Empirical evaluation of traffic fingerprinting against the Nym mixnet

Simon Wicky
Nym Technologies

Ricardo Ferreira Ribeiro
EPFL/Nym Technologies

Ania M. Piotrowska
Nym Technologies

Carmela Troncoso
EPFL

ABSTRACT

Website fingerprinting (WF) enables a passive eavesdropper to infer which web page a client is browsing through encrypted or anonymized network connections. The attack has been widely studied in the context of Tor, demonstrating its damaging effect on users' anonymity. However, there has not been any analysis of fingerprinting attacks against *mix networks*, i.e., anonymous communication networks that aim to protect against even global network adversaries. In this work, we conduct the first empirical evaluation of WF against a mix network on the example of the Nym network. We show, that Nym offers stronger protection against fingerprinting attacks than onion-routing-based protocols. In the *closed-world* setting, we observe a 43% drop in attack accuracy when browsing through Nym mixnet compared to Tor. An even bigger improvement is achieved in the *open-world* setting where the $F1 - Score$ drops by 58%. We also perform a thorough feature importance analysis, to identify which traffic characteristics impact the effectiveness of the classifier. Next, we study the privacy vs overhead trade-off and compare how different configurations and parametrisations of Nym's cover traffic and mixing delay impact the susceptibility to the attack. Finally, we propose an improved cover traffic mechanism for the Nym client, which better thwarts fingerprinting attacks.

KEYWORDS

website fingerprinting, mixnets, anonymous communication

1 INTRODUCTION

Since 2004, Tor [17] has become the largest and most popular anonymous communication tool, attracting millions of users daily. Its popularity is due in part to its perceived strong anonymity properties. Tor is based on onion-routing [20], a protocol in which clients route their internet communication via multi-hop circuits, preventing any single relay on the routing path from linking the source and destination of the communication. Although Tor hides the routing information and communication content, research has shown that it offers limited security guarantees against traffic analysis techniques [27, 34, 36, 39, 45] because it does not alter the shape or patterns of network traffic. Thus, an adversary eavesdropping on both the Tor entry and exit relays can perform end-to-end correlation by analysing metadata features [48]. Notably, even when the adversary only has access to a client's local connection, it is possible to identify web pages visited by the user via Tor through *website fingerprinting* attacks (WFP), which exploit persistent and distinctive traffic patterns. Many previous works demonstrate the detrimental effects of such attacks on Tor [22, 25, 28, 29, 37–39].

An alternative design for anonymous communication networks are mixnets. A mixnet is a decentralised overlay network comprised of a set of independently-run servers [6], in which packets traverse via multi-hop routes. Similarly, as in the case of onion routing, all routed data is layer encrypted, once for each relay, and padded to a constant length. The long-lived bidirectional circuits used in Tor, which require the intermediate relay to maintain a per-connection state limiting the total number of concurrent anonymous connections that can take place simultaneously, are replaced in mixnets by independent paths randomly selected by the sender. This makes it significantly harder to trace end-to-end data flows, even for adversaries that monitor all connections in the network and allows the network to scale better. Critically, what differentiates mixnets from onion-routing is that in addition to transforming the routed packets cryptographically, each mix retains packets for a randomized amount of time before forwarding them to the next hop, altering their flow and preventing linking the incoming and outgoing packets. Thus, mixnets hinder traffic analysis and provide strong metadata protection against both local and global adversaries performing traffic analysis attacks.

Even though the concept of a mixnet predates Tor or other onion-routing-based systems [17, 20, 47], there has been no successful deployed implementation with users, mainly due to perceived higher latency that cannot accommodate real-time communications and significant computational requirements. However, there have been significant research efforts to tackle the scalability, privacy, and latency problems of the early mix network designs [7, 8, 42, 49]. Over the past few years, in response to the increasing awareness that Tor is not sufficient to hide metadata and provide traffic analysis resistance, there has been an increased interest in mix network technologies, with companies building novel mix-based overlay networks, with widely contrasting designs, and collectively attracting millions in investments [9, 23, 26, 31, 32, 35]. But while there have been several works studying overall anonymity provided by mixnets [10, 11, 15, 16, 44], there has not been much work measuring mixnet's resistance against fingerprinting. In this paper, we present the first empirical measurement of website fingerprinting against a mixnet. We perform our analysis on the Nym mixnet, which already offers a deployed mainnet with hundreds of nodes, a SOCKS5 proxy client, and a one-button interface to connect several

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–14

https://doi.org/XXXXXXX.XXXXXXX

applications to the mixnet. It also provides an open-source, well-documented code base with a test suite and SDK allowing for easy integration and analysis. Moreover, recent studies show that Nym offers the best anonymity vs performance trade-offs [41].

In this paper, we make the following contributions:

- We perform the first empirical evaluation of the website fingerprinting attack against the real-world Nym mixnet in both closed and open-world settings and compare it against the resistance offered by onion-routing protocols like Tor.
- In order to better understand which distinctive traffic characteristics are leaked by the Nym client, we perform a detailed feature analysis.
- We perform a thorough analysis of the impact the cover traffic and mixing delay have on the resistance against the attack. Further, we examine how those obfuscation methods impact overall performance and overhead.
- We show that the current cover traffic strategy used by the Nym client does little in obfuscating incoming communication, thus leaking patterns which the attacker can exploit to correlate website visits. To this end, we propose an improved cover traffic strategy for the Nym client, which increases resistance to WFP. Our technique introduces minimal overhead in comparison to the currently deployed one.

2 BACKGROUND AND RELATED WORK

2.1 Nym Network Overview

The Nym network is a decentralised overlay network designed to protect the privacy of online users. Developers can build on or connect existing applications and services to the Nym infrastructure for built-in privacy.

2.1.1 Network Architecture. The Nym mixnet consists of a set of *mix nodes*, which route encrypted data on behalf of Nym *clients*. In order to address the scalability limitations of the original Chaumian mixnet [6], mixes are grouped into layers and arranged into a *stratified topology* [18], i.e., each mix in layer i is connected with each mix in the previous and next layer, and traffic flows from the first to the last layer. As opposed to circuit-based Tor [17], in Nym mixnet packets are routed via independent communication paths, even between the same pair of users, i.e., for each packet, the Nym client chooses a fresh route, composed of one random mix from each layer, that a given packet will traverse before reaching its final destination. The mixnet is accessible via a set of gateways, which act as entry and exit proxies that forward packets from clients to the mixnet and keep the received replies for users when they are offline. Nym client establishes a long-lived secure channel with a gateway of its choice. Nym clients are not anonymous w.r.t their gateway, however, since the traffic is encrypted and routed via several mixes, the gateway cannot identify with whom the client communicates. As the user communicates, the Nym client sends all requests to the associated gateway, which then forwards them into the mixnet. The last layer mix node forwards the requests to the destination's gateway, which then redirects the request to the required client. Packets routed through the Nym mixnet are layer encrypted using the Sphinx packet format [12], which ensures bitwise unlinkability and integrity protection. Each packet is encoded in four layers, one for each mix node plus an additional layer for the exit gateway.

2.1.2 Metadata Obfuscation. To hinder traffic analysis, each mix delays packets independently before forwarding them to the next hop [30, 42]. The amount of time a packet dwells in a node is chosen by the sender, who samples it at random from an exponential distribution with publicly known parameter λ_M, and includes it in the vector of Sphinx routing commands to the corresponding relay.

To obfuscate the persistent patterns of users' activities and prevent active attacks [13], Nym clients continuously generate cover traffic, which is indistinguishable from real traffic. Each cover packet traverses an independent mixnet route before returning back to the initiating client, thus it is referred to as *loop cover traffic* [14, 42]. As a user sends a request, the Nym client does not immediately forward it into the mixnet, but instead, the Sphinx encoded packet is placed into the client's internal FIFO queue that stores all the messages scheduled to be sent. Following the exponential distribution with parameter λ_Q, the Nym client checks whether any packets are queued. If yes, a single packet is removed from the queue and sent. Otherwise, a *loop cover packet* is generated and sent in lieu. Thus, regardless of whether a user actually is communicating or not, there is always a stream of packets being sent according to a Poisson process $Pois(\frac{1}{\lambda_Q})$. Independently from the above, each Nym client emits also a separate stream of only *loop cover* traffic at rate $Pois(\frac{1}{\lambda_L})$. The use of loop cover packets allows the end users to obfuscate their activity patterns: when an adversary sees participants sending or receiving messages, it cannot distinguish whether they are actively communicating or just sending and receiving empty loops. Sending cover traffic in routes that loop back to the sender enables also maintaining local updated knowledge on the health of the network, including the ability to detect when nodes go down, are congested, or are under attack [13].

2.1.3 Anonymous Replies and reliable transport. To enable anonymous replies, Nym leverages the *Single-Use Reply Blocks* (SURBs) [12]. A SURB is a data frame containing a pre-computed Sphinx packet header encoding a mixnet route that ends in the participant who generated that SURB, a set of keys for payload encryption and the address of the first mix hop. A sender can generate one or more SURBs and include them in the Sphinx payload for the recipient, who can then use the SURBs by simply layer encrypting the response using the provided payload keys and combining the result with the supplied header. Since each SURB header is encrypted by the sender, the recipient sending it back cannot infer from it any information about the sender. Moreover, such single-use replies are indistinguishable from forward messages even for the mix nodes, thus they can share the same anonymity set.

