, f_{mix}. We first evaluate how different f_{gw} and f_{mix} affect the average probability of having

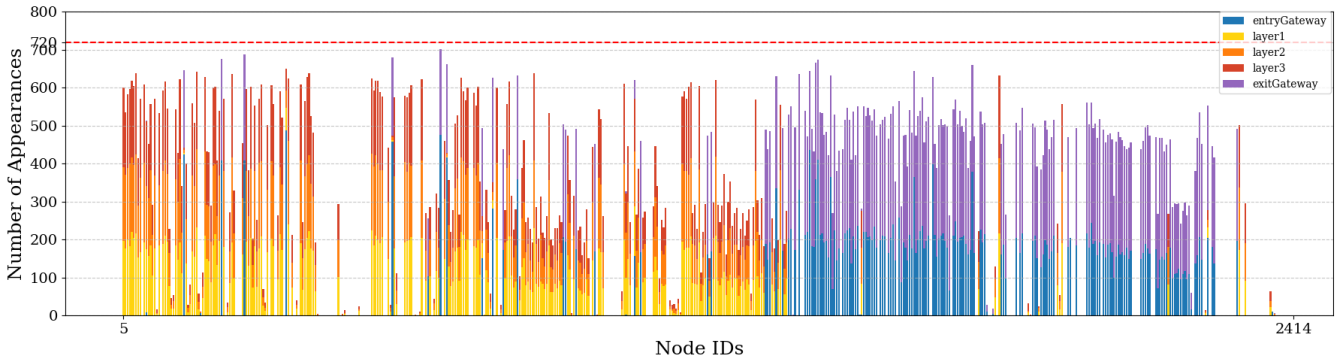

Figure 7: Active set nodes' appearance times. This shows the cumulative number of epochs any given node ID has appeared in the consecutive 720 epochs we observed; the number of times a given role/layer was observed is indicated by color.

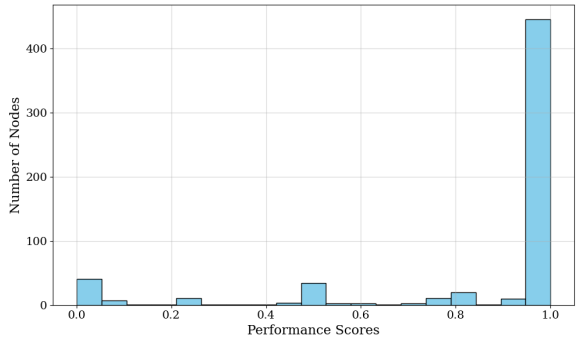

Figure 8: Distribution of all 562 bonded nodes by performance scores.

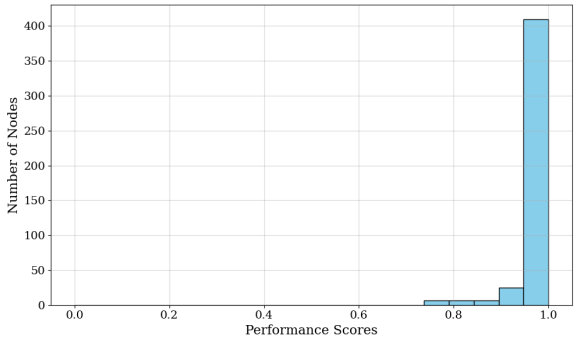

Figure 9: Distribution of all nodes that got selected into the active set for at least once over 30 days by performance scores.

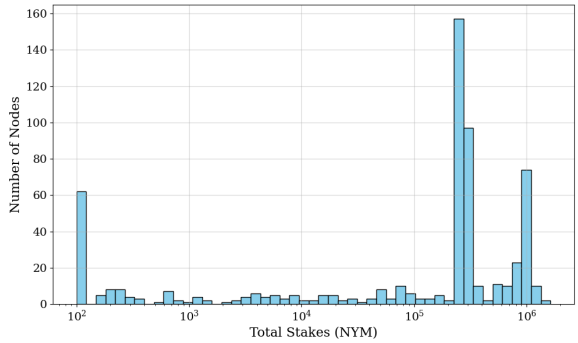

Figure 10: Distribution of all 562 bonded nodes by stakes per node.

