Computer Communications 228 (2024) 107976

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Practical and lightweight defense against website fingerprinting ☆

Colman McGuan [a,*], Chansu Yu [a], Kyoungwon Suh [b]

[a] Department of Electrical and Computer Engineering, Cleveland State University, 2121 Euclid Avenue, Cleveland, 44115, OH, United States of America
[b] School of Information Technology, Illinois State University, Campus Box 5150, Normal, 61790, IL, United States of America

ARTICLE INFO

Keywords:
Beta distribution
Censorship
Cybersecurity
Deep learning
Machine learning
Rayleigh distribution
Website fingerprinting

ABSTRACT

Website fingerprinting is a passive network traffic analysis technique that enables an adversary to identify the website visited by a user despite encryption and the use of privacy services such as Tor. Several website fingerprinting defenses built on top of Tor have been proposed to guarantee a user's privacy by concealing trace features that are important to classification. However, some of the best defenses incur a high bandwidth and/or latency overhead. To combat this, new defenses have sought to be both lightweight — i.e., introduce a small amount of bandwidth overhead — and zero-delay to real network traffic. This work introduces a novel zero-delay and lightweight website fingerprinting defense, called BRO, which conceals the feature-rich beginning of a trace while still enabling the obfuscation of features deeper into the trace without spreading the padding budget thin. BRO schedules padding with a randomized beta distribution that can skew to both the extreme left and right, keeping the applied padding clustered to a finite portion of a trace. This work specifically targets deep learning attacks, which continue to be among the most accurate website fingerprinting attacks. Results show that BRO outperforms other well-known website fingerprinting defenses, such as FRONT, with similar bandwidth overhead.

1. Introduction

It is no secret that Internet surveillance exists and poses a persistent threat to individual privacy [1,2]. One such type of surveillance is website fingerprinting, which seeks to uncover the website a user visits despite the use of privacy services such as Tor by exploiting patterns in network traffic. Using website fingerprinting, an adversary can detect whether a user visited a website targeted for surveillance with high accuracy [3–9].

Classic state-of-the-art attacks utilize machine learning [3–5,8,10] and exploit features such as packet ordering statistics, the number of incoming and outgoing packets, packet lengths, etc. More recent attacks [6,7,9,11] leverage deep learning, which does not burden the attacker with manually selecting a feature set. Furthermore, deep learning attacks continue to pose a threat as some of the most accurate attacks even against modest defenses [6,12]. In response to attacks, many defenses [10,12–18] have been proposed to conceal sensitive features of website traffic used in classification. However, some of the best defenses require a high bandwidth and/or latency overhead to be carried out [10,13,15,17]. Bandwidth overhead burdens the network with useless traffic and latency overhead diminishes user experience by increasing webpage load times. Moreover, some defenses that strive to be lightweight with zero latency remain vulnerable to deep learning classifiers [6]. As such, these defenses have not been implemented in any privacy service such as Tor.

In light of these factors, this research proposes a novel website fingerprinting defense titled Beta Randomized Obfuscation (BRO) which is both lightweight and introduces zero latency overhead. BRO heavily disrupts both the feature-rich beginning of a trace as well as important packet ordering statistics; both of which are crucial for trace classification [4,12]. BRO is evaluated in several metrics to show its effectiveness and practicality. BRO is demonstrated to be effective against state-of-the-art attacks including a deep learning attack, Deep Fingerprinting (DF) [6], widely considered to be one of the best attacks.

BRO diminishes the accuracy of Deep Fingerprinting to below all of the machine learning attacks used in the experiment despite initially performing better on undefended traffic. Furthermore, BRO outperforms FRONT [12], another lightweight and zero-delay website fingerprinting defense, with similar bandwidth overhead.

The contributions made by this work are twofold. First and foremost, this paper demonstrates that BRO is effective at diminishing the accuracy of deep learning website fingerprinting attacks over that of previously proposed zero-delay and lightweight website fingerprinting defenses. Secondly, this work examines what sets BRO apart from FRONT with respect to the clustering and location of dummy packets

☆ This work was supported in part by the U.S. National Science Foundation under Grant 2028397.
* Corresponding author.
E-mail addresses: c.mcguan@vikes.csuohio.edu (C. McGuan), c.yu91@csuohio.edu (C. Yu), kwsuh@ilstu.edu (K. Suh).

https://doi.org/10.1016/j.comcom.2024.107976
Received 22 June 2023; Received in revised form 25 June 2024; Accepted 10 October 2024
Available online 16 October 2024

applied by each defense. Experimental results indicate that dummy packets in BRO are clustered closer together and experience a higher degree of trace-to-trace randomness concerning their location in a defended trace than in FRONT.

This work is organized as follows. Section 2 gives background information related to website fingerprinting including the mechanics of Tor and the premise of attacks and Section 3 describes related work. Section 4 gives the necessary definitions to understand bandwidth and latency overhead. Section 5 describes the implementation and fine-tuning of the BRO defense and analyzes what sets BRO apart from FRONT. Section 6 details the experimental setup used in this research for data collection and defense evaluation. Section 7 presents the findings of this work and finally, Section 8 gives the conclusion and future work.