Besides enabling anonymous replies, the Nym network uses SURBs to provide reliable transport even if there is packet loss in the network. To this end, the Nym client generates a small single-use Sphinx-encoded ACK packet and includes it in the forward Sphinx packet in the layer addressed to the recipient's gateway. The recipient's gateway extracts the ACK packet and sends it back. The sender re-transmits a message if no acknowledgement is received within the expected time frame.

2.2 Website Fingerprinting

2.2.1 Attack model. The *website fingerprinting* attack (WFP) [39] is a class of traffic analysis attacks performed by a passive *local* eavesdropper, who observes the encrypted traffic at a single capture point between the client and the anonymization network, and tries to draw conclusions about the user's online activities. To perform the attack, the adversary first collects unique traffic fingerprints of certain websites by performing multiple web requests through the studied network. Next, the adversary extracts various features and patterns, such as volume of transferred data, timing, packet sizes or direction, and trains a machine-learning algorithm. When a user visits a website, the attacker passively collects the user's network traffic and feeds it into the classifier to identify whether a client visited any of the target websites. Thus, the WFP attack allows an adversary to infer information about a user's online activities, without breaking cryptography.

The attack is evaluated in two different settings: *closed-world* and *open-world*. In a *closed-world* setting, the set of potential web pages a user can visit is limited and known in advance by the attacker. Under the more realistic *open-world* model, the adversary monitors a small set of known web pages (referred to as *foreground set*), but a client can also browse a large number of unmonitored web pages (referred to as *background set*). The goal of the attacker is to determine if the user is visiting any of the monitored web pages. An example of an open-world scenario is an adversary trying to infer whether a victim is visiting any of the censored websites.

2.2.2 Related work. In 2009, Hermann et al. [24] proposed the first website fingerprinting against multihop networks, including Tor and JAP, based on multinomial naïve-Bayes classifier. Their results indicated that Tor is secure against website fingerprinting with a detection rate of less than 3%. However, Panchenko et al. [39] proposed a WFP attack that relies on *Support Vector Machine* (SVM) classifier. By carefully selecting features from the raw traces they increased the precision in the dataset provided by Hermann et al. to almost 55%. Additionally, the authors analyzed WFP in the open world setting where they achieved a recognition rate of up to 73%.

Similarly, Cai et al. [5] also utilized an SVM but introduce new methods for computing the similarity of traces generated while loading a web page. Wang and Goldberg [51] further enhanced the accuracy of Cai et al.'s scheme on Tor by introducing a new distance-based metric (OSAD) for comparing packet traces, and improved the data preprocessing phase by removing Tor management packets.

Wang et al. [50] proposed using the k-Nearest Neighbour (k-NN) classifier to perform WFP. Their attack was the first to use a diverse set of features, including bursts, packet ordering, the concentration of the packets, the number of incoming and outgoing packets, transmission time etc.

CUMUL [37] achieves similar results as k-NN, while experimenting with large datasets, by extending the early idea of using the SVM [39] against features based on the cumulative sum of packet lengths.

Hayes et al. [22] devised a novel *k-fingerprinting* attack (k-FP), which uses a random forest classifier to extract a fingerprint for each webpage load, i.e., the leaves of the trees in the random forest represent the actual fingerprint. To calculate the distance between the different fingerprints the authors use the Hamming distance and classify a test instance using the k-NN classifier.

With the rise of deep learning a new line of research on using neural networks for WFP attacks with automated feature extraction was brought into play [1, 2, 43, 46], with VarCNN [2] technique achieving best results.

2.2.3 Countermeasures. To counter the WFP attacks against Tor, a broad range of defences has been put forward [5, 19, 19, 19, 28, 28, 33, 39, 53]. Most of the proposed defences are based on *link padding*, which aims to conceal network traffic patterns by adding varying amounts of dummy packets or delays to the packet flow. Panchenko et al. [39] suggested loading a background decoy page in parallel to a real page to add background noise that significantly decreases the accuracy of WFP as shown in [28]. *Traffic morphing* [53] selectively pads and splits packets to mimic the distribution of another website's packet sizes, however, at a cost of high bandwidth overhead [19]. In a similar manner, Tor pads all packets to fixed-size cells of 514 bytes. *HTTPOS* [33] uses several techniques at the application layer to alter the traffic features, including splitting a single request into multiple overlapping requests, generating fake requests, requests reordering or HTTP GET headers modification. At the TCP level, HTTPOS perturbs the size and order of packets. Another application layer defence is *randomized pipelining* [40], the only WFP countermeasure that has been implemented in Tor. It batches clients' requests in the HTTP pipeline and randomizes the size and the order of the pipeline queue. However, several studies have shown that these defences are not effective [5, 28].

Another group of WFP defences is based on *constant rate padding*. In *BuFLO* [19] fixed-length packets are sent at fixed intervals. However, this approach requires high bandwidth and time overhead and does not obfuscate the size of web pages that take longer to load than the pre-defined threshold. *CS-BuFLO* [3, 5] and *Tamaraw* [4] propose optimizations to the *BuFLO* design. Both CS-BuFlow and Tamaraw group web pages into different anonymity sets based on the total size of a webpage. In Tamaraw, the incoming and outgoing traffic is treated differently, using different packet intervals. In contrast to BuFLO, CS-BuFLO uses a scale-independent padding scheme and monitors the state of the page loading process to avoid some unnecessary overheads.

Another category are the *zero-delay lightweight* defences, that include *WTF-PAD* [29], *Walkie-Talkie* [52], *FRONT* [21] and *GLUE* [21]. *WTF-PAD* is based on *adaptive padding* originally proposed by Shmatikov and Wang [45], in which both endpoint clients generate dummy packets based on the outgoing traffic pattern to mask traffic bursts. However, in contrast to constant rate padding, it does not delay application data. WTF-PAD extends the original adaptive padding scheme by also sending padding messages as a response to messages received from the other end. In contrast to WTF-PAD, FRONT focuses mainly on obfuscating the feature-rich trace fronts as these leak the most useful features for WFP classification. In order to ensure that a webpage class does not leak any meaningful patterns, FRONT adds dummy packets in a highly random fashion, ensuring different traces of the same webpage look different to each other. As result, FRONT can outperform WTF-PAD in terms of attackers' performance, while introducing the same data overhead. *GLUE* in contrast explores the fact that most of the WFP attacks rely

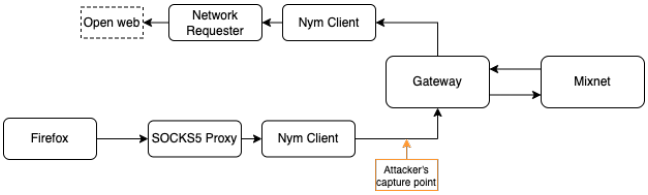

Figure 1: Attack setup

on the assumption that the adversary can easily separate singleton traces. Therefore, GLUE adds dummy packets between traces to make it seem as if the client is visiting pages consecutively without pause. When a client loads a new page, GLUE stops sending the dummy packets, thus hiding the start point of the next trace. Distinguishing singleton traces is now difficult thus the attacker is likely to fail.

3 DATA

3.1 Network configuration

To facilitate the data collection, we set up our own test Nym mixnet. The reason behind running a controlled environment, instead of using the open Nym mainnet[1], is reliability. At the time of our study, the Nym mainnet was just launching and thus many of the nodes were not yet set up correctly (i.e., some nodes were offline, some had their advertised ports closed, some supported only one of IPv6 or IPv4), making it challenging to collect captures. However, we used the same software that is run by the nodes in the Nym mainnet, and all nodes are run in a distributed manner. Since the website fingerprinting attack is performed by a local adversary, we argue that running the mixnet on a smaller scale does not bias the results. Thus, we leave measuring the attack on the Nym mainnet as future work. Since the website fingerprinting attack is performed by a local adversary, we argue that running the mixnet on a smaller scale does not bias the results. Thus, we leave measuring the attack on the Nym mainnet as future work.

Experimental Setup. On the architectural level, there are three components facilitating web browsing: a SOCKS5 proxy, a Nym client and a network requester. To visit a webpage, the browser sends the request to the SOCKS5 proxy, which then forwards it to the ingress Nym client which encodes it into the Sphinx packet and schedules it for sending. Once the encrypted packet carrying the request reaches the egress Nym client associated with the server, the client removes the encryption and forwards the TLS-encrypted query to the network requester.

We run the Nym mixnet using the Linode platform.[2] The mixnet consists of three layers with two mixnodes per layer, a gateway, a SOCKS5 proxy and a requester to relay the request from the proxy into the open web. All mix nodes run as separate Nanode 1GB instances. We run also a single Linode 8 GB instance on which we deploy a gateway, two SOCKS5 proxies, with the associated pair Nym client-requester. Every component of this mixnet runs version 0.9.2 with the exception of the requester which uses 0.10.0 to avoid a known bug while causing no compatibility issue.

