Hydra: An Efficient, Provably Secure Website Fingerprinting Defense Based on Traffic Splitting

Alexander Di

Abstract

As Internet privacy becomes an increasing concern in the Digital Age, millions of people have turned to Tor, the world's largest anonymous communication network. However, Tor suffers from website fingerprinting (WF), a type of traffic analysis attack in which an adversary uses a machine learning classifier on a user's web traffic to determine his or her web activity. Although previous studies have proposed various defenses against WF, these defenses either incur high overheads or provide no formal guarantee of privacy. We propose Hydra, a lightweight WF defense that splits the user's traffic among several different Tor nodes. Our results show that Hydra is able to reduce an attacker's accuracy from 92.2% to 7.5% while only incurring 68.2% bandwidth overhead and 59.8% time overhead. In comparison to the existing state-of-the-art defenses, Hydra reduces the attacker's accuracy by over 65% while incurring at least 15% less bandwidth and time overheads.

1 Introduction

In today's Digital Age, the Internet provides users with a platform to freely express their views and disseminate truthful information. However, this is not possible in highly censored countries. Instead, many users ranging from journalists to activists rely on the Tor network, which is the world's largest anonymous communication network with over 2.5 million daily users. [13][12]. The Tor network allows users to evade government censorship and access critical resources unavailable in their country. Tor works by encrypting the web traffic and routing it through various Tor nodes, whereby each Tor node only knows its preceding node and succeeding node, but no other nodes in the circuit, thereby providing privacy for its users.

However, Tor is susceptible to traffic analysis attacks. One such attack that has been extensively studied is website fingerprinting (WF) because it is an easy attack to perform. In a WF attack, an adversary monitors the traffic passing between the victim and the Tor entry node, as shown in Figure 1. The attacker is considered passive in that he does not insert, drop, or delay data packets in the user's traffic; he simply observes.

Although Tor encrypts its data packets and pads each packet to a fixed size of 512 bytes, the adversary can look at the timestamp and direction of the packets to identify patterns in the traffic. During each website visit, the user generates a sequence of data packets with their respective timestamps, which is defined as a *trace*. Typically, the attacker first collects traffic traces of various websites that he seeks to identify. He then extracts features from each of these traffic traces, such as the total number of packets or the average interpacket timing of the trace. Finally, the attacker trains a machine learning classifier on these features and attempts to classify the victim's traffic trace, thereby revealing the website that the victim visited.

Various defenses against WF attacks have been proposed; these defenses seek to modify the traffic trace by delaying certain packets or inserting dummy packets. However, many of these defenses are ineffective as state-of-the-art attacks that employ deep learning have been able to reach over 90% accuracy despite these countermeasures [15]. These attacks leverage the long uninterrupted packet sequences that the defenses fail to break up [10][1]. Other defenses, such as CS-BuFLO and Tamaraw, do significantly decrease attacker accuracy, but at the expense of significant overheads in time and bandwidth [2][3].

In this paper, we propose a highly tunable, provably secure WF defense that is based on traffic splitting and incurs reasonable overheads.

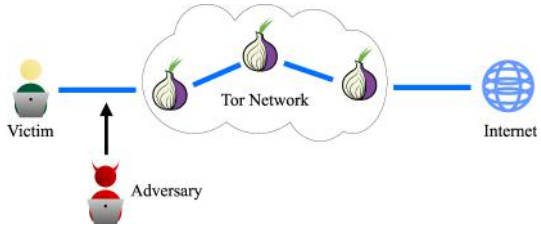

Fig. 1: A local adversary, such as an ISP or a workspace administrator, monitors the traffic between the victim and the Tor entry node. Through the traffic, the attacker can infer the website that the victim is visiting.

2 Literature Review

Various website fingerprinting attacks and defenses have been proposed by researchers over the last few years.

2.1 Existing Attacks

Researchers have proposed various attack models for WF. They evaluate their attacks in two scenarios:
- *Closed world setting*. The victim can only visit a certain amount of web pages. In this setting, the attacker can train on all possible websites that the victim can visit.
- *Open world setting*. The victim can visit any website. The attacker has to also determine whether the victim is visiting a *monitored site*, a site that the attacker trains on, or an *unmonitored site*, a site that the attacker does not train on.