2. Background

2.1. Tor background

Research on website fingerprinting consistently uses Tor as a foundation for testing [3,4,6,8,12–17]. This is due to Tor's reputation as an effective anonymity network used to provide privacy to users. Tor creates a circuit typically consisting of three nodes; network traffic is sent from the client through the circuit and at each node, a layer of encryption is removed. The use of multiple layers of encryption and a multi-node routing methodology makes it impossible for any network intermediary to know both the origin and final destination of the communication; this forms the premise of website fingerprinting attacks—identifying the website visited despite the use of Tor. Furthermore, Tor packs network traffic into packets — termed cells by the Tor Project — of fixed size, nullifying the use of individual packet lengths as a feature for classification. In this work cell and packet are used interchangeably.

2.2. Attacking Tor traffic

Prior work [3,4,6,8,9] has shown that a passive analysis of encrypted network traffic is effective at identifying the website visited by a user. As such, this work considers an adversary consistent with other research on website fingerprinting: a passive adversary situated between the user and the Tor entry node. In this sense, passive means that the adversary does not disrupt the network traffic of the user—they do not delay, drop, or inject any packets in the stream. The adversary knows the identity of the user but is unaware of which websites they are browsing. To design a successful attack, the adversary collects their own sequence of packets (AKA a trace) sourced from/destined to websites they would like to be able to identify and feeds the traces into a classifier for training. The training process extracts important and identifying features from each website. Once the adversary has trained their classifier, they collect traces from the user's connection and attempt to identify the website visited using their classifier.

3. Related work

Website fingerprinting is a thoroughly researched topic dating as far back as the late 1990s with analysis of HTTP traffic encrypted by SSL [19,20]. Research on website fingerprinting is broadly divided into two categories of work: attacks and defenses. Website fingerprinting attacks are concerned with the classification of encrypted network traffic using machine learning or deep learning. Website fingerprinting defenses attempt to overcome attacks with as little disruption to the user as possible. At their core, most defenses boil down to introducing some sort of delay and/or a reordering of network traffic and/or padding traffic with dummy packets. Attacks and defenses can be evaluated in both the closed- and open-world setting. In the open-world setting, websites are split into two categories: monitored and unmonitored. The monitored set is typically much smaller. The attacker must determine if a trace belongs to either the monitored or unmonitored set. This presents a more realistic challenge to the attacker as it mimics real web browsing behavior. In the closed-world setting, all websites are considered monitored, and the attacker must choose which website a trace belongs to. This presents a challenge to a defense as the classification problem for the attacker is much smaller.

3.1. Website fingerprinting attacks

Website fingerprinting attacks have become increasingly accurate in recent years in both the closed- and open-world scenario with some [3,6] boasting precision in the open-world setting of over 90% on undefended traffic. Many older attacks [3,4,8,10], utilize machine learning for classification whereas some newer attacks [6,7,9,11] are beginning to use deep learning.

3.1.1. FineWP

Shen et al. [5] argue that previous research on website fingerprinting has solely focused on individual, differing websites and ignored website fingerprinting of multiple webpages from the same website—e.g., multiple items on a shopping website. They propose FineWP, which splits a trace into two distinct states: uplink and downlink. An uplink-dominant stage is defined as when the client is making requests to the server; specifically, when there are at least 4 uplink packets for every downlink packet. A downlink-dominant stage is the inverse. Traditionally, a packet in a sequence is classified as either positive or negative for outgoing and incoming traffic; FineWP alters this by marking outgoing packets as 0. In this way, when a cumulative sum of a trace is taken, the uplink-only stages are clearly defined.

To classify a webpage, they use block features, sequence features, and statistical features. Block features describe the number and location of uplink-only blocks within a trace. Sequence features define the location and length of uplink-dominant stages (which can consist of 1 or more uplink-only blocks). Statistical features consist of statistics such as minimum, maximum, mean, standard deviation, etc. on the uplink packets, downlink packets, and the entire trace together. These features are input into a random forest classifier. The authors note that, like other attacks, theirs is susceptible to the fact that website content is not necessarily static. Therefore, the accuracy of their attack degrades with time and requires a retraining of the classifier with updated traces.

3.1.2. k-Fingerprinting

k-Fingerprinting [4] is an attack that utilizes a random forest classifier in conjunction with k-nearest neighbors. The attack first generates what the authors denote as a fingerprint using the random forest classifier then performs the classification of the trace using the k-nearest neighbors of the training data. Note however that only the random forest is used in the closed-world scenario.

The authors of [4] also make notable contributions to the importance and ranking of different trace features for classification—something that was lacking previously. They calculate the importance of 150 features and find that some of the most important features include the total number of incoming packets and several packet ordering statistics.

3.1.3. CUMUL

CUMUL [3] is proposed by Panchenko et al. as an attack leveraging the cumulative sum of the packet lengths in a trace as a feature. The cumulative sum as a feature allows for easy visualization of the shape of the trace (such as where in the trace there are more outgoing packets than incoming and vice versa). The authors use a support vector machine (SVM) as the classifier with an RBF kernel. Notably, the authors tune and optimize their feature sampling rate to create the most accurate attack without burdening the attacker with high computational overhead. The result is an accurate attack with comparatively smaller computation time.

3.1.4. Deep fingerprinting

In 2018, Sirinam et al. proposed Deep Fingerprinting (DF) [6] as a state-of-the-art website fingerprinting attack. DF leverages a deep convolutional neural network (CNN) to fingerprint encrypted network traffic on Tor. Deep Fingerprinting differs from conventional website fingerprinting attacks in that the attacker is not forced to determine which features are most important for classification—it has automatic feature extraction. Furthermore, the attack only requires a sequence of packet directions as input. Their implementation is widely considered to be one of the most effective and state-of-the-art attacks due to its ability to perform well even against moderate defenses.