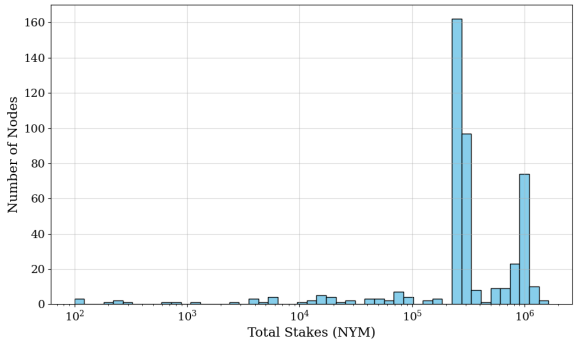

Figure 11: Distribution of all nodes that got selected into the active set for at least once over 30 days by stakes per node.

an $A^{***}A$ denoted by $Pr(A_{gw})$, or $^*AAA^*$ path denoted by $Pr(A_{mix})$ for one path that the client makes. The path probability here is calculated by the total product of the percentage of attacking nodes on each layer of the active set:

$$\Pr(A_\alpha) = \prod_{i \in L_\alpha} \frac{A_i}{V_i + S_i + A_i}, \quad \alpha \in \{\text{gw}, \text{mix}\},$$

where

$$L_\alpha = \begin{cases} \{0, 4\}, & \text{if } \alpha = \text{gw (A***A)} \\ \{1, 2, 3\}, & \text{if } \alpha = \text{mix (*AAA*)} \end{cases}$$

L represents the set of layer indices corresponding to each path scenario, and V_i, S_i, A_i, represent the number of victim nodes, sacrifice nodes, and attacking nodes on a layer in the respective active set of either gateways or mixnodes.

Since we run the simulated 24-epoch attack 100 times, we take the average of f_{gw}, f_{mix} and each associated $Pr(A)$ over 100 runs for each stake and node counts combination, and use those in our analysis below.

Considering that for either baseline or performance scoring attack method, if the fractions of the attacking nodes in the active set were the same, although the exact distribution of those attacking nodes to active set layers would differ somewhat in each simulation run, on average, the probability of having an attacker-controlled path would be similar. The result is thus independent of the attack method. We plotted the relationship between fractions of the active set and $Pr(A)$ for both methods, and the graphs confirmed that they indeed overlap each other. For simplicity, we used data from the baseline attack in Figure 12. Note that on the x-axis, each fraction represents the attacking nodes in its corresponding set of active set based on different attack objectives such that for **A***A** the fraction represents f_{gw} whereas for ***AAA*** the fraction represents f_{mix}.

E.0.2 Calculating the number of required client connections and packets. Given the fraction of attacking nodes in the active set and corresponding **A***A** or ***AAA*** path probability for every path, we calculate the minimum number of unique *connections* N_c or *packets* N_p that a client needs to generate for one of them to successfully route through one instance of an **A***A** or ***AAA*** path with success probability $Pr(S_\gamma), \gamma \in \{\text{A***A, *AAA*, AAAAA}\}$. Thus, we want to find N_c or N_p such that:

$$1 - (1 - \Pr(A_\alpha))^{N_\beta} \geq \Pr(S_\gamma),$$
$$\alpha \in \{\text{gw, mix}\},\ \beta \in \{\text{c, p}\},\ \gamma \in \{\text{A***A, *AAA*}\}.$$

Since connections or packets must be an integer, we take its ceiling value. After rearranging, we find N_c or N_p as the following:

$$N_\beta = \left\lceil \frac{\log(1 - Pr(S_\gamma))}{\log(1 - Pr(A_\alpha))} \right\rceil,$$
$$\alpha \in \{\text{gw, mix}\},\ \beta \in \{\text{c, p}\},\ \gamma \in \{\text{A***A, *AAA*}\}.$$

We show the minimum connections required for each f_{gw} in Figure 13 for path objective **A***A**, and the minimum packets required for each f_{mix} in Figure 14 for path objective ***AAA***. Although the **A***A** scenario requires significantly fewer connections compared to the packets for ***AAA***, the **A***A** scenario would take longer, considering that the gateways chosen by the client remain fixed to facilitate long-lived sessions, whereas three mixnodes change every packet. In practice, each instance of user reconnection presents an opportunity for a new path, and reconnection occurs very frequently, not only when a user intentionally starts a new session, but also due to connection issues in the client software, such as when the device resumes activity after going to sleep, or a previously selected server is no long available due to its low performance scores. The estimation of time for a set amount of packets is more straightforward, as it is simply the number of packets multiplied by the average packet sending rate of 0.02 seconds per packet.

F Verifying Expected Network Behavior

We did not execute our attacks against the Nym network, for obvious ethical reasons. To understand the experience of operating a Nym node, we did operate a bonded node on mainnet, following the extensive instructions offered by the Nym project [31]. We did not use this node to carry out disruptive experiments or to route real user traffic. Our goal in this was mainly to understand the process of operating a node, and to confirm several attack assumptions. Specifically: (1) the network monitor behaves as indicated in open source code (*i.e.,* new nodes will be incorporated into test packet paths with nodes currently in the active set), and (2) the distribution of active set predecessors/successors in mainnet is consistent with the results of our simulations. Additionally, through instrumenting the node source code, we verified (3) that previous and successor node IPs are available, and can be accessed within the code paths responsible for relaying packets. We further verified that (4) traffic received by our (non-active-set) node was consistent with our understanding that non-active-set nodes receive test packets only. And finally, (5) we confirmed the expected network score impact of failure to relay test packets (through the expedient of simply turning our non-active set node on and off.) The final experiment was the only active experiment that could impact other nodes on the network; however, nodes being disabled for various reasons (*e.g.,* for software updates or other unexplained causes) appears to be extremely common behavior on Nym's decentralized network. Because our node was not in the active set (and not relaying any network traffic) the impact on the operation of the real network is expected to be minimal and within normal operating bounds.