Currently, there are the following major attack models:
• **k-NN** [16]: Wang et al. utilize a k-nearest neighbors classifier to match traffic traces with known websites. k-NN is able to achieve above 80% accuracy on defenses involving traffic morphing.
• **CUMUL** [11]: The attack devised by Panchenko et al. utilizes a support vector machine that takes into account the total number of incoming and outgoing packets, as well as their respective sums. CUMUL is effective in that it does not require training on many instances per website.
• **k-fingerprinting (k-FP)** [6]: Proposed by Hayes et al., this attack uses random forests to classify traffic traces. It is highly effective in an open-world scenario. k-FP allows for classification of other features particular to the relationship between incoming and outgoing packets such as transmission rate per direction [4].
• **Deep Fingerprinting (DF)** [15]: Sirinam et al. proposed an attack that leverages convolutional neural networks. This attack is very effective against defenses such as WTF-PAD (90.7% accuracy) [8] and reaches just below the maximum attacker accuracy of Walkie-Talkie (49.7% accuracy) [17].

2.2 Existing Defenses

Many defenses have been proposed against WF. Typically, defenses are divided into two categories: provably and not provably secure. *Provably secure* defenses have a computable upper bound on the attacker's accuracy. No attacker can achieve a higher accuracy than the upper bound. Such defenses will remain resilient against future attacks.

Defenses also incur overheads, specifically bandwidth and time. *Bandwidth overhead* measures the amount of dummy packets added to a given trace, calculated as the total number of dummy packets divided by the total number of real data packets. *Time overhead* measures the amount of latency or delay in the page load. This is calculated as the extra time taken for the defended page to load divided by the time of an undefended page load.

• **Walkie-Talkie** [17]: This defense pairs sensitive pages with non-sensitive pages so that the morphed traffic trace of both websites are identical. Thus, even if the attacker recognizes the trace, he cannot determine

which website produced the trace. Although WT has a maximum attacker accuracy of 50%, it suffers from top-N prediction since the attacker knows that the trace was produced by one of two websites [7]. In addition, WT requires a database of all the pairings between many different websites.

• **WTF-PAD** [8]: WTF-PAD builds upon Adaptive Padding [14] by sending dummy packets when noticeable gaps appear in the traffic trace. This defense eliminates gaps as a distinguishable feature for an attacker to train on. Although WTF-PAD is a lightweight defense, meaning that it incurs low overheads, it is ineffective against deep learning attacks [15], neither is it provably secure.

• **BuFLO and CS-BuFLO** [5][2]: BuFLO sends packets at a fixed rate such that no distinguishable features are present other than the trace length [5]. If no packets need to be sent, BuFLO sends dummy packets for t seconds either until more data packets enter the queue or until t seconds has passed. Cai et al. built upon BuFLO to create CS-BuFLO, which varies its transmission rate according to the network congestion, thereby reducing the overheads [2].

• **Tamaraw** [3]: Tamaraw, like BuFLO [5], sends packets at a constant rate, hiding the inter-packet timing. However, Tamaraw is able to send outgoing packets at a different rate than incoming packets, further reducing the overheads. In addition, Tamaraw pads each trace to a multiple of a number L to hide the trace length. Tamaraw is especially effective against state-of-the-art attacks [15][6][3]. However, it still suffers from high bandwidth overhead due to the amount of dummy packets injected into the traffic to maintain a constant rate of packet flow, as high as 128% bandwidth overhead.

• **Traffic Splitting** [4]: Instead of padding the traffic, De la Cadena et al. evaluate the several strategies for splitting the traffic among several different Tor entry nodes. They find that using a weighted random splitting strategy greatly reduces the attacker's accuracy (30.27% against DF and 34.09% against k-FP) when traffic is split among three or more different paths.

• **HyWF** [7]: Henri et al. use a different approach by splitting the traffic among two different WiFi networks. This method removes the need to modify Tor nodes but requires the user to have access to two different internet services. HyWF distributes the packets in groups rather than individually and randomly changes the splitting probability, which reduces the attacker's accuracy to around 15.3% and 36.3% against k-NN and k-FP, respectively. Against Deep Fingerprinting, HyWF yields a true positive rate of 48.6%.

3 Purpose

Previous works on WF defenses often incur high overheads or guarantee no upper bound on any attacker's accuracy. The goal of this work is three-fold:

1. Design and implement a provably secure website fingerprinting defense that utilizes traffic splitting (dividing the traffic among multiple Tor entry nodes) while incurring low overheads.
2. Evaluate the effectiveness and efficiency of our defense on existing state-of-the-art attacks.
3. Compare our defense with existing state-of-the-art defenses.

4 Hydra

At a high level, our defense morphs the user's traffic into a constant stream and splits the traffic among several different Tor entry nodes, generating several sub-traces (in this paper, we refer to each individual Tor entry node as a *network* or *path*). In addition, Hydra produces a variety of distinct sub-trace lengths for each website, which allows for a greater chance for the attacker to misclassify traffic traces when he looks for patterns in the trace lengths.