3.2. Website fingerprinting defenses

Website fingerprinting defenses employ a myriad of different strategies, attempting to conceal trace features important for classification. Below gives an overview of different strategies.

3.2.1. Walkie-Talkie

Walkie-Talkie (WT) [17] is a defense that implements half-duplex communication of HTTP requests between the client and server; that is, between the client and the server, only one is sending data at a time. The half-duplex implementation only needs to be implemented on the client side—i.e., the client will queue up any additional requests until it has received all responses to the outstanding requests from the server at which point it will send the next burst of requests. On top of that, two websites are paired with each other and padded in burst sequences; the front of the traces are aligned and the trace with the smaller burst is padded to the length of the larger burst. If the traces are not the same size, the shorter trace is padded out with the remaining bursts from the longer trace.

The goal of Walkie-Talkie is to create collisions between two websites by requiring that the loading of paired websites generates the same trace sequence. The main drawbacks of WT are that the client and proxy must maintain a database of burst sequences for the padding step and the delay introduced to webpage loading by half-duplex communication.

3.2.2. Tail Time

Tail Time (TT) [15] is proposed by Liang et al. as an improvement to Walkie-Talkie. As such, it is built on top of WT, sharing the same underlying defense. The implementation of TT in [15] chooses a single reference sequence (the longest trace in the dataset) to be used for all webpage loads during the padding step. TT earns its name through its concern for page loading time—WT suffers from long page-loading times due to its strict use of half-duplex communication. TT does not allow queued requests to wait for outstanding requests longer than the configured tail time value, thereby decreasing page-loading time over WT. In [15], the authors uncover that WT suffers from long page loading times due to webpage resources (such as ads) being given a lower priority to process other resources more quickly, but the half-duplex nature of WT does not allow for any new requests to be sent until all outstanding ones are received. This causes the queued requests to block for slow resources. Importantly, TT does not experience an increase in fingerprintability over WT.

3.2.3. FRONT

Gong et al. propose FRONT [12] as a zero-delay and lightweight defense. Similar to TT, FRONT is concerned with the delay associated with many state-of-the-art website fingerprinting defenses. FRONT builds off previous research that the very beginning of a trace is an important feature for identifying which website a trace belongs to. To combat this, FRONT obfuscates the beginning of a trace by using a Rayleigh distribution. A Rayleigh distribution is shaped in such as way that the dummy packets sampled from it are more likely to have values closer to 0—i.e., the beginning of the trace. Both the client and proxy randomly sample a padding budget (the number of dummy cells to be sent) and a padding window (to determine where most of the dummy packets will be injected). Both properties are sampled from a uniform distribution. The Rayleigh distribution is shaped by the padding window and the dummy packets are then sampled from the distribution. The client and proxy perform each step of the defense independently to enhance the degree of randomness. Any dummy packets that are scheduled to be sent after the last real packet are canceled, which decreases the bandwidth overhead and keeps the latency overhead at zero. FRONT outperforms another lightweight defense, WTF-PAD [16], which was previously defeated by Deep Fingerprinting.

3.2.4. GLUE

GLUE [12] is built in conjunction with FRONT to provide coverage between multiple website loads. The defense attempts to elongate and merge several traces by sending dummy packets between the client and proxy until another browsing request is made or a timeout value is reached. This forces the attacker to find a split point where one trace ends and the next one starts. A client will start in FRONT mode during an initial browsing request. When the website finishes loading, GLUE mode is entered where dummy packets are exchanged back and forth between the client and proxy. If a request for a new website is made before GLUE mode times out, BACK mode is entered, and the website is loaded as though it were an undefended trace. Once the new page loads, GLUE mode is reentered. The state will continue to switch between GLUE mode and BACK mode until GLUE mode times out, at which point it will return to FRONT mode. The attacker is forced to find a split point potentially for several websites.

In [12], the authors devise their own attack to identify the split point of a given trace defended by FRONT and GLUE, and from there identify which website each subtrace belongs to. They demonstrate that even if the attacker is given the correct number of subtraces within a defended trace (i.e., they know exactly how many websites the user visited within one trace), identifying where to split the trace still proves difficult.

3.2.5. QCSD

HTTP/3 will not support TCP and will instead use QUIC, a protocol built on top of UDP aimed at speeding up communication [21]. As such, research on the effects of QUIC on website fingerprinting has become more prevalent, such as QCSD [22]. QCSD emulates existing website fingerprinting defenses on top of the QUIC protocol.

QCSD has two main advantages over the conventional implementation of website fingerprinting defenses: (1) QUIC is implemented in user space on top of UDP, which enables defenses to be deployed on a per-application basis; and (2) QCSD is implemented strictly on the client side. The second point is particularly important as it does not require coordination between the client and proxy to implement a defense, which contrasts with the standard implementations of many state-of-the-art website fingerprinting defenses [12,14–17]. By omitting the requirement for a proxy to participate in the defense, the implementation is greatly simplified and a defense can be packaged into a browser.