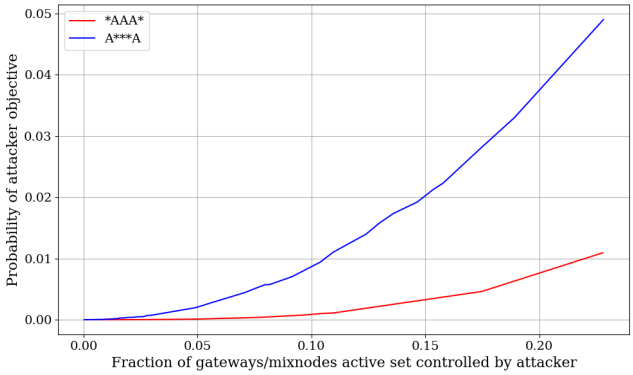

Figure 12: The probability that an attacker will achieve the objective of an A*A or *AAA* path, after exactly one path is selected by the client. This probability is evaluated for different attacker-controlled fractions of the active set, with the fraction being f_{gw} for A***A and f_{mix} for *AAA*.**

F.0.1 Sacrifice nodes appear on the same test packet paths as active set nodes. Our framing attacks rely on the assumption that sacrifice nodes will be placed on test paths with active set nodes. To verify this, we modified our Nym node software to log the previous and next node's IP addresses for every packet it received, and then ran the node as a mixnode for the entire epoch 23972 (this began at 2025-08-10 12:26+00:00 until 2025-08-10 13:26+00:00.) **Ethical note:** *We stress that since our node was not in the active set, this test did not encounter any real network traffic (beyond test packets), and the only data we collected involved the public IP address of active Nym nodes, which is not intended to be private information.* At the start of the epoch, we first collected the list of all active set node IDs

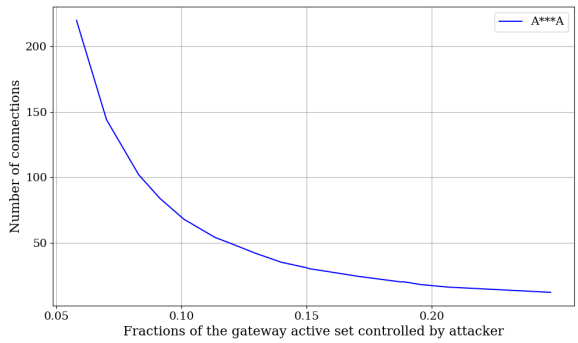

Figure 13: Number of distinct connections (gateway resamplings) that a client must generate to achieve a 50% probability of having one A*A path.**

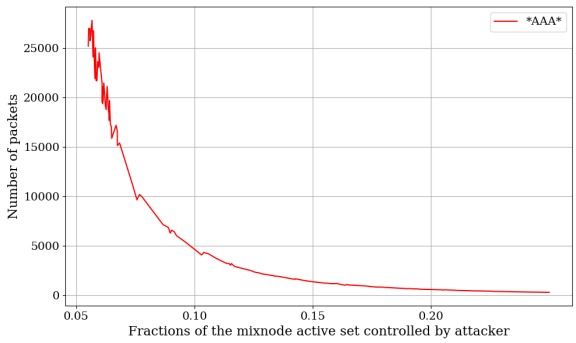

Figure 14: Number of distinct packets that a client must generate to achieve a 99% probability of having one *AAA* path.

from the Nym Network Explorer, and then recorded every node's ID and their IP address (including both active set nodes and non active set nodes.) By the end of the epoch, our node recorded a total of 10,882 total IPs (146 unique ones) including both previous and next hops. 2416 of those IPs (49 unique ones) belong to nodes in the active set in that epoch, indicating that our node had been on the same test packet path with 20% of the nodes in the active set. For the remainder of the 8,466 IPs (97 unique IPs), 7291 of them (57 unique) belong to non-active set nodes reachable from the Network Explorer. Interestingly, the final 1,275 IPs recorded during the epoch represent 40 unique IP addresses that are not related to any node on the Network Explorer during that time. We then repeated the same experiment during epoch 24,017 (from 2025-08-12 09:28+00:00 to 2025-08-12 10:28+00:00), and obtained similar results: our node recorded 54 unique IPs belonging to active set nodes (22.5% of the active set), and 44 IPs unknown to any node on the Network Explorer. Only two of those unknown IPs appeared in both experiments. We considered the possibility that these unknown IPs might be simulated clients managed by the network monitor. However, since our node is running as a mixnode and its mixnet layer is randomly assigned in each packet, it can be at most on the last layer of mixnet, where the next hop is always a gateway, indicating that our node will not be able to communicate with the destination client directly. Thus, the exact roles of these IPs remain unclear to us.