4.1 Hydra Design

Our defense consists of three parts: 1. traffic morphing, 2. traffic splitting, and 3. end padding.

1. *Traffic morphing*. Similar to Tamaraw [3], Hydra utilizes a constant stream of packets. Our defense delays packets and adds dummy packets in order to maintain a pattern of one outgoing packet followed by four incoming packets. The interpacket timing is set to 0.005 seconds.

2. *Traffic splitting*. Hydra utilizes m networks to route the user's traffic. Each network i has its own threshold T_i such that $T_{i-1} < T_i < T_{i+1}$ for $1 \leq i \leq m$, which dictates when to utilize each network. Hydra begins by sending packets through network 1. Since our defense delays packets in order to maintain a strict pattern, packets will accrue in both the server's queue and client's queue. Once the number of packets in either queue reaches T_i, our defense opens up network i and begins sending packets through network i in addition to networks $1, 2, ..., i-1$. Once the number of packets in both queues drops below T_i, network i closes off. After a network is used, it cannot be reopened again. In total, Hydra generates k sub-traces for $1 \leq k \leq m$. For each website visit, the order in which networks are opened is randomized so that the attacker cannot deduce which path he is monitoring.
3. *End Padding*. The traffic morphing part of our defense eliminates distinguishable features such as inter-packet timing and packet ordering. However, the defended trace lengths of a video-streaming service is vastly different than those of a simple search engine. Hydra implements an exponential padding scheme to hide the trace length. Each sub-trace is padded to $\{10\lfloor b \rfloor, 10\lfloor b^2 \rfloor, 10\lfloor b^3 \rfloor, 10\lfloor b^4 \rfloor, ...\}$ packets for $b > 1$. We multiply each power of b by 10 so that our defense, which implements a five-packet pattern, is able to terminate.

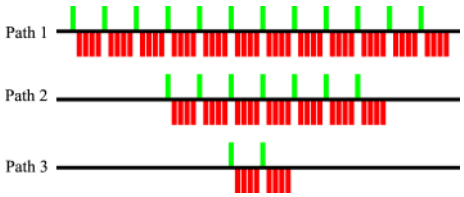

Fig. 2: Hydra opens up new networks as packets accumulate in the queue, which produces sub-traces of varying lengths. In this scenario, an adversary monitoring path 3 does not know whether he is viewing the trace of a small website or that of a large website.

Hydra differs from previous traffic splitting defenses [4][7] in that for every website visit, our defense generates sub-traces of drastically different lengths, further hiding the original trace length (see Figure 2). Thus, when the attacker sees one of the sub-traces, he has no way of determining whether the trace was generated by a large website or a small website.

5 Methods

5.1 Dataset

We used the dataset provided in DynaFlow containing the traffic traces of 100 monitored sites with 100 instances each, as well as 9000 unmonitored sites [9]. However, some traces had unreasonably low packet numbers compared to the total time of the traffic trace. The first quartile of the ratios of packets to total trace time in the original dataset was 140 packets per second. Therefore, we decided to remove all traces that had a packet-time ratio of less than 125 packets per second. The remaining dataset has 91 monitored sites with 50 instances each and 6418 unmonitored sites.

5.2 Implementation

Implementing our defense on the Tor browser would be difficult because it would require modifying the Tor source code. Thus, we wrote Python code to simulate Hydra[1]. With our Python simulation, we produced defended traffic traces from the original traces provided in the dataset. Our Python simulation of Hydra adds delays to certain packets, inserts dummy packets into the traffic, and splits the traffic onto various paths.

Although Hydra creates several different sub-traces from each individual traffic trace, we randomly pick and keep only one of those sub-traces. We make a reasonable assumption that the attacker monitors only one path so that each time the victim visits a website, the attacker will only see one of those sub-traces.

[1] https://github.com/alexdi0421/Hydra

6 Evaluation

We evaluated our defense by running two configurations of Hydra on our modified dataset: (1) For medium security with reasonable overheads, we used three networks and padded to powers of 1.5. (2) For higher security at a greater cost with both overheads and network setup, we used four paths and padded to powers of 2. We then ran k-FP, using the code provided in the paper [6], on the defended dataset generated by Hydra. Finally, we computed an upper bound on any attacker's accuracy.

We did not run DF on Hydra because it uses a large convolutional neural network and was computationally infeasible to run on our machine [15]. Instead, we used k-FP, a state-of-the-art non-deep learning attack [6].

6.1 Closed World Results

We evaluated two configurations of Hydra. The results are summarized in Table 1.