In [22], FRONT [12] and Tamaraw [18] are emulated. The authors use the QUIC `PADDING` and `PING` control frames to send client-to-server padding, and they create what they term *chaff-steams* to pad in the server-to-client direction by forcing the server to send chaff data using `HTTP GET` requests. Furthermore, the implementation of QUIC in user-space enables them to delay traffic to shape the transmission as needed. Using these methods, they are able to successfully emulate both FRONT and Tamaraw with some important insights on the difference in both latency and bandwidth overhead—e.g., that defenses with smaller bandwidth overhead remain small when implemented in QCSD but those with larger bandwidth overheads increased. Particularly of note is that the FRONT implementation with QCSD is able to successfully defend against attacks with results similar to the simulated version.

4. Preliminaries

Here, definitions necessary to understand the criteria used to evaluate website fingerprinting defenses are given. First, overhead definitions are given. Then, definitions of accuracy and True Positive Rate (TPR), which are used to measure the performance of a website fingerprinting attack against a defense, are given. In some works, accuracy and TPR are used somewhat interchangeably; however, this work makes a distinction between the two. Accuracy is used in the closed-world where the attacker knows that all websites are monitored and is limited to choosing which website a trace belongs to. In contrast, TPR is used in the open-world where attacker is also given the option of labeling a trace as unmonitored. This distinction yields the two equations below.

4.1. Bandwidth overhead

Bandwidth overhead describes the additional traffic that the network must bear to carry out a defense. Two metrics are used to evaluate the defense in this manner: the mean bandwidth overhead and the median bandwidth overhead. Both metrics provide insight into the effect a defense has on an individual trace. It is important to note that the dummy packets are only exchanged between the client and the proxy (i.e., the Tor entry node) and that the resulting bandwidth overhead is thus limited to this portion of the network. The destination server does not receive dummy packets and furthermore is unaware of the defense altogether. Limiting the bandwidth overhead to this portion of the network has the effect of decreasing the amount of time and number of hops that dummy packets spend traversing the network, thereby minimizing congestion. Moreover, it makes a defense simpler to implement by only having 2 entities taking part. The following definitions are given to succinctly define bandwidth overhead.

4.1.1. Trace

A trace is a sequence of packets where each index defines a tuple containing a timestamp t and the direction d. Since Tor cells are all the same size, +1 and −1 are used to define outgoing and incoming cells respectively. The sequence is ordered by timestamp. Note that in this research each trace consists of 15 s worth of packets. Therefore, a trace is defined by the following.

$$T = \{(t_1, d_1), (t_2, d_2), \ldots, (t_n, d_n)\} \tag{1}$$

4.1.2. Bandwidth overhead of a single trace

The bandwidth overhead definition of a trace from [12] is maintained. The bandwidth overhead of a single trace explains how much additional padding is applied to the trace relative to its original length. Let T denote a trace before padding and T' denote a trace after padding. The bandwidth overhead O of a single trace T is given by the following equation where $|T|$ is the number of packets in T.

$$O(T) = \frac{|T'| - |T|}{|T|} \tag{2}$$

4.1.3. Mean bandwidth overhead

The mean bandwidth overhead applied by a defense gives a metric of how much bandwidth overhead is applied to an individual trace on average. It is calculated by taking the sum of the bandwidth overhead for each individual trace and dividing that by the number of traces in the dataset. The mean bandwidth overhead of a defense D is given by the following equation where N defines the total number of traces in the dataset.

$$O_{mean}(D) = \frac{\sum_{i=1}^{N} O(T_i)}{N} \tag{3}$$

4.1.4. Median bandwidth overhead

The median bandwidth overhead of the defense defines the median bandwidth overhead observed in the defended dataset. This is a useful metric to show the skew in the distribution of bandwidth overhead. The median bandwidth overhead of a defense D is given by the following equation where $S_{bandwidth}$ defines a sorted array of all individual trace bandwidth overheads and N defines the total number of traces in the dataset.

$$O_{median}(D) = S_{bandwidth}\left[\left\lfloor \frac{N}{2} \right\rfloor\right] \tag{4}$$

4.2. Latency overhead

The latency overhead describes the additional time required to carry out a defense relative to the original trace. To be consistent with related work [12,17] bandwidth overhead and latency overhead are independent; that is, dummy packets sent before the last real packet do not contribute to latency overhead. Similar to bandwidth overhead, a defense is evaluated by the mean latency overhead and the median latency overhead. The mean latency overhead provides a metric for how much latency a defense applies to individual traces on average. The median is included as a metric as well to indicate skew. The relevant definitions for latency overhead are given below.

4.2.1. Latency overhead of a single trace

The latency overhead of a single trace is defined by the additional time required to carry out a defense on a trace divided by the time taken to load the undefended trace. Let T_n denote the final timestamp in an undefended trace and T'_m denote the final timestamp in the same trace with a defense applied. Then, the latency overhead L of a single trace T is given as the following.

$$L(T) = \frac{T'_m - T_n}{T_n} \tag{5}$$

4.2.2. Mean latency overhead

The mean latency overhead defines the mean latency that a defense applies to traces within the dataset. It can be computed by taking the sum of latency overhead for each individual trace and dividing that by the number of traces in the dataset. The following equation gives the mean latency overhead for a defense D where N is the number of traces in the dataset.

$$L_{mean}(D) = \frac{\sum_{i=1}^{N} L(T_i)}{N} \tag{6}$$

4.2.3. Median latency overhead

The median latency overhead provides another measure of how much latency overhead a defense applies to traces within the dataset. The median latency overhead of a defense D is given by the following equation where $S_{latency}$ defines a sorted array of the latency overhead for all individual traces in the dataset and N defines the total number of traces in the dataset.

$$L_{median}(D) = S_{latency}\left[\left\lfloor \frac{N}{2} \right\rfloor\right] \tag{7}$$

4.2.4. Accuracy

The accuracy of an attack defines the percentage of traces correctly identified within a dataset. The purpose of using accuracy as a metric of evaluation is twofold: (1) it indicates how well an attack performs despite the use of a defense; and (2) it demonstrates the ability of a defense to evade an attack and provide protection to the user—i.e., a lower accuracy means a better defense. Furthermore, the use of accuracy in the closed-world setting is consistent with related work in website fingerprinting [3–6,8–10,13–18].