Unless specified differently, we use the default configuration hardcoded in the Nym client, namely the inter packets delay λ_Q is currently configured for 20 milliseconds, while λ_L is configured for 200 milliseconds. The mixing delay is sampled from an exponential distribution with mean λ_M set to 50 milliseconds.[3]

In the current setup, a single Sphinx packet carries only one SURB with it, allowing only one reply per packet sent. In order to work around this current limitation the SOCKS5 proxy we use leaks the Nym address of the proxy (defined as `user@gateway`) to the requester, and the requester sends the replies directly to that address. From a system standpoint, this breaks the sender's anonymity. From the standpoint of our attacker, this has the effect that the requester does not have to wait for more SURBs to respond and there is no cap on the number of response packets the requester can send. This is a temporary solution, which will soon be replaced with a protocol for anonymous replies.[4]

3.2 Data Sets and Experimental Setup

To access webpages, we use a Python script that handles a Mozilla Firefox browser automated by Selenium[5] and Geckodriver[6]. The SOCKS5 proxy as well as the browser runs in a separate network namespace, i.e. a separate machine from a network standpoint in order to isolate the proxy traffic from the rest of the machine. The traffic between the SOCKS5 proxy client and the gateway is captured using dumpcap. This corresponds to a malicious ISP observing the traffic of a target user. Page loading was performed with no caches. We also disable Firefox's tracking protection, as the packets generated by it would make up almost the entire capture and behave very differently with different setups of the mixnet. Because of that, the captures were very inconsistent and not comparable. Moreover, we are trying to assess if Nym alone is efficient against WFP attacks, thus no need for additional countermeasures.

For comparison to previous work, we perform all of our evaluations using one of the previous largest datasets collected by Panchenko et al. [37], that consists of 1125 unique pages for the *foreground class* and 143212 unique pages for the *background class*.[7] This allows us to re-use the traces collected via Tor which were used in previous works. The foreground class contains 40 complete traces per page while the background class needs only one. Although the background dataset has 143212 URLs, only 111841 traces collected via the Tor network are included.

Since the Panchenko dataset was collected in 2016, some pages do not exist anymore and were therefore removed from the final dataset. In particular, we removed 266 such web pages, which amounts to approximately 23% of the total number. Moreover, some pages were too slow to load and hit the three minutes timeout that was set. A foreground page that failed to produce 40 correct traces out of 80 attempts is considered faulty and also removed from the final dataset. For the background class, this criterion was one correct

[1] https://explorer.nymtech.net/
[2] https://www.linode.com/
[3] https://github.com/nymtech/nym/blob/1733aaa4ccb24eb6f0d156c8a84d876bbca1919a/clients/client-core/src/config/mod.rs#L26
[4] https://github.com/nymtech/nym/tree/nym-binaries-v1.1.4
[5] https://www.selenium.dev
[6] https://github.com/mozilla/geckodriver
[7] https://www.informatik.tu-cottbus.de/~andriy/zwiebelfreunde/

trace out of three attempts. The final dataset hence contains 664 foreground pages with 40 traces each and 90428 background pages with one trace each. Similarly, as Panchenko et al. [37], we keep only the size, direction and timestamp of the packets traversing between the proxy and the Nym gateway, excluding the TCP ACKs. We do not remove any outliers, as the random forest classifier we use is robust to outliers.

4 EVALUATION OF WEBSITE FINGERPRINTING ON NYM

In this section, we discuss the methodology and metrics used to evaluate the effectiveness of website fingerprinting against Nym mixnet, present our findings and compare them with the impact of WFP on Tor.

4.1 Methodology

In our analysis, we use both *random forest classifier*, as well as *k-fingerprinting* methodology (combination of random forest and a k-NN classifier) introduced by Hayes et al. [22].

We first evaluate the attack in the closed-world setting, i.e., a multi-class classification problem, in which the classifier is trained on all the sites a user can visit. Given a webpage, we consider a result to be a *true positive* if a trace generated by a visit to that webpage is correctly assigned by the classifier to that webpage. A result is a *false positive* if instead a trace is generated by a visit to another webpage that is incorrectly classified as that webpage. Similarly, we consider a *false negative* if the trace of the webpage is incorrectly classified as coming from another page, and *true negative* if the classifier correctly assigned traces of other pages as not related to the webpage.

The used dataset consists of 664 web pages with 40 samples each. 75% of these samples were used as a training set, while the remaining 25% constitute the testing set. The training set is fed to a Random Forest Classifier with 200 trees. We used the Gini impurity to evaluate feature importance and the quality of each split. The classifier was then tested on the testing set. As noted in [22], the closed-world setting does not need the additional fingerprint layer for classification, we can simply use the classification output of the random forest since all classes are balanced.

Next, we evaluate the attack in the open-world setting. In this setting, we consider only *monitored* and *unmonitored* webpages. Thus, we have a true positive if a trace of a monitored webpage is correctly classified as monitored, and a false positive if a trace of an unmonitored webpage is incorrectly classified as monitored. Similarly, a true negative is a trace of an unmonitored webpage correctly classified as unmonitored, while a false negative is a trace of a monitored webpage classified as unmonitored.

The foreground class (monitored) consists of the closed world dataset, whereas the background class (unmonitored) is made from 90428 web pages with a single sample each. As before, 75% of these samples are fed to a Random Forest Classifier during the training phase. In this setup, the classifier has 500 trees. For each sample, each tree in the forest associates a leaf identifier to it. We call the vector formed by all the identifiers of a sample its *fingerprint*, which is used in a *k-NN* layer to finally classify the samples. A test sample is classified as the label of the k closest training samples in terms of Hamming distance if all these labels are identical. If it is not the case, the sample is classified as a background sample.

Feature Idx	Description	Importance
143	Average incoming packet size	0.01595044
142	Average packet size	0.01448705
148	Std of packet sizes	0.01423634
145	Variance of packet sizes	0.01421266
43	Fraction of incoming packets	0.01400223
44	Fraction of outgoing packets	0.01371360
140	Sum of sizes of incoming packets	0.01120885
12	25p of incoming packet timing	0.00950561
150	Std of outgoing packet sizes	0.00931428
144	Average outgoing packet size	0.00926206
36	Std of indices of incoming packets	0.00921170
147	Variance of outgoing packet sizes	0.00915977
13	50p of incoming packet timing	0.00906833
14	75p of incoming packet timing	0.00905640
34	Average of indices of incoming	0.00879392
22	75p of packet timing	0.00875545
24	Number of incoming packets	0.00871674
19	100p of outgoing packet timing	0.00853928
23	100p of packet timing	0.00851032
15	100p of incoming packet timing	0.00842832

Table 1: Top 20 features for Nym in closed world setting

4.2 Feature importance analysis

To better understand which features matter in the fingerprint classification we perform a thorough feature importance analysis. For that, we use the set of network-level features proposed by Hayes et al. [22]. Their features extracted from the TCP streams include variations in timing, size and volume features. Similarly, as in [22] we use the Random Forest classifier to analyse how significant each feature is in the classification process.

Both Nym and Tor pad all packets to fixed-size cells, 2413 bytes and 514 bytes respectively. Each of the packets is then additionally wrapped in the underlying lower-level protocols, e.g., TCP, TLS, IP or WebSocket, which perform packet fragmentation and clumping in order to improve efficiency. Those clustering and multiplexing mechanisms depend on the overall volume and timing of the communication. Thus, if the underlying persistent patterns are not unified, different web pages will result in different sizes on the wire making the fixed packet sizes obsolete. Therefore, we include in our analysis the effect of the size features.

Closed world: In the case of Tor, features related to size, number of packets, their concentration and direction dominate the ranking and are sufficient for the attacker to infer which web pages the user is visiting (Figure 2b, Table 2). This is expected as different web pages have varying resource sizes, thus the number of the exchanged packets and their direction or concentration can be easily observed. Furthermore, the lack of timing obfuscation in Tor is another key criterion for the classifier.

We note a visible difference in the feature analysis for Nym, as the majority of features which exhibit high importance for Tor are now significantly less informative (Figure 2a, Table 1). Indeed, *sum of sizes of outgoing packets* which was a key feature for Tor, places as 40th in the case of Nym with an importance decrease of 71%. Similarly, *sum of sizes of all packets* drops from the 3rd to 30th position. In general, we observed that all features related to the *outgoing* traffic dropped significantly in the ranking. This is due to the fact that Nym clients, in contrast to Tor clients, inject packets into the