Tab. 1: The accuracy of k-FP on two configurations of Hydra and the original undefended dataset.

Num. Paths	Thresholds	Base	Bandwidth Overhead	Time Overhead	k-FP Accuracy
3	0, 300, 800	1.5	68.2%	59.8%	7.5%
4	0, 200, 400, 800	2	99.7%	43.2%	4.9%
Undefended Traces			N/A	N/A	92.2%

Hydra significantly reduces k-FP's accuracy. In the first configuration with three paths, Hydra is able to significantly reduce the accuracy from 92.2% to 7.5% while only incurring 68.2% bandwidth overhead and 59.8% time overhead. With more paths and more padding, the second configuration further lowers k-FP's accuracy to 4.9% with 99.7% bandwidth overhead and 43.2% time overhead.

6.2 Open World Results

The same configurations of Hydra also significantly decrease the efficacy of k-FP in the open-world (See Figure 3). When no defenses are applied, k-FP easily achieves a 72.6% true positive rate (TPR) with only a 0.8% false positive rate (FPR). When the first configuration of Hydra is applied, k-FP achieves a 3.8% TPR with a 94.8% FPR. As shown in Figure 3, reducing the FPR to 26.4% reduces the TPR to 1.2%. Our second configuration is even stronger. k-FP achieves a 2.5% TPR with a 95.5% FPR, or equivalently, a 0.8% TPR with a 25.9% FPR.

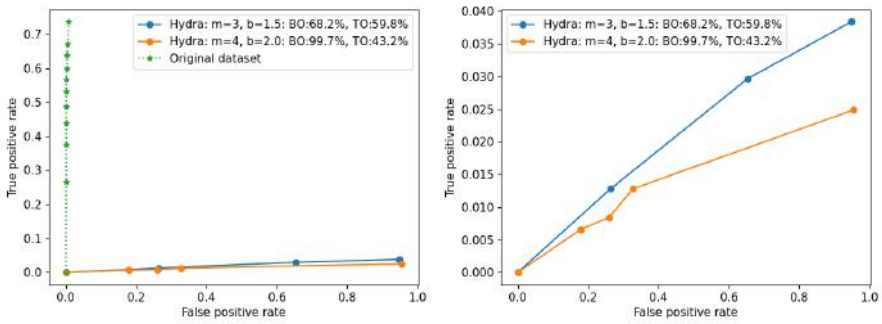

Fig. 3: True positive rate vs. false positive rate of the two configurations of Hydra tested against k-FP. The graph on the left includes the original dataset whereas the graph on the right compares only the two configurations of Hydra. Bandwidth overhead (BO) and time overhead (TO) are labeled in the legend.

6.3 Maximum Attacker Accuracy

Our defense is provably secure because the only distinguishing feature in defended traces is the trace length; other features such as interpacket timing and packet order remain constant for all defended traces. Thus, we can compute an upper bound on any attacker's accuracy. This is the highest accuracy any attacker can achieve assuming that the attacker has perfect knowledge of the nature of the traces generated by each website, i.e., the trace lengths. The MAA is computed as follows:

$$\sum_{i=1} P(l_i) \cdot max\{P(w_1 \mid l_i), P(w_2 \mid l_i), ..., P(w_n \mid l_i)\},$$

where $P(l_i)$ is the probability that a given trace length is l_i and $P(w_k \mid l_i)$ is the probability that website w_k generates a trace with length l_i. $P(w_k \mid l_i)$ is computed as the number of occurrences of l_i in website w_k divided by the total number of occurrences of l_i.

In essence, given a trace of length l_i, the optimal attacker always picks the website that generates the most traces of length l_i. The probabilities of success for each distinct trace length are then summed up to yield an upper bound on any attacker's accuracy (MAA) when Hydra is applied.

6.4 Maximum Attacker Accuracy of Hydra

In the closed world setting, we find that the MAA of Hydra remains below 10% for both configurations (see Table 2). With the same overheads as before, the medium security configuration guarantees a MAA of 8.5% while the higher security configuration promises a 5.6% upper bound on any attacker's accuracy.

Tab. 2: The MAA of both configurations of Hydra.

Num. Paths	Thresholds	Base	Bandwidth Overhead	Time Overhead	MAA
3	0, 300, 800	1.5	68.2%	59.8%	8.5%
4	0, 200, 400, 800	2	99.7%	43.2%	5.6%

In the open world, when Hydra is applied, the optimal attacker still achieves a very low TPR compared to high FPR (see Figure 4). At a 99.5% FPR, the medium configuration of Hydra guarantees a TPR no greater than 3.9%. Similarly, the second configuration of Hydra reduces the maximum TPR to 2.5% with a 99.9% TPR.