The accuracy A of an attack is given by the following equation where T_c defines the number of traces correctly identified and N defines the total number of traces in the dataset.

$$A = \frac{T_c}{N} \tag{8}$$

4.2.5. True positive rate

In the context of website fingerprinting, the TPR describes the percentage of correctly labeled monitored traces. The TPR is commonly used for evaluation in the open-world setting [3,4,6–9,12,13,16,17,21,22]. Just as with accuracy in the closed-world setting, the TPR in the open-world setting measures the effectiveness of a website fingerprinting defense at providing a user with protection—a lower TPR means a better defense.

The TPR of an attack is given by the following equation where TP defines the number of correctly labeled monitored traces and FN defines the number traces that have been incorrectly identified as unmonitored.

$$TPR = \frac{TP}{TP + FN} \tag{9}$$

5. Beta randomized obfuscation

Recent website fingerprinting defenses have sought to decrease the latency and bandwidth overhead required by previous defenses. The goal is to provide the best protection without burdening the network with unnecessary traffic, as well as avoid diminishing the experience of the user, who may perceive delays in browsing as an unreliable connection. Furthermore, the defense should be simple (i.e., minimize computational overhead) and practical to implement. FRONT [12] is a recently proposed zero-delay, lightweight website fingerprinting defense. The authors of FRONT base their defense on the fact that the beginning of a trace — the first few seconds worth of a website load — leaks important information as to which website the trace belongs to. In addition, some attacks [4,8] explicitly use the front of a trace as an independent feature for classification, which points to its importance as a feature. To obscure these important features, FRONT allocates the majority of its padding budget to the beginning of the trace using a Rayleigh distribution and a randomized padding window.

In the case of [4], statistics of the front of the trace for incoming and outgoing packets are ranked as the 19th and 20th, respectively, out of 150 tested features, which emphasizes their importance to classification. Despite this, other high-ranking features, such as a variety of packet ordering statistics, may not be as easily obfuscated by FRONT due to the shape of a Rayleigh distribution—that is, as the window size used by FRONT increases, the spread of the distribution for sampling dummy packets also increases. This spreads the padding budget thin throughout a larger portion of the trace. Furthermore, many of the machine learning attacks with manual feature selection are dated, and newer attacks [6,7,9] trend toward the use of deep learning, such as Deep Fingerprinting, which has automatic feature extraction. With automatic feature extraction, it is not clear which features are the highest ranked for classification. The authors of [9] stress that the continued use of manual feature selection in machine learning attacks will lead to what they call an arms race between attacks and defenses where attacks gain the upper hand by including previously unconsidered features, and then new defenses rush to conceal the new features, only for attacks to subsequently exploit newer features yet again. Thus, a defense must be able to conceal several important features simultaneously and take into account changes in attack methodology such as the use of deep learning.

Considering all these factors, this paper proposes a new defense, titled Beta Randomized Obfuscation (BRO). The constraints of the FRONT defense are maintained—i.e., the defense must introduce no delay to real traffic and be lightweight. The defense also must maintain that the beginning of the trace will remain the most likely portion of the trace to receive padding due to its importance as a feature for classification. Furthermore, this defense seeks to diminish the accuracy of deep learning attacks, which continue to be some of the most effective attacks.

The novelty of BRO is summarized by three key points. Firstly, BRO uses a randomized beta distribution for sampling dummy packets timestamps. This ensures that the distribution for sampling timestamps does not severely increase its spread as the defense's padding window size increases. Secondly, this work makes the important observation that clustering dummy packets closer together within a defended trace provides better protection compared to other website fingerprinting defenses where the spread of dummy packets is sparser. Finally, experimental results show that the use of a beta distribution increases the degree of trace-to-trace randomness regarding the location of dummy packets within a defended trace over previous website fingerprinting defenses.

Table 1
BRO Defense Parameters and Variables.

	Notation	Description
Parameters	W_{min}	Minimum padding window
	M_{max}	Maximum padding window
	α_{min}	Minimum α parameter for beta distribution
	α_{max}	Maximum α parameter for beta distribution
	β_{min}	Minimum β parameter for beta distribution
	β_{max}	Maximum β parameter for beta distribution
	P_{min-c}	Client's minimum padding budget
	P_{max-c}	Client's maximum padding budget
	P_{min-s}	Proxy's minimum padding budget
	P_{max-s}	Proxy's maximum padding budget
Variables	$w_c = U(W_{min}, W_{max})$	Client's padding window
	$w_s = U(W_{min}, W_{max})$	Proxy's padding window
	$\alpha_c = U(\alpha_{min}, \alpha_{max})$	Client's alpha parameter
	$\beta_c = U(\beta_{min}, \beta_{max})$	Client's beta parameter
	$\alpha_s = U(\alpha_{min}, \alpha_{max})$	Proxy's alpha parameter
	$\beta_s = U(\beta_{min}, \beta_{max})$	Proxy's beta parameter
	$p_c = \tilde{U}(P_{min-c}, P_{max-c})$	Client's dummy packet number
	$p_s = \tilde{U}(P_{min-s}, P_{max-s})$	Proxy's dummy packet number

The defense maintains that the start of the padding distribution is always at time 0 — meaning that the sampled dummy packet timestamps are not deliberately offset — to conceal the beginning of the trace. The defense parameters and variables are shown in Table 1; U denotes a uniform distribution and $\tilde{U}$ denotes a discrete uniform distribution where both are bounded by the terms in parentheses.