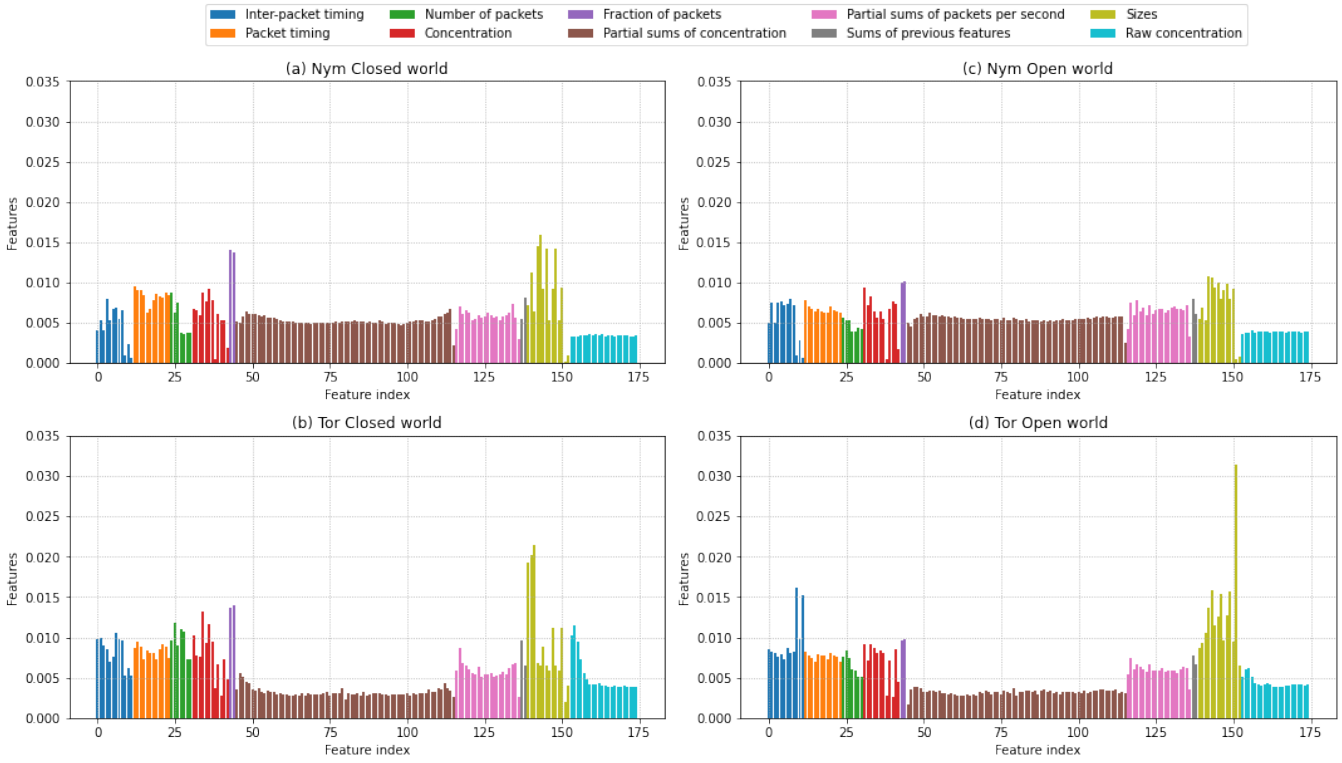

Figure 2: Feature analysis

Feature Idx	Description	Importance
141	Sum of sizes of outgoing packets	0.02142797
140	Sum of sizes of incoming packets	0.02022826
139	Sum of sizes of all packets	0.01926666
44	Fraction of outgoing packets	0.01391487
43	Fraction of incoming packets	0.01359591
34	Average of indices of incoming packets	0.01318452
25	Number of outgoing packets	0.01180980
36	Std of indices of incoming packets	0.01156945
154	Array of concentrations	0.01154150
150	Std of outgoing packet sizes	0.01121878
147	Variance of outgoing packet sizes	0.01113287
27	Number of incoming packets in the first 30 packets	0.01095341
28	Number of outgoing packets in the first 30 packets	0.01073876
6	Std of incoming inter-packet timing	0.01053112
153	Array of concentrations	0.01023003
31	Std of num. of outgoing packets per group of 20 packets	0.01022495
1	Maximum outgoing inter-packet timing	0.00989105
7	Std of outgoing inter-packet timing	0.00983684
0	Maximum incoming inter-packet timing	0.00975704
137	Sum of features 0 to 11	0.00959513

Table 2: Top 20 features for Tor in closed world setting

Feature Idx	Description	Importance
142	Average packet size	0.01071977
143	Average incoming packet size	0.01061818
44	Fraction of outgoing packets	0.01009181
145	Variance of packet sizes	0.01003532
43	Fraction of incoming packets	0.00997685
148	Std of packet sizes	0.00987713
31	Std of num. of outgoing packets per group of 20 packets	0.00941208
144	Average outgoing packet size	0.00936155
150	Std of outgoing packet sizes	0.00928927
147	Variance of outgoing packet sizes	0.00903957
33	Std of packets per second	0.00829567
7	Std of outgoing inter-packet timing	0.00799167
137	Sum of features 0 to 11	0.00798014
146	Variance of incoming packet sizes	0.00794594
149	Std of incoming packet sizes	0.00791801
119	xSums of packets per second split in 20, plus rest	0.00780392
12	25p of incoming packet timing	0.00778222
40	Minimum of packets per second	0.00769579
4	Average outgoing inter-packet timing	0.00758407
117	xSums of packets per second split in 20, plus rest	0.00754518

Table 3: Top 20 features for Nym in open world setting

network following the Poisson process. Thus, there is no distinctive pattern in the outgoing communication regardless of the visited webpage and therefore those features rank low. However, we find it interesting that the sizes and fractions of the incoming packets are so informative (see Table 1). This might be surprising, given that Nym uses cover traffic to alter the volume of communication. We deep dive into the obfuscation mechanism and find that while it disperses patterns in the outgoing traffic, it does little for the incoming communication (we discuss it further in Section 6). Because the statistics regarding all packets in the capture depend on the incoming traffic, which constitutes the majority of the trace as the response is usually significantly bigger than the request, those features also ranked high.

Open-world: Similarly, as before, the most important features of Tor communication are related to the size of packets on the wire. In particular, the statistics related to the size of incoming packets rank among the top 5 features, with the *maximum size of incoming*

Feature Idx	Description	Importance
151	Maximum size of incoming packet	0.03149505
9	75p incoming inter-packet timing	0.01611047
143	Average incoming packet size	0.01592640
149	Std of incoming packet sizes	0.01573075
146	Variance of incoming packet sizes	0.01544565
11	75p inter-packet timing	0.01520152
142	Average packet size	0.01373980
148	Std of packet sizes	0.01275595
145	Variance of packet sizes	0.01252173
144	Average outgoing packet size	0.01153761
141	Sum of sizes of outgoing packets	0.01058474
44	Fraction of outgoing packets	0.00978896
10	75p outgoing inter-packet timing	0.00974414
43	Fraction of incoming packets	0.00967728
147	Variance of outgoing packet sizes	0.00959002
150	Std of outgoing packet sizes	0.00952633
140	Sum of sizes of incoming packets	0.00930174
31	Std of num. of outgoing packets per group of 20 packets	0.00923107
33	Std of packets per second	0.00909151
139	Sum of sizes of all packets	0.00874112

Table 4: Top 20 features for Tor in open world setting

packets surpassing by a large margin the remaining features (2d and Table 10).

In contrast, the analysis of Nym captures did not identify any outstanding features, and the overall importance is distributed among various statistics, mostly related to sizes, fractions and concentration of the packets. This suggests that the classifier was only able to identify a few web pages that were particularly different in some features, but there is no common feature which imposes a significant information leakage.

4.3 Evaluation metrics

We evaluate the effectiveness of the WFP attack against the Nym mixnet using the metrics applied in previous works:

- *Accuracy* metric measures how many overall cases were correctly identified by the classifier.
- *Precision*, also denoted as Positive Predictive Value, measures how many positives were correctly identified from all the predicted positive cases. Thus, it is the ratio of true positives to the total number of traces that were classified as positive (true positives and false positives). Hence, it allows us to reason how accurate the model is.
- *Recall* defines the true positive rate, i.e., the ratio of true positives to the total number of relevant elements (true positives and false negatives). Thus, *recall* calculates how many of the actual positives the model captures.
- *F-1 score* is a harmonic mean between Precision and Recall, thus measuring the trade-off between these two.

In contrast to *Accuracy*, *F-1 score* takes into account how the data is distributed, hence it is a better metric if we consider classes that are unbalanced. Thus, *Accuracy* is a better metric for the closed world setting, while *F-1 score* is more suitable for the open world. However, previous works [22, 29] opt for only the *F-1 score* in both closed and open worlds. Therefore, for a detailed comparison, we measure all of the above-mentioned metrics.

4.4 Attack results

In Table 5, we present the results of the attack. As mentioned before, in the closed-world setting we use the classification output of the random forest, while in the open-world, we additionally add a $k-NN$ layer. We find that $k = 1$ gives the best results both in the case of Nym and Tor. Due to our methodology, increasing k can only move samples from the positive class to the negative class. For this reason, the *recall* can only go down as k increases. In our case, the *precision* gained by increasing k is not enough to balance that drop, since our number of false positive samples is already really low.

Setting		Accuracy	Precision	Recall	F1 Score
Closed world	Nym	**0.3329**	0.3305	0.3329	0.3126
	Tor	**0.5872**	0.5799	0.5872	0.5707
Open world	Nym	0.7913	0.6760	0.1555	**0.2529**
	Tor	0.8541	0.7978	0.4786	**0.5983**

Table 5: Comparison of website fingerprinting attack

As expected, Nym performs significantly better against fingerprinting both in closed-world and open-world settings. Our analysis showed that while the accuracy for Tor was 0.59, the attacker obtained only 0.33 accuracy while fingerprinting traffic routed via Nym. In the open world, the difference in attack success is even more significant, i.e., in the case of Nym the F1-Score drops by 58% in contrast to Tor. As mentioned before, the metric which should be taken into account in the case of the open world is the F1 score. As the foreground and background sets are not balanced, accuracy is not an optimal metric. Indeed, the background set constitutes around 77% of the overall data. Thus, by predicting the background each time we test a capture we will get a 77% accuracy, which is close to the obtained result of 79%, thus the accuracy metric does not provide a strong indication of the actual attack success rate. The results are in line with the insights provided by the feature analysis (Figure 2c, 2d and Table 9, Table 10), showing that in the case of Tor features like inter-packet timing or size leak significant information to the attacker, making the classification much easier than in the case of Nym.

5 PRIVACY VS OVERHEAD

In order to provide strong anonymity within the network, and to obfuscate the timing and volume of users' active communication Nym uses cover traffic and packet mixing. In this section, we examine the impact of cover traffic and mixing delay on the resistance of the Nym mixnet against website fingerprinting. First, we evaluate how significant each stream of cover traffic is in defeating the attack. Next, we analyse how different parameterizations of the Nym client impact its resistance against the website fingerprinting. Moreover, we also measure the impact of the different scenarios on the bandwidth and time overhead.