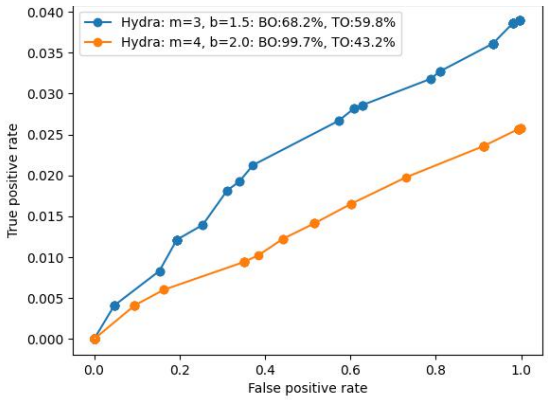

Fig. 4: The attacker's optimal TPRs graphed against their respective FPRs when Hydra is applied.

6.5 Comparison with Other Defenses

We compare our defense with HyWF, the traffic splitting defense proposed by De la Cadena et al. (in this paper, we will refer to this defense as TrafficSplit), as well as Tamaraw, a more heavyweight defense

that is provably secure [3][4][7].

We used our modified version of the dataset provided in DynaFlow [9], which contains 91 monitored sites, each with 50 instances, and 6418 unmonitored sites. Due to compatibility issues, we rewrote the codes for HyWF, TrafficSplit, and Tamaraw [3][4][7]. To run k-FP, we used the original code provided in the paper [6].

6.5.1 Closed World

We use k-FP to evaluate the defenses in addition to the MAA when applicable. The results are summarized in Table 3.

Both TrafficSplit and HyWF do not delay packets, nor do they add dummy packets [4][7]. Thus, both defenses incur no overheads. k-FP is able to achieve a 43.5% TPR against HyWF and 38.6% TPR against TrafficSplit. However, we cannot evaluate an upper bound on the attacker's accuracy because both TrafficSplit and HyWF are not provably secure defenses. Although Hydra incurs moderate overheads, k-FP only achieves a 7.5% and 4.9% accuracy for both the medium security and high security configurations of Hydra, respectively. At the same time, both configurations of Hydra guarantee that any attacker cannot achieve higher than a 8.5% and 5.6% accuracy, respectively.

Tab. 3: The medium and high security configurations of Hydra compared with HyWF, TrafficSplit, and Tamaraw.

Defense	Bandwidth Overhead	Time Overhead	k-FP Accuracy	MAA
Undefended trace	N/A	N/A	92.2%	N/A
HyWF	0%	0%	43.5%	N/A
TrafficSplit	0%	0%	38.6%	N/A
Tamaraw: $L = 500$	52.9%	102.7%	20.1%	21.6%
Tamaraw: $L = 1000$	65.4%	102.7%	13.9%	14.6%
Hydra: $m = 3$, $b = 1.5$	**68.2%**	**59.8%**	**7.5%**	**8.5%**
Hydra: $m = 4$, $b = 2$	**99.7%**	**43.2%**	**4.9%**	**5.6%**

Compared to Tamaraw, our defense yields lower attacker accuracies as well as lower overheads. We adjust Tamaraw's parameters so that it yields low overheads on our dataset. Both configurations of Tamaraw have outgoing and incoming interpacket timings of 0.005 seconds and 0.025 seconds, respectively, and a padding parameter L. The MAA of both configurations of Hydra (8.5% and 5.6%) is lower than that of Tamaraw (21.6% and 14.6%), as well as k-FP's accuracy (7.5% and 4.9% compared to 20.1% and 13.9%). In fact, the aggregate overheads (the sum of bandwidth overhead and time overhead) of both configurations of Hydra, 128.0% and 142.9%, are lower than those of Tamaraw, 155.6% and 168.1%, respectively.

6.5.2 Open World

Both configurations of Hydra outperform TrafficSplit, HyWF, and Tamaraw (see Figure 5). At a 0.0% FPR, TrafficSplit and HyWF only reduce k-FP's TPR to 3.1% and 5.6%, respectively, whereas both configurations of Hydra drop the TPR to 0%.

Hydra outperforms Tamaraw in an open world setting. At a FPR of 98.8%, the higher security configuration of Tamaraw ($L = 1000$) only promises a TPR of 6.1% whereas our medium configuration of Hydra drops the TPR to 2.5% at a FPR of 99.9%. Similarly, at a 38.1% FPR, Tamaraw only guarantees a 3.1% TPR compared to the 1.0% TPR that the higher security configuration of Hydra guarantees. As stated before, the aggregate overheads of Hydra are less than those of Tamaraw.