5.1. BRO defense design

In BRO, padding timestamps are sampled from a randomized beta distribution to ensure that as the size of the padding window increases, the padding budget is not spread thin throughout the trace while also maintaining randomness. To provide a fair comparison with other defenses, BRO is implemented as a defense mechanism in simulations in which padding is applied to a previously collected trace. This is typical of other defenses [10,12,13,15,17]. Note, however, that much like other defenses, BRO is practical to implement in real-time and use on live traffic—the requirement is to generate a timetable, in exactly the same manner as the simulated defense, for each website load and send dummy packets when specified. When the client loads a website, it first generates a new timetable, and then sends a special control packet to the proxy indicating that a new website is being loaded and that it needs to generate its own timetable. For each dummy packet in the timetable, a timer is started, and when a timer expires, an interrupt is triggered causing a dummy packet to be sent. When the website has finished loading — i.e., the client has received all website artifacts — the client sends a second control packet indicating to the proxy to cancel any timers for outstanding dummy packets. If all dummy packets are sent and the website has not yet finished loading, traffic proceeds uninterrupted.

BRO has two steps, followed by both the client and proxy independently: sampling a number of dummy packets and scheduling dummy packets. Below explains the process for each step.

5.2. Sampling a number of dummy packets

The client and proxy first sample a number of dummy packets from a discrete uniform distribution bounded by 1 and a defined maximum, P_{max-c} and P_{max-s} for the client and proxy respectively. For each trace, the client samples p_c from $\bar{U}(P_{min-c}, P_{max-c})$, and the proxy samples p_s from $\bar{U}(P_{min-s}, P_{max-s})$. By varying the amount of padding added to each trace from both the client and the proxy independently and using a lower bound of $P_{min-c} = P_{min-s} = 1$, the randomness from trace-to-trace for the same website is greatly enhanced. The process of choosing P_{max} is explained below in Section 5.4.

5.3. Scheduling dummy packets

The next step is to schedule the dummy packets. Scheduling dummy packets has three sub-steps: sample a padding window, sample beta distribution parameters, and sample dummy packet timestamps. The padding window is used to scale the beta distribution; x-values of a beta distribution are always bounded by 0 and 1—therefore, the distribution is multiplied by the padding window size to randomize its scale. Finally, the client and proxy then sample timestamps for dummy packets from the appropriate beta distribution.

5.3.1. Sampling a padding window

The client and proxy each sample a padding window. The padding window determines the scale of the beta distribution and therefore the location of the dummy packets applied to the trace. In this experiment, a maximum window size of 14 s is used to provide a fair comparison with FRONT. Therefore, following the naming conventions used in FRONT, the client's padding window, w_c, and the proxy's padding window, w_s, are sampled from $U(W_{min}, W_{max})$.

5.3.2. Sampling beta distribution parameters

The client and proxy sample α and β parameters for a beta distribution from a uniform distribution. This is done independently and on a per-trace basis; that is, for each trace, the client and proxy obtain new beta distribution parameters. The α parameter is chosen from $U(\alpha_{min}, \alpha_{max})$, and the β parameter from $U(\beta_{min}, \beta_{max})$. Section 5.4.1 below describes the process for selecting α_{min}, α_{max}, β_{min}, and β_{max}. By choosing a beta distribution with randomized parameters, the peak of the probability density function can be shifted, improving upon the randomness of padding. The probability density function for a beta distribution is defined by the equation below where α corresponds to α_c and α_s, β corresponds to β_c and β_s, and t is time.

$$f(t, \alpha, \beta) = \frac{1}{B(\alpha, \beta)} t^{\alpha-1}(1-t)^{\beta-1} \tag{10}$$

Fig. 1 shows the probability density function for different alpha and beta parameters. As α becomes larger than β, the peak of the distribution shifts to the right and vice versa. The varying skew of the beta distribution enables dummy packets to be scheduled into different parts of a defended trace while remaining clustered together—this point is discussed further in Section 5.6.

5.3.3. Sampling dummy packet timestamps

The client's beta distribution is scaled by w_c, and the proxy's beta distribution is scaled by w_s. From there, the client samples p_c dummy packets from its beta distribution, and the proxy samples p_s dummy packets from its beta distribution. The two resulting sets are the timestamps for dummy packets where each timestamp corresponds to a single dummy packet. Dummy packets are injected into the trace at the corresponding timestamps and in the appropriate direction. Dummy packets with timestamps later than the last real packet in the trace are not sent.

Table 2
Beta Distribution Testing Parameters. All configurations use the following values for the remaining defense variables: $P_{min-c} = P_{min-s} = 1$, $P_{max-c} = P_{max-s} = 2000$, $W_{min} = 1$, and $W_{max} = 14$.