5.1 Impact of individual cover traffic streams

We evaluate the attack's performance given different configurations of the Nym client To this end, we define the following set of configurations for the Nym client. *Default* configuration is the default Nym client configuration, in which both Pois_{λ_Q} and Pois_{λ_L} operate as described in Section 2. In the $covered_{\lambda_Q}$ setting, Pois_{λ_Q} stream operates without changes, while the Pois_{λ_L} stream is deactivated. Conversely, in $covered_{\lambda_L}$ setting Pois_{λ_L} operates without

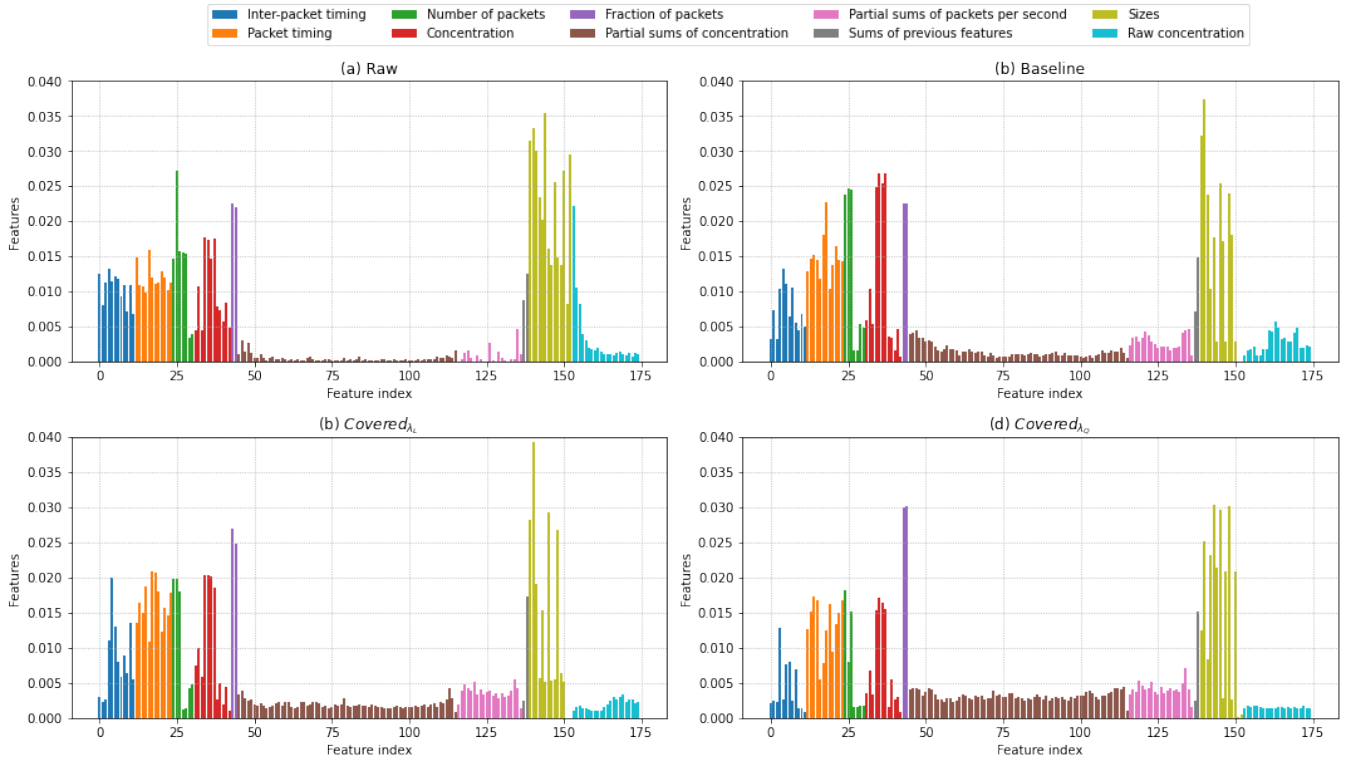

Figure 3: Feature analysis given different Nym client configurations

changes, but we deactivate the cover traffic in Pois$_{\lambda_Q}$, meaning Pois$_{\lambda_Q}$ sends only real packets if available or nothing otherwise. In the *baseline* configuration only real payload traffic is being sent. In other words, we turn off the stream Pois$_{\lambda_L}$ and the substitute cover traffic in Pois$_{\lambda_Q}$, while the payload packets are sent with a random exponential delay. Finally, we also analyse an extended baseline configuration with no mixing delays (i.e., no mixing and no cover traffic). For comparison, we also collect *raw* captures without any anonymous network.

Given our limited hardware resources, we could not afford to run a full-scale analysis for all settings. Therefore, to evaluate the effectiveness of the different parameterizations and configurations against WFP, we decided to perform the attack on a subset of the closed-world setup containing 100 web pages with 40 traces each. As before, 75% of these samples were used as the training set, while the remaining 25% constitute the testing set. Although we use a smaller subset, we can already observe significant patterns and trends in our results, which can be extrapolated to a full-scale analysis.

In Table 6, we present the results of the attack given different configurations. Figure 3 depicts the impact of different configurations on the importance of feature groups. Since the attack is performed on a much smaller and less diverse subset, it is reflected in the overall results, in contrast to the full-scale analysis presented in the previous section. However, those results are only intended to provide insights and highlight trends.

Configuration	Accuracy	Precision	Recall	F1 Score
Raw	0.923	0.927	0.923	0.922
Baseline	0.885	0.893	0.885	0.882
Covered$_{\lambda_L}$	0.793	0.799	0.793	0.791
Covered$_{\lambda_Q}$	0.702	0.706	0.702	0.691
Default	0.696	0.697	0.696	0.686

Table 6: Effectiveness of website fingerprinting given different cover traffic configurations

As expected, in the raw setting the attack is highly successful. Indeed, in the absence of any disguise techniques simple counting of the network packets can be a sufficient indicator of which webpage is being visited. This is reflected as well in the size, as the most informative feature was the average outgoing packet size, confirming that sending the request to the web server already leaks significant information [21].

We observe a slight improvement in the baseline setup in contrast to the raw captures. Although in the baseline setting, there is no cover traffic, the random exponential delay between sending packets increases entropy in the outgoing communication. As result, the sum of the sizes of incoming packets becomes the most informative feature. However, overall the sole delaying of packets before sending does little shielding against the fingerprinting.

As the results show, the majority of protection in the *default* setting comes from the cover traffic produced by $Pois(\lambda_Q)$. This

Configuration	Bandwidth Overhead	Time Overhead
Baseline	1.88	11.81
Covered$_{\lambda_L}$	2.46	12.25
Covered$_{\lambda_Q}$	6.85	12.29
Default	7.36	12.54

Table 7: Size and time overhead

is expected, as the second Poisson process, $Pois(\lambda_L)$, produces a loop cover packet on average every 200 ms. Thus, such loops offer little protection when real packets are sent on average every 20 ms. This is also highlighted in the list of top 20 features (see **??** in Appendix A) which shows that such sparse cover traffic is not enough to obfuscate the communication patterns of neither client nor the responding server. Indeed, we observe that although the importance of the features related to the number of packets and their concentration is demoted, the ordering of the packets and their timing increase significantly in relevance. Thus, $Pois(\lambda_L)$ stream should be considered more as an additional obfuscation that a user might want to use to detect active attacks [13] or check the health of the network, however, it could be also disabled to limit the bandwidth overhead, as we show next.

5.2 Bandwidth overhead

Using the collected data, we also analyse the impact of different cover traffic configurations on the *bandwidth* and *time overhead*. To this end, we compare the capture sizes observed when accessing the webpages via the Nym network in various configurations against the actual size of the webpages.

To calculate the overhead, we define it in the following manner: Let n be the number of instances of capture and N the number of web pages. Let s_{ij} be the size of the $j-th$ instance of the $i-th$ webpage and $S_i = \sum_{j=1}^{n} s_{ij}$ the sum of all captured sizes for the $i-th$ webpage using mixnet. Similarly, let SR_i be the sum of all captured sizes for the $i-th$ webpage without using the mixnet. We define the size overhead of the $i-th$ webpage as $OS_i = \frac{S_i}{SR_i}$. We compute the time overhead in the same manner. In our captures, we observe that the collected captures have significant outliers in terms of the loading time, and thus in the size. Therefore, to measure the overall overhead we calculate the *median*, to prevent the outliers from distorting the result. The results are summarized in Table 7.

As the *baseline* configuration does not use any cover traffic, the size overhead is minimal, as it is only impacted by the use of the Sphinx encoding, and acks and confirmation cells sent between the Nym client and the gateway. For comparison, given the captures via Tor for the same set of webpages, we get 1.44 size overhead and 10.22 time overhead. Similarly, the size overhead in the *Covered*$_{\lambda_L}$ configuration is low since the sparse cover traffic constitutes a small fraction of the overall communication. Significantly higher size overhead is observed in *Covered*$_{\lambda_Q}$ configuration, as the client sends continuous cover traffic (at the default rate of 50 packets per second on average) while waiting for the response from the web server. Finally, the *default* configuration has the highest size overhead as it produces the largest number of cover traffic.