6.6 Hydra's Parameters

Hydra offers high tunability; it can be configured for a wide range of MAAs, bandwidth overheads, and time overheads. Here, we evaluate the effects of changing Hydra's parameters.

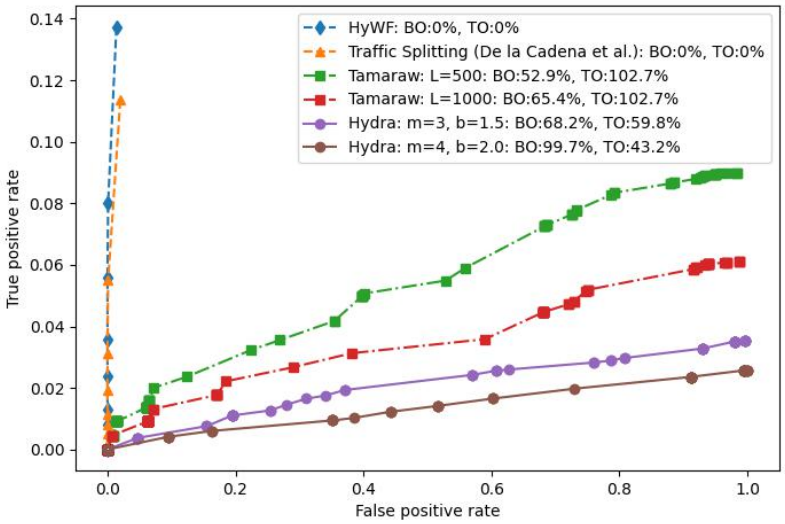

Fig. 5: The TPR graphed against the FPR for Hydra, TrafficSplit, HyWF, and Tamaraw. We used k-FP to evaluate the TPR and FPR of TrafficSplit and HyWF. We graphed the optimal attacker's TPR and FPR when Hydra and Tamaraw were applied.

6.6.1 Number of networks

We run Hydra with a base b of 1.5 on different numbers of networks. As the number of paths increases from two to six, the MAA decreases from 10.1% to 6.9% (see Figure 6). In addition, while the bandwidth overhead remains at 69.2 $\pm$ 1.1%, the time overhead drops from 84.4% with two paths to 37.5% with six paths. However, we noticed that the MAA of six paths is only 0.1% less than that of five paths (6.9% compared to 7.0%). In fact, we suggest that the cost of setting up more than five networks outweighs the guaranteed privacy of five or more networks.

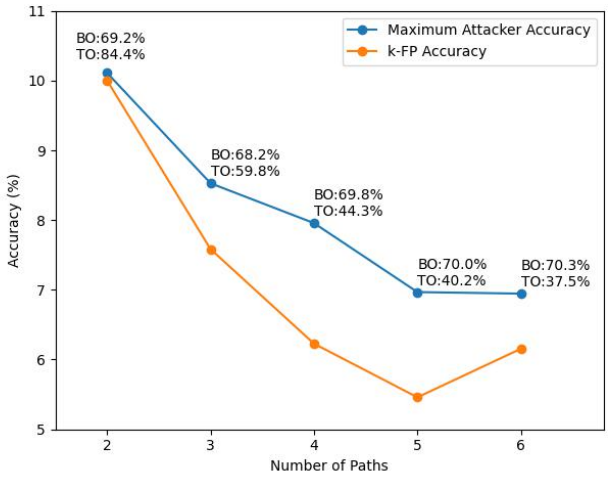

Fig. 6: The change in the maximum attacker accuracy (MAA) and k-FP accuracy (closed world) with the number of paths. Configurations with the same number of paths have the same bandwidth overhead (BO) and time overhead (TO). $b = 1.5$ for each configuration.

6.6.2 Change of Base

Increasing the base b from 1.1 to 1.5 to 2 increases the bandwidth overhead from 44.2±0.5% to 69.0±0.8% to 99.1±0.7%, regardless of the number of paths used (see Table 4). The MAA decreases as well. For instance, when changing the base for four networks, the MAA drops from 12.2% to 8.0% to 5.6% (also see Figure 7).

Tab. 4: The overheads and accuracies of different configurations of Hydra. These results are graphed in Figure 7.