Configuration	Parameter settings
BTB1	$\alpha_{min} = 1$ $\alpha_{max} = 2$ $\beta_{min} = 9$ $\beta_{max} = 10$
BTB2	$\alpha_{min} = 1$ $\alpha_{max} = 4$ $\beta_{min} = 7$ $\beta_{max} = 10$
BTB3	$\alpha_{min} = 1$ $\alpha_{max} = 6$ $\beta_{min} = 5$ $\beta_{max} = 10$
BTB4	$\alpha_{min} = 1$ $\alpha_{max} = 8$ $\beta_{min} = 3$ $\beta_{max} = 10$
BTB5	$\alpha_{min} = 1$ $\alpha_{max} = 10$ $\beta_{min} = 1$ $\beta_{max} = 10$

5.4. Parameter selection

The following sections describe the process for selecting defense parameters. First, beta distribution parameters are chosen. Then, two P_{max} values are selected to create one lightweight defense configuration and a second configuration with slightly higher bandwidth overhead.

5.4.1. Choosing beta distribution parameters

The range of possible beta distribution parameters, α and β, should be optimized to create the best defense. This section demonstrates the necessity of fully randomizing the shape of the beta distribution. To test this, five configurations are created that place constraints on the shape of the beta distribution used within BRO. The beta distribution parameters for each configuration, BTB1-BTB5, are shown in Table 2. The other defense variables remain constant. A P_{max} value for both the client and proxy of 2000 is chosen for each configuration to give a medium bandwidth overhead. Similarly, the value of W_{max} is held constant across all configurations at the previously mentioned value of 14 s.

At each successive configuration, the amount of rightward skew permitted in the shape of the beta distribution is increased. This means that BTB1 is strictly limited to a leftward skew — i.e., it is only possible to sample dummy packet timestamps positioned at the beginning of the trace — whereas BTB5 implements a fully randomized beta distribution that can have a shape skewed to both the extreme left and right. The reasoning for beginning with a strictly leftward skew and gradually increasing the amount of rightward skew is twofold: (1) given the knowledge that the beginning of the trace contains sensitive features, this will ensure that the amount of rightward skew is not excessive — that is, this ensures that the rightward skew permitted is only that which is beneficial to the defense; and (2) the amount of rightward skew permitted and the degree of trace-to-trace randomness increase simultaneously — therefore, this experiment will examine the importance of an increased degree of trace-to-trace randomness.

Each configuration BTB1-BTB5 is tested against the three attacks chosen for this research — Deep Fingerprinting (DF) [6], k-Fingerprinting (k-FP) [4], and CUMUL [3] — to find the most effective configuration. The results are shown in Fig. 2. Against all attacks, BTB5 performs the best, emphasizing the importance of fully randomizing the shape of the beta distribution. Furthermore, this reinforces the importance of a high degree of trace-to-trace randomness, which is

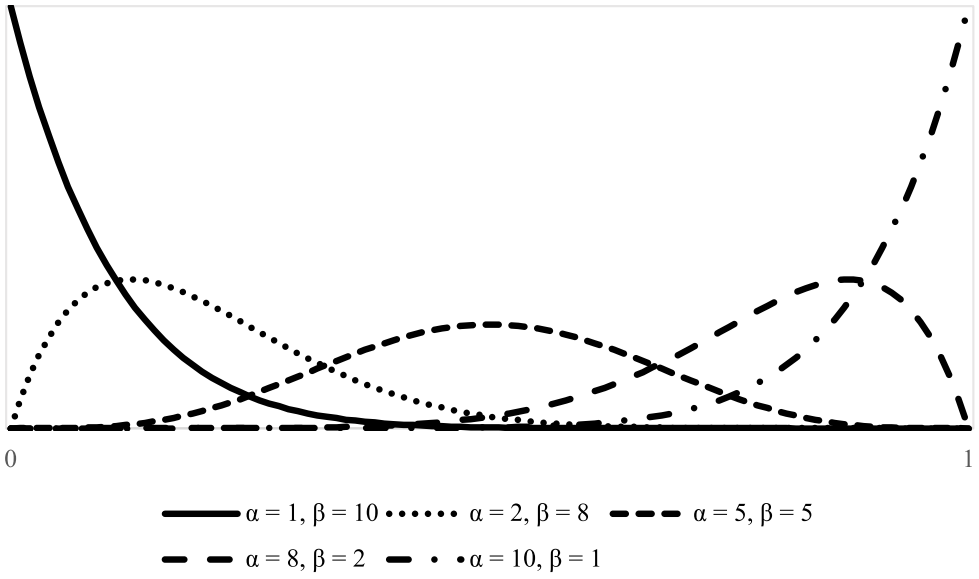

Fig. 1. Beta Distribution Shape. Note that as α becomes greater than β, the peak of the distribution shifts to the right and vice versa.