The time overhead is mostly influenced by the mixing delay and the sending rate λ_Q of the Nym client, thus different settings

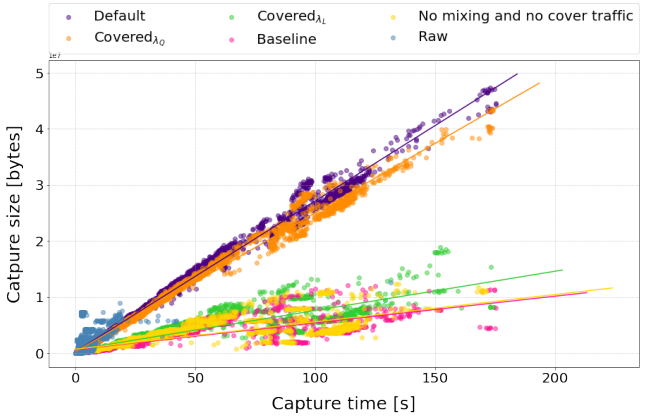

Figure 4: Size vs Time of each capture given different configurations of the mixnet. Each data point refers to a single instance.

of the client do not impact it significantly. It is also only slightly higher than the time overhead observed when browsing via Tor. However, as expected, the size overhead heavily depends on the client configuration.

Our analysis indicates that the capture size is correlated to its duration (see Figure 4). This is due to the fact, that the longer the loading of the webpage takes, the more cover traffic is emitted by the client. This has a huge impact on the data overhead, as well as the overall fingerprintability (i.e., even if we discarded the size features from our analysis, the model can use the timing information to classify webpages based on the overall loading duration). Therefore, in the next section, we investigate how different parametrizations of the cover traffic and mixing delay impact the resistance against the attack and bandwidth overhead. As the Pois$_{\lambda_L}$ stream has little impact on hiding the communication patterns, we focus our analysis only on the Pois$_{\lambda_Q}$ stream, while λ_L remains unchanged at set to the default 200 ms.

5.3 Impact of different parametrization

In this section, we investigate the impact of varying parametrizations on the fingerprinting resistance and the size and time overhead. As before, we perform our evaluation on a set of 100 web pages with 40 traces each. 75% of the collected samples were used for training, while the remaining 25% for testing. Let us define as λ_{Q_c} the parameter λ_Q of the requesting client and as λ_{Q_s} the parameter λ_Q of the responding client (i.e., the one operating on the side of the web server). As introduced before, λ_D denotes the average mixing delay.

Table 8 draws a good picture of what impact those three parameters have. A fast client increases the bandwidth overhead but does not impact the time overhead. On the other hand, a fast server decreases both the time and size overhead. The reason for these results is that while a faster client sends more cover traffic, thus it increases the bandwidth overhead, the real traffic it sends is too small to impact the capture duration. The server's client, on the other hand, sends most of the packets of the communication. Therefore, the speed at which the server sends its packets dictates the

Parametrization λ_{Q_C} / λ_{Q_S} / λ_D	Accuracy	F1 Score	Precision	Recall	Size Overhead	Time Overhead
20 / 20 / 50	0.696	0.686	0.697	0.696	7.36	12.54
10 / 20 / 50	0.594	0.574	0.598	0.594	12.91	12.08
40 / 20 / 50	0.659	0.645	0.658	0.659	4.47	12.98
20 / 10 / 50	0.688	0.674	0.689	0.688	6.01	8.77
20 / 40 / 50	0.574	0.559	0.579	0.574	12.14	21.57
20 / 20 / 20	0.609	0.593	0.620	0.609	6.98	11.04
20 / 20 / 100	0.582	0.566	0.574	0.582	8.76	14.93

Table 8: Website fingerprinting given different parametrisations

communication duration. Since longer captures are also bigger, due to the increase of the cover traffic, this rate also impacts the size overhead. For the same reason, short mixing delays decrease the time and size overhead.

The results also show that different parametrizations not only impact the overhead, but also the WFP protection of the mixnet. A faster client protects better against WFP attacks as it sends more cover traffic. A slower server also decreases the accuracy of a WFP attack. The analysis of feature importance shows that this is primarily because a slower server decreases the importance of the incoming packet timings and concentration in the classification. This tells us that the real packets are better hidden within the cover traffic with a slower server. Finally, this confirms that mixing delays have only a small impact on WFP protection. It is worth noticing that we could not test significantly higher mixing delays because of the HTTP timeouts. These types of high delays might have a noticeable impact on WFP resistance. However, they would also make the mixnet unusable for web browsing.

The results outline the trade-off between overhead and attack resistance and suggest that Nym could achieve faster communication with the same fingerprinting resistance and size overhead by using a fast server and a fast client. It would be interesting to evaluate the trade-offs on the actual Nym mainnet, thus we leave it as a future work.

6 IMPROVED COVER TRAFFIC TECHNIQUE

In this section, we outline proposals for new cover traffic strategies which can be implemented by the Nym client. As we observed in our earlier analysis (Section 4) the cover traffic generated by the Nym client does not adapt to the volume of incoming traffic. Each loop cover packet translates to two outgoing packets (a loop dummy packet sent and an acknowledgement being sent) and two incoming packets (a loop dummy packet coming back and an acknowledgement coming back). Thus, when a user is idle, the Nym client keeps generating loop cover packets, and thus the outgoing and incoming traffic is perfectly symmetrical. Once the user becomes active and sends a *request* packet to the web server the number of outgoing packets does not change, i.e., the client will just replace the loop packets with real packets. However, the volume of incoming packets changes as the web server has multiple packets to send back (and as pointed in Section 2 due to the SOCKS5 configuration there is no cap on the number of packets the web server can return). Such incoming packets carrying the response create a *burst* which can be easily identified. As different web pages have different resources, the size of the burst changes leaking information to the classifier

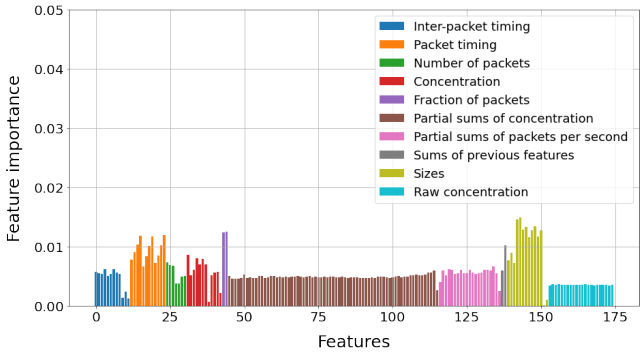

Figure 5: Feature importance given new cover traffic strategy

allowing identifying users' activities. Moreover, such bursts will occur not only in the case of web browsing but also e.g., file-sharing. Therefore, we propose an alternative technique that aims to hide the bursts of communication.

6.1 Dynamic use of drop cover traffic

The lack of burst obfuscation of incoming packets has already been studied by Juarez et al. [29] in their work for WTF-PAD. However, their solution requires a trusted third-party element (i.e., Tor bridge). Moreover, computing and keeping the histogram of inter-arrival times up-to-date is a tedious task that requires significant logistics. Nonetheless, we can adapt this idea to Nym by making two key observations.

Nym does not need a third-party element to control the amount of incoming traffic. Indeed, we did not exploit the drop cover packets from Loopix [42] so far. Drop cover packets are similar to loop cover packets in the sense that they are dummy packets that are indistinguishable from real packets. However, instead of looping back to the client, they are dropped by the last mixnode in the path. The issue with loop cover packets is that there is one incoming packet for each packet the client injects into the mixnet. The loop packets sent while the client is receiving reply packets do not obfuscate the incoming traffic and might even accentuate the asymmetry by increasing the number of received packets. A key observation is that we can leverage the drop packets to decrease the number of received packets during bursts and better obfuscate them. However, we cannot simply re-introduce the stream of drop cover traffic as proposed in Loopix [42]. The new cover strategy must dynamically react to bursts to be efficient. Switching the stream of drop cover on and off would change the sending distribution of the client, which is detectable, and increase significantly the data overhead.

Thanks to the feature importance analysis, we noticed that the two most significant leaks are the fraction of incoming and outgoing packets. Another key observation is that we can efficiently attenuate the importance of these features by mapping each incoming packet to an outgoing packet with the help of drop-cover packets. At the same time, this will also impact the feature "Number of incoming packets", which we identified as the third most important feature.

Instead of the naive solution, we propose the following. When a client detects a burst of incoming packets, the loop packets from the stream of real traffic are replaced with drop cover packets. That way,

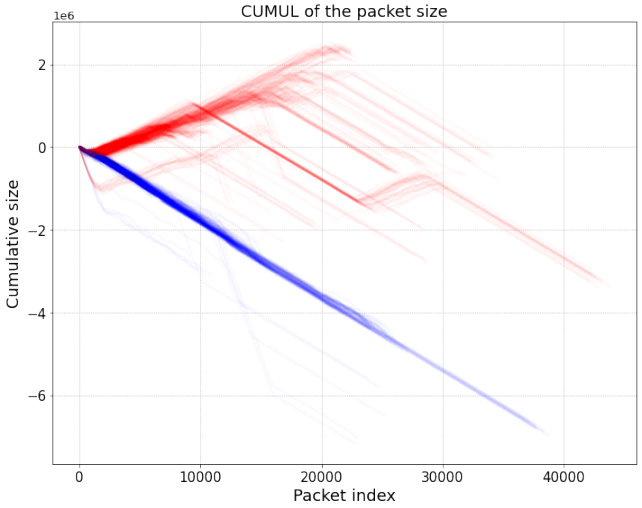

Figure 6: CUMUL representation of traces generated by a client accessing default cover traffic strategy (red) and proposed strategy leveraging drop cover packets (blue).