Num. Paths	Thresholds	Base	Bandwidth Overhead	Time Overhead	k-FP Accuracy	MAA
2	0, 400	1.1	43.8%	85.5%	16.2%	18.2%
2	0, 400	1.5	69.2%	84.4%	10.0%	10.1%
2	0, 400	2	99.3%	83.6%	7.1%	7.3%
3	0, 300, 800	1.1	44.0%	61.3%	10.0%	14.1%
3	0, 300, 800	1.5	68.2%	59.8%	7.5%	8.5%
3	0, 300, 800	2	98.4%	58.6%	6.2%	6.5%
4	0, 200, 400, 800	1.1	44.7%	46.2%	8.3%	12.2%
4	0, 200, 400, 800	1.5	69.8%	44.3%	6.2%	8.0%
4	0, 200, 400, 800	2	99.7%	43.2%	4.9%	5.6%

However, for configurations with the same number of paths, the time overhead remains relatively constant, regardless of the base. Configurations with two, three, and four networks incur 84.6±1.0%, 60.0±1.4%, and 44.7±1.5% time overhead, respectively. This is because changing the base does not further delay any packets, but rather changes the amount of padding added to the end of the trace. Thus, the final time overhead remains the same.

Based on these observations, we suggest that if the user wishes for a faster surfing experience, the user should allow more time for Hydra to configure more paths. If the user is concerned with network congestion, the user should use a smaller base.

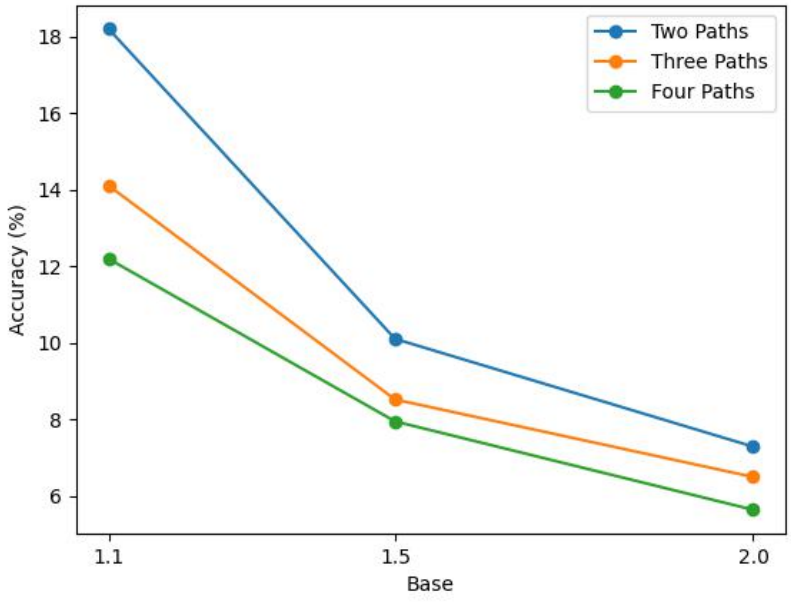

Fig. 7: The effect of changing the base b on the maximum attacker accuracy (MAA) for different numbers of networks.

7 Discussion

Hydra is effective in that it allows the same website to generate traces of different lengths. Since the only distinguishing feature in Hydra's defended traces is the trace length, the attacker is more prone to misclassification due to the variety of trace lengths that a single website can produce.

Additionally, Hydra is a provable defense, which means that our defense guarantees an upper bound on any attacker's accuracy. As future WF attacks become more powerful, Hydra will remain secure due to its provable nature.

In fact, Hydra provides lower overheads than existing state-of-the-art provable defense [3][2]. Even when Hydra produces bandwidth overheads equivalent to those of Tamaraw, not only does Hydra yield lower attacker accuracies, but Hydra also reduces the time overhead by almost 50%. This is in part due to Hydra's ability to split the traffic among several different Tor nodes, thereby speeding up the transmission rate and reducing the latency.

When deployed in the real world, Hydra also offers an additional layer of security. With Hydra's traffic-splitting strategy, some website visits may not trigger the usage of certain networks. This leads some networks to remain closed during the entire website visit. In our evaluation, we calculated the MAA and k-FP's accuracy with the assumption that the attacker would always see a non-empty sub-trace, which is not always the case in the real world. Thus, Hydra provides yet another layer of security in which the attacker may not even be aware that a website visit has occurred.

8 Future Work

In this section, we discuss the possible areas of study for future work.

Other Attacks. In this paper, we did not evaluate the efficacy of DF against Hydra due to the amount of computational power required to run DF. However, we hypothesis that DF will achieve higher accuracies than k-FP. Whereas k-FP focuses mainly on features related to timing [6], DF disregards the timing all together [15]. Thus, we assume that DF may perform better than k-FP since Hydra normalizes interpacket timing. In our future work, we plan to investigate DF's accuracies against Hydra.