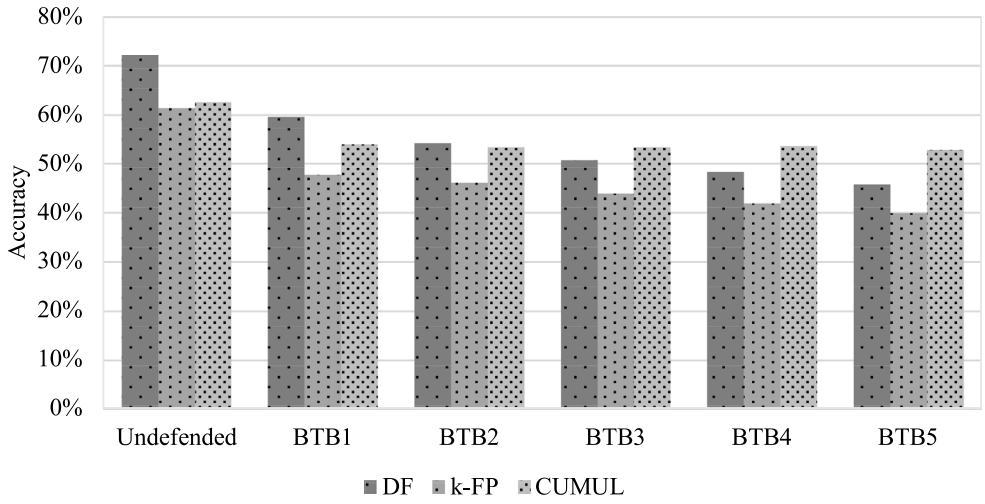

Fig. 2. Evaluation of Beta Distribution Parameters. As the amount of rightward skew permitted in the beta distribution's shape increases, the accuracy of all attacks decreases; thus, BTB5 provides the best protection.

explored further in Section 5.6.2. Note the trend among DF and k-FP that the attack accuracy continues to decrease each time the amount of rightward skew permitted increases; this reinforces the intuition above that solely targeting the beginning of a trace leaves other important features undefended. Also of note is the sensitivity of DF to an increased rightward skew, particularly in BTB4 and BTB5. Between BTB1 and BTB5, DF experiences a drop in accuracy from 59.6% to 45.8%. With these results, the beta distribution parameters of BTB5 are chosen as the final defense settings of BRO.

5.4.2. Choosing a padding budget

The next step in designing the defense is to choose a padding budget—i.e., the maximum number of dummy packets that will be present on the network as part of the defense. It is important to optimize this value; it must provide the most protection while introducing the smallest amount of bandwidth overhead. To test this, BRO is simulated using the parameters of BTB5 and varying the value of P_{max} at increments of 250 starting at 250 through 3000. Note that here P_{max} refers to the value of both P_{max-c} and P_{max-s}. Fig. 3 shows the relationship between P_{max} and attack accuracy.

Observe that the accuracy of all attacks declines as P_{max} increases. The protection provided by BRO is particularly apparent against DF, whose accuracy is decreased below both CUMUL and k-FP despite being the strongest attack against undefended traffic. The results also indicate that even a relatively small padding budget is effective at defending against attacks; all attacks experience at least a 5% drop in accuracy with a P_{max} value of just 250 (2.8% median bandwidth overhead). Two values of P_{max} are chosen for two different BRO configurations: one resulting in a lightweight configuration with moderate protection, and the other a configuration with slightly increased bandwidth overhead but excellent protection, similar to FT-1 and FT-2 in [12].

DF and CUMUL are used to tune the value of P_{max} since they are the two best-performing attacks in this experiment, outperforming k-FP. The accuracy of DF is reduced below 50% with a P_{max} value of 1750; however, it is important to keep each configuration's bandwidth overhead similar to FRONT for a fair comparison. Therefore, for the first configuration, BRO-1, a P_{max} value of 1500 is chosen, which

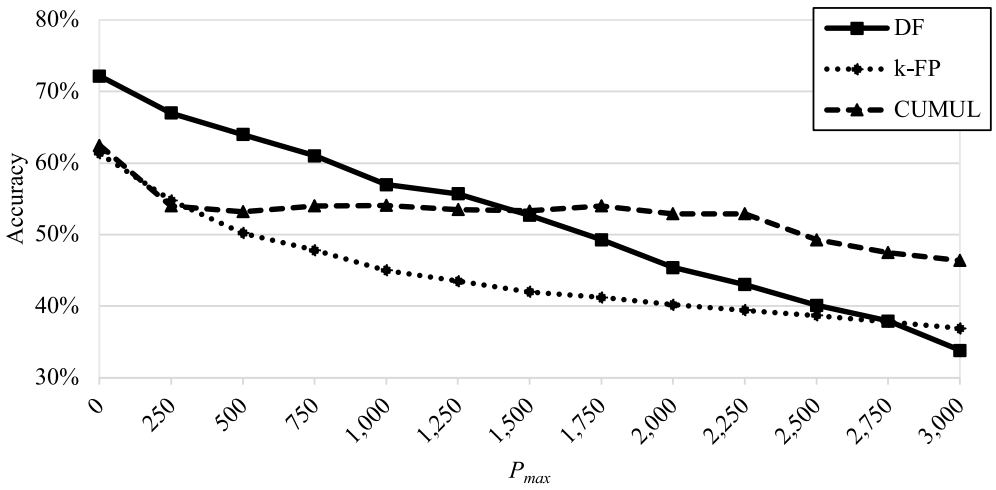

Fig. 3. P_{max} vs. Attack Accuracy. BRO reduces the accuracy of DF from 72.2% on undefended traffic to 33.8% with a padding budget of 3000.

yields a median bandwidth overhead of 16.8% vs. 16.1% for FT-1. The accuracy of CUMUL holds steady between 52% and 54% until P_{max} is increased to 2500, where it dips below 50%; however, the accuracy