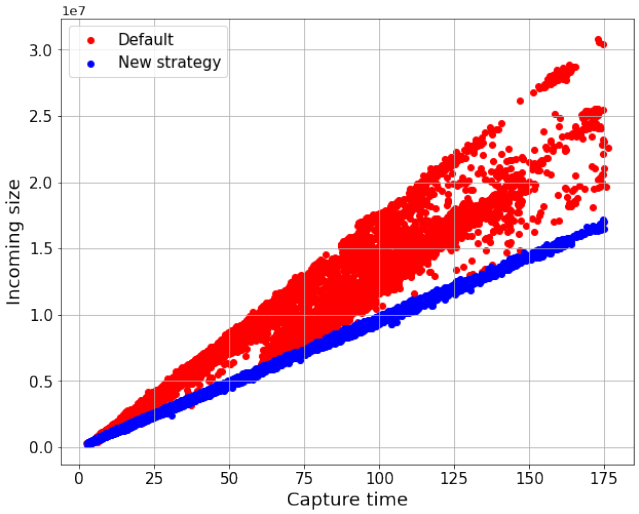

Figure 7: Size of incoming communication vs time for the new cover traffic strategy

each real incoming packet can be mapped to an outgoing packet. This mechanism should provide an enhanced obfuscation of the bursts.

Bursts detection: To detect the bursts, we add a counter at the Nym client, which tracks the difference between the number of sent and received packets. The counter is negative when the client sends fewer packets than it receives. When the counter indicates that the client received more packets than it sent, i.e., a burst might be happening, the loop packets in stream Pois_{λ_Q} are replaced with drop cover packets. These drop packets are sent to a randomly selected node in the last layer of the mixnet where it is dropped. That way, each incoming packet can always be one-to-one mapped to an outgoing packet, keeping the same sending and receiving rate at any time. In order to achieve that, the counter should always oscillate around zero.

However, an issue with this solution is that if a client sends more than they receive for a while (e.g., the client is sending a large file), their counter grows without bound. At some point, the client is not protected anymore when incoming bursts occur because the counter is too high. For this reason, we add a threshold t, which signals when the counter should be reset. To determine the optimal threshold value, we capture (sequentially) one instance per website of the analysed dataset and count how many times the proxy had to reset its counter. We want this value to be as small as possible while keeping the threshold value as small as possible. The optimal value of the threshold differs depending on the type of web pages a user often visits.

Evaluation: We evaluate the new cover traffic strategy in the full-scale closed-world setup using the dataset comprising 672 web pages mentioned in Section 3. First, we look into the feature analysis, see Figure 5. As expected, we observe a drop in the importance of the features related to the size and fraction of the incoming traffic. The importance is now more distributed among various groups of features, indicating that our proposed methodology better hides distinctive characteristics of individual web pages. In Figure 6, we depict the difference between the traces generated by a client using the default strategy in comparison to the proposed new one. To this end, we represent each trace using the cumulative (CUM) sum of packets as introduced by Panchenko [39], i.e., from a trace of N packet sizes $T = (p_1, ..., p_N)$ where $p_i > 0$ is an incoming packet and $p_i < 0$ an outgoing packet, the CUMUL representation of this trace is defined as $C(T) = ((0,0), (a_1, c_1), ..., (a_N, c_N))$, where $c_1 = p_1$, $a_1 = |p_1|$, $c_i = c_{i-1} + p_i$, and $a_i = a_{i-1} + |p_i|$.

We can clearly observe, that the bursts of incoming traffic can be easily identified as the currently deployed obfuscation mechanism (red) for the Nym client does very little in hiding the user's activity patterns. In contrast, the usage of drop cover packets which we propose significantly better obfuscates individual communication patterns to foil traffic analysis. This is reflected in the attacker's accuracy and F1-Score, which drops to 0.301 and 0.283 respectively, so by almost 10%. Moreover, as depicted in Figure 7, our technique generates also a lower bandwidth overhead.

7 CONCLUSION

Although the website fingerprinting attack has been widely studied in the context of onion-routing protocols, in particular, Tor. An open question remained whether mix network-based systems offer better protection against the attack. In this work, we analysed the resistance of the Nym mixnet against k-fingerprinting attack in both closed and open-world settings. To this end, we used the actual software, instead of a simulated environment. This allows us to realistically compare the fingerprinting resistance of Nym against Tor. Our results confirm that Nym offers much better protection against attack than Tor. As we showed, this is achieved thanks to the use of cover traffic which obfuscates communication patterns.

To identify which traffic characteristics impact the effectiveness of the classifier, we presented a detailed feature analysis and investigated how different configurations and parameterizations of the Nym client influence how informative each feature becomes for the attack. To understand the performance cost of the strong privacy offered by Nym, we quantified the bandwidth and time overhead. While we show that better resistance against fingerprinting comes at a cost of increased bandwidth overhead, this is a choice which users of anonymous communication systems must consider.

Notably, we identify a flaw in the design of the obfuscation mechanism deployed by the Nym client, i.e., the currently used loop cover traffic fails to hide incoming bursts of communication. To address that issue, we propose an alternative cover traffic mechanism which provides better protection, while yet slightly decreasing the bandwidth overhead.

We hope our work will spark interest in further analysis of mix networks technologies, which in recent years gained much popularity and are currently being developed at a high pace.

REFERENCES

[1] Kota Abe and Shigeki Goto. 2016. Fingerprinting attack on Tor anonymity using deep learning. *Proceedings of the Asia-Pacific Advanced Network* 42 (2016), 15–20.

[2] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. 2019. Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning. *Proc. Priv. Enhancing Technol.* 2019, 4 (2019), 292–310. https://doi.org/10.2478/popets-2019-0070

[3] Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society, WPES 2014, Scottsdale, AZ, USA, November 3, 2014*, Gail-Joon Ahn and Anupam Datta (Eds.). ACM, 121–130. https://doi.org/10.1145/2665943.2665949

[4] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 227–238. https://doi.org/10.1145/2660267.2660362

[5] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a distance: website fingerprinting attacks and defenses. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, Ting Yu, George Danezis, and Virgil D. Gligor (Eds.). ACM, 605–616. https://doi.org/10.1145/2382196.2382260

[6] David Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* 24, 2 (1981), 84–88. https://doi.org/10.1145/358549.358563

[7] 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 - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10355)*, Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi (Eds.). Springer, 557–578. https://doi.org/10.1007/978-3-319-61204-1_28

[8] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. 2015. Riposte: An Anonymous Messaging System Handling Millions of Users. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 321–338. https://doi.org/10.1109/SP.2015.27

[9] Leigh Cuen. 2019. "Nym Technologies Raises $2.5 Million to Anonymize Crypto Apps". https://www.coindesk.com/business/2019/05/13/nym-technologies-raises-25-million-to-anonymize-crypto-apps/.

[10] George Danezis. 2003. Statistical Disclosure Attacks. In *Security and Privacy in the Age of Uncertainty, IFIP TC11 18th International Conference on Information Security (SEC2003), May 26-28, 2003, Athens, Greece (IFIP Conference Proceedings, Vol. 250)*, Dimitris Gritzalis, Sabrina De Capitani di Vimercati, Pierangela Samarati, and Sokratis K. Katsikas (Eds.). Kluwer, 421–426.

[11] George Danezis. 2004. The Traffic Analysis of Continuous-Time Mixes. In *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 3424)*, David M. Martin Jr. and Andrei Serjantov (Eds.). Springer, 35–50. https://doi.org/10.1007/11423409_3

[12] George Danezis and Ian Goldberg. 2009. Sphinx: A Compact and Provably Secure Mix Format. In *30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*. IEEE Computer Society, 269–282. https://doi.org/10.1109/SP.2009.15

[13] George Danezis and Len Sassaman. 2003. Heartbeat traffic to counter (n-1) attacks: red-green-black mixes. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, WPES 2003, Washington, DC, USA, October 30, 2003*, Sushil Jajodia, Pierangela Samarati, and Paul F. Syverson (Eds.). ACM, 89–93. https://doi.org/10.1145/1005140.1005154

[14] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. 2021. The Nym Network: The Next Generation of Privacy Infrastructure. (2021). https://nymtech.net/nym-whitepaper.pdf

[15] Claudia Díaz, Steven J. Murdoch, and Carmela Troncoso. 2010. Impact of Network Topology on Anonymity and Overhead in Low-Latency Anonymity Networks. In *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6205)*, Mikhail J. Atallah and Nicholas J. Hopper (Eds.). Springer, 184–201. https://doi.org/10.1007/978-3-642-14527-8_11

[16] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. 2002. Towards Measuring Anonymity. In *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers (Lecture Notes in Computer Science, Vol. 2482)*, Roger Dingledine and Paul F. Syverson (Eds.). Springer, 54–68. https://doi.org/10.1007/3-540-36467-6_5

[17] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, Matt Blaze (Ed.). USENIX, 303–320. http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html

[18] Roger Dingledine, Vitaly Shmatikov, and Paul F. Syverson. 2004. Synchronous Batching: From Cascades to Free Routes. In *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 3424)*, David M. Martin Jr. and Andrei Serjantov (Eds.). Springer, 186–206. https://doi.org/10.1007/11423409_12

[19] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*. IEEE Computer Society, 332–346. https://doi.org/10.1109/SP.2012.28

[20] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. 1999. Onion Routing. *Commun. ACM* 42, 2 (1999), 39–41. https://doi.org/10.1145/293411.293443

[21] Jiajun Gong and Tao Wang. 2020. Zero-delay Lightweight Defenses against Website Fingerprinting. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, Srdjan Capkun and Franziska Roesner (Eds.). USENIX Association, 717–734. https://www.usenix.org/conference/usenixsecurity20/presentation/gong

[22] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, Thorsten Holz and Stefan Savage (Eds.). USENIX Association, 1187–1203. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/hayes

[23] ORNELLA HERNÁNDEZ. 2022. "Nym Technologies raises $300M to advance internet privacy, sending token price up". https://cointelegraph.com/news/nym-technologies-raises-300m-to-advance-internet-privacy-sending-token-price-up.