Traffic splitting latency. Setting up more networks for Hydra to route the user's traffic may take up more time compared to using one path. At the very least, Hydra must connect to m different Tor entry nodes. Although Hydra shows that increasing the number of paths decreases the time overhead, using too many paths may in fact overload the entry nodes, thereby increasing the time overhead. In our future work, we plan to calculate the cost of setting up networks and bandwidth overload and determine the optimal number of paths to use.

End of page load. Hydra stops sending packets when the total trace length is padded to a power of b and when there are no more packets in the queue. However, if the total trace length reaches a power of b and no packets are in either the client's queue or the server's queue, our defense terminates regardless of whether the website has finished loading or not. This could potentially result in a partial page load. In our evaluation, we kept Hydra running until all packets from the original trace were sent. However, in the real world, Hydra would not know when to stop sending packets. This is also a problem with other constant-stream defenses [5][3]. We propose a solution involving sending a dummy packet at the very end of a page load that signals to Hydra that the page has finished loading.

Parameters of Hydra. Hydra offers high tunability; the user may configure the number of paths to use and the amount of padding to add. However, the relationship between the thresholds and the overheads (as well as the accuracy) is murky. In our evaluation, we tested many different thresholds for different numbers of networks and chose the thresholds that yielded the least overheads, which are listed in Table 5. We plan to determine a formula to optimize the thresholds for any given number of networks in future work.

9 Conclusion

In this paper, we developed Hydra, an efficient, provably secure defense based on traffic splitting that offers high tunability. Hydra splits the user's traffic into various streams and normalizes the traffic trace to

Tab. 5: The thresholds used in our evaluation.

Num. Paths	Thresholds
2	0, 400
3	0, 300, 800
4	0, 200, 400, 800
5	0, 200, 400, 800, 1200
6	0, 200, 600, 400, 800, 1200

remove distinguishable features from the attacker. Each website visit generates several different sub-traces of drastically different lengths, which allows for many sites' trace lengths to collide. Hydra is able to drop any attacker's accuracy from 92.2% to 8.5% while only incurring 68.2% bandwidth overhead and 59.8% time overhead. Thus, Hydra promises both efficiency and efficacy when existing state-of-the-art defenses only guarantee one.

10 Acknowledgments

I would like to thank my mentor David Lu for supporting me and providing valuable resources during this project.

References

[1] Ahmed Abusnaina, Rhongho Jang, Aminollah Khormali, DaeHun Nyang, and David Mohaisen. Dfd: Adversarial learning-based approach to defend against website fingerprinting. 2020.

[2] Xiang Cai, Rishab Nithyanand, and Rob Johnson. Cs-buflo: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, page 121–130, New York, NY, USA, 2014. Association for Computing Machinery.

[3] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 227–238, New York, NY, USA, 2014. Association for Computing Machinery.

[4] Wladimir De la Cadena, Asya Mitseva, Jan Pennekamp, Jens Hiller, Fabian Lanze, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. Poster: Traffic splitting to counter website fingerprinting. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2533–2535, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, page 332–346, USA, 2012. IEEE Computer Society.

[6] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1187–1203, Austin, TX, August 2016. USENIX Association.

[7] Sébastien Henri, Gines Garcia-Aviles, Pablo Serrano, Albert Banchs, and Patrick Thiran. Protecting against website fingerprinting with multihoming. volume 2020, pages 89 – 110, Berlin, 2020. Sciendo.

[8] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an efficient website fingerprinting defense. volume 9878, pages 27–46, 09 2016.

[9] David Lu, Sanjit Bhat, Albert Kwon, and Srinivas Devadas. Dynaflow: An efficient website fingerprinting defense based on dynamically-adjusting flows. In *Proceedings of the 2018 Workshop on Privacy*

in the Electronic Society, WPES'18, page 109–113, New York, NY, USA, 2018. Association for Computing Machinery.

[10] N. Mathews, P. Sirinam, and M. Wright. Understanding feature discovery in website fingerprinting attacks. In *2018 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pages 1–5, 2018.

[11] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *NDSS*, 2016.

[12] The Tor Project. Who uses Tor? 2019. `https://2019.www.torproject.org/about/torusers.html.en`.

[13] The Tor Project. Users - Tor Metrics. 2020. `https://metrics.torproject.org/userstats-relay-country.html`.

[14] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of the 11th European Conference on Research in Computer Security*, ESORICS'06, page 18–33, Berlin, Heidelberg, 2006. Springer-Verlag.

[15] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1928–1943, New York, NY, USA, 2018. Association for Computing Machinery.

[16] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, San Diego, CA, August 2014. USENIX Association.

[17] Tao Wang and Ian Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1375–1390, Vancouver, BC, August 2017. USENIX Association.