Next, we replicated the same experiment using our simulator to verify that its behaviors are consistent with the live Nym network. In this simulation, we created a topology that mirrors all the nodes in the Nym network including performance scores and stakes gathered from scraping the Network Explorer API, along with an additional node with minimum stake to ensure that it will not be selected into the active set. First, we selected an active set among all the nodes by replicating the real Nym selection process, then we ran the test packet path generation process in our simulator for a number of paths equivalent to the total paths expected in one epoch. For all test paths generated, we recorded the number of active set nodes our additional node encountered as either the previous or next hop. We repeated this simulation 100 times, and our results show that on average across all the simulation runs, the one node detected 50 distinct active set nodes across all test packet paths it was placed on, which is approximately 20% of the entire active set. Our results strongly indicate that our basic assumptions about the performance monitor strategy (drawn from code and documentation) are reflective of the system's real operation.

G Additional Simulation Results

Here, we show additional simulation results that further compare NMv1, NMv2, and countermeasure (which we denote by NMv3.) First, in Table 5, we compare the costs of baseline attack, framing attack, and attack with constraints for a fixed f_{gw} and f_{mix} goal considering NMv1, NMv2, and NMv3. Overall, the costs of framing attack in all three versions remain significantly lower than baseline attack. Next, in Figure 15, we show how the fraction of A nodes in the active set changes as a multi-epoch attack proceeds with NMv2.

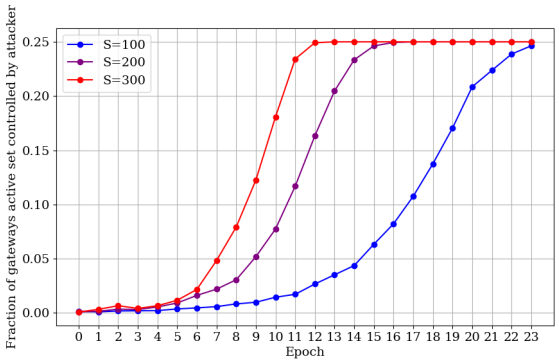

Figure 15: Simulation of the framing attack showing f_{gw} increasing over multiple hours. Attack results are averaged across 50 simulations. Here $|V| = 562$, $|A| = 30$ and each A node is staked at 1000 NYM, while $|S| \in \{100, 200, 300\}$. (Note that f_{gw} cannot exceed 0.25.)

Table 5: Two attack objectives, each with three example 24-epoch attack scenarios on NMv1, NMv2, and NMv3. N_c is the minimum number of connections that will be made by the target client (resulting in selection of fresh gateway pairs.) N_p refers to the minimum number of packets that the target client is expected to send through each connection (if applicable.) f_{gw} and f_{mix} indicate the target fractions of attacker-controlled gateways and mixnodes in their respective active sets. $Pr(S_\gamma)$ refers to the attacker's probability of success for an A*A or AAAAA path. $|S|$, $|A|$ represent the size of the attacker's node sets. Costs include the USD cost of both stake (refundable) and node hosting costs (non-refundable). Note that we omit showing refundable vs. non-refundable costs, and stake per node here for simplicity. We evaluate three attacks: our baseline "stake-only" attack; the performance scoring attack with unrestrained nodes amount that results in the lowest attack cost; and a variant that constrains the number of total nodes to the lowest amount that can still result in savings.**

Attack Parameters						Results					
Goal	N_c	f_{gw}	N_p	f_{mix}	$Pr(S_\gamma)$	**Method**	**Version**	**Costs ($)**	**Savings**	$\|S\|$	$\|A\|$
A***A	50	0.13	n/a	n/a	0.5	stake		254,400	n/a	0	600
						framing	NMv1	2,800	99%	80	20
							NMv2	2,520	99%	70	20
							NMv3	3,800	99%	90	20
						w/ constraints	NMv1	81,040	68%	20	20
							NMv2	81,040	68%	20	20
							NMv3	81,320	68%	30	20
AAAAA	50	0.13	1000	0.18	$\tilde{0.5}$	stake		1,484,000	n/a	0	3500
						framing	NMv1	4,200	99%	130	20
							NMv2	6,160	99%	170	50
							NMv3	165,720	89%	170	40
						w/ constraints	NMv1	282,960	81%	70	20
							NMv2	202,320	86%	40	50
							NMv3	204,560	86%	120	50