[24] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 2009. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the first ACM Cloud Computing Security Workshop, CCSW 2009, Chicago, IL, USA, November 13, 2009*, Radu Sion and Dawn Song (Eds.). ACM, 31–42. https://doi.org/10.1145/1655008.1655013

[25] Andrew Hintz. 2002. Fingerprinting Websites Using Traffic Analysis. In *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers (Lecture Notes in Computer Science, Vol. 2482)*, Roger Dingledine and Paul F. Syverson (Eds.). Springer, 171–178. https://doi.org/10.1007/3-540-36467-6_13

[26] Hopr. 2022. "HOPR secures more than $1M in Web3 ecosystem grants". https://cointelegraph.com/press-releases/hopr-secures-more-than-1m-in-web3-ecosystem-grants.

[27] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul F. Syverson. 2013. Users get routed: traffic correlation on tor by realistic adversaries. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM, 337–348. https://doi.org/10.1145/2508859.2516651

[28] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Díaz, and Rachel Greenstadt. 2014. A Critical Evaluation of Website Fingerprinting Attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 263–274. https://doi.org/10.1145/2660267.2660368

[29] Marc Juárez, Mohsen Imani, Mike Perry, Claudia Díaz, and Matthew Wright. 2016. Toward an Efficient Website Fingerprinting Defense. In *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 9878)*, Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows (Eds.). Springer, 27–46. https://doi.org/10.1007/978-3-319-45744-4_2

[30] Dogan Kesdogan, Jan Egner, and Roland Büschkes. 1998. Stop-and-Go-MIXes Providing Probabilistic Anonymity in an Open System. In *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998, Proceedings (Lecture Notes in Computer Science, Vol. 1525)*, David Aucsmith (Ed.). Springer, 83–98. https://doi.org/10.1007/3-540-49380-8_7

[31] Natasha Lomas. 2021. "Mixnet maker Nym tops up with $13M led by a16z's crypto fund". https://techcrunch.com/2021/11/19/nym-raises-13m-from-a16zs-crypto-fund/.

[32] Natasha Lomas. 2021. "Nym gets 6M for its anonymous overlay mixnet to sell privacy as a service". https://techcrunch.com/2021/07/16/nym-gets-6m-for-its-anonymous-overlay-mixnet-to-sell-privacy-as-a-service/.

[33] Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. 2011. HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society. https://www.ndss-symposium.org/ndss2011/httpos-sealing-information-leaks-with-browser-side-obfuscation-of-encrypted-flows

[34] Steven J. Murdoch and George Danezis. 2005. Low-Cost Traffic Analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (S&P 2005), 8-11 May 2005, Oakland, CA, USA*. IEEE Computer Society, 183–195. https://doi.org/10.1109/SP.2005.12

[35] BRIAN NEWAR. 2022. "New private messaging app claims to be decentralized and quantum-resistant". https://cointelegraph.com/news/new-private-messaging-app-claims-to-be-decentralized-and-quantum-resistant.

[36] Rebekah Overdorf, Mark Juárez, Gunes Acar, Rachel Greenstadt, and Claudia Díaz. 2017. How Unique is Your .onion?: An Analysis of the Fingerprintability of Tor Onion Services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 2021–2036. https://doi.org/10.1145/3133956.3134005

[37] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/website-fingerprinting-internet-scale.pdf

[38] Andriy Panchenko, Asya Mitseva, Martin Henze, Fabian Lanze, Klaus Wehrle, and Thomas Engel. 2017. Analysis of Fingerprinting Techniques for Tor Hidden Services. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, Dallas, TX, USA, October 30 - November 3, 2017*, Bhavani M. Thuraisingham and Adam J. Lee (Eds.). ACM, 165–175. https://doi.org/10.1145/3139550.3139564

[39] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES 2011, Chicago, IL, USA, October 17, 2011*, Yan Chen and Jaideep Vaidya (Eds.). ACM, 103–114. https://doi.org/10.1145/2046556.2046570

[40] Mike Perry. 2011. Experimental website traffic fingerprinting defense. (2011). https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting/

[41] Ania M. Piotrowska. 2021. Studying the Anonymity Trilemma with a Discrete-event Mix Network Simulator. In *WPES '21: Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society, Virtual Event, Korea, 15 November 2021*. ACM, 39–44. https://doi.org/10.1145/3463676.3485614

[42] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. 2017. The Loopix Anonymity System. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, Engin Kirda and Thomas Ristenpart (Eds.). USENIX Association, 1199–1216. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska

[43] Vera Rimmer, Davy Preuveneers, Marc Juárez, Tom van Goethem, and Wouter Joosen. 2018. Automated Website Fingerprinting through Deep Learning. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-1_Rimmer_paper.pdf

[44] Andrei Serjantov and George Danezis. 2002. Towards an Information Theoretic Metric for Anonymity. In *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers (Lecture Notes in Computer Science, Vol. 2482)*, Roger Dingledine and Paul F. Syverson (Eds.). Springer, 41–53. https://doi.org/10.1007/3-540-36467-6_4

[45] Vitaly Shmatikov and Ming-Hsiu Wang. 2006. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4189)*, Dieter Gollmann, Jan Meier, and Andrei Sabelfeld (Eds.). Springer, 18–33. https://doi.org/10.1007/11863908_2

[46] Payap Sirinam, Mohsen Imani, Marc Juárez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM, 1928–1943. https://doi.org/10.1145/3243734.3243768

[47] Paul F. Syverson. 2003. Onion Routing for Resistance to Traffic Analysis. In *3rd DARPA Information Survivability Conference and Exposition (DISCEX-III 2003), 22-24 April 2003, Washington, DC, USA*. IEEE Computer Society, 108–110. https://doi.org/10.1109/DISCEX.2003.1194938

[48] Paul F. Syverson, Gene Tsudik, Michael G. Reed, and Carl E. Landwehr. 2000. Towards an Analysis of Onion Routing Security. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 2009)*, Hannes Federrath (Ed.). Springer, 96–114. https://doi.org/10.1007/3-540-44702-4_6

[49] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. 2015. Vuvuzela: scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles, SOSP 2015, Monterey, CA, USA, October 4-7, 2015*, Ethan L. Miller and Steven Hand (Eds.). ACM, 137–152. https://doi.org/10.1145/2815400.2815417

[50] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, Kevin Fu and Jaeyeon Jung (Eds.). USENIX Association, 143–157. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao

[51] Tao Wang and Ian Goldberg. 2013. Improved website fingerprinting on Tor. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, Ahmad-Reza Sadeghi and Sara Foresti (Eds.). ACM, 201–212. https://doi.org/10.1145/2517840.2517851

[52] Tao Wang and Ian Goldberg. 2017. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, Engin Kirda and Thomas Ristenpart (Eds.). USENIX Association, 1375–1390. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao

[53] Charles V. Wright, Scott E. Coull, and Fabian Monrose. 2009. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, USA, 8th February - 11th February 2009*. The Internet Society. https://www.ndss-symposium.org/ndss2009/traffic-morphing-efficient-defense-against-statistical-traffic-analysis/

A APPENDIX

A.1 Feature Importance in Open world Setting

Feature Idx	Description	Importance
142	Average packet size	0.01071977
143	Average incoming packet size	0.01061818
44	Fraction of outgoing packets	0.01009181
145	Variance of packet sizes	0.01003532
43	Fraction of incoming packets	0.00997685
148	Std of packet sizes	0.00987713
31	Std of num. of outgoing packets per group of 20 packets	0.00941208
144	Average outgoing packet size	0.00936155
150	Std of outgoing packet sizes	0.00928927
147	Variance of outgoing packet sizes	0.00903957
33	Std of packets per second	0.00829567
7	Std of outgoing inter-packet timing	0.00799167
137	Sum of features 0 to 11	0.00798014
146	Variance of incoming packet sizes	0.00794594
149	Std of incoming packet sizes	0.00791801
119	xSums of packets per second split in 20, plus rest	0.00780392
12	25p of incoming packet timing	0.00778222
40	Minimum of packets per second	0.00769579
4	Average outgoing inter-packet timing	0.00758407
117	xSums of packets per second split in 20, plus rest	0.00754518

Table 9: Top 20 features for Nym is open world setting

Feature Idx	Description	Importance
151	Maximum size of incoming packet	0.03149505
9	75p incoming inter-packet timing	0.01611047
143	Average incoming packet size	0.01592640
149	Std of incoming packet sizes	0.01573075
146	Variance of incoming packet sizes	0.01544565
11	75p inter-packet timing	0.01520152
142	Average packet size	0.01373980
148	Std of packet sizes	0.01275595
145	Variance of packet sizes	0.01252173
144	Average outgoing packet size	0.01153761
141	Sum of sizes of outgoing packets	0.01058474
44	Fraction of outgoing packets	0.00978896
10	75p outgoing inter-packet timing	0.00974414
43	Fraction of incoming packets	0.00967728
147	Variance of outgoing packet sizes	0.00959002
150	Std of outgoing packet sizes	0.00952633
140	Sum of sizes of incoming packets	0.00930174
31	Std of num. of outgoing packets per group of 20 packets	0.00923107
33	Std of packets per second	0.00909151
139	Sum of sizes of all packets	0.00874112

Table 10: Top 20 features for Tor is open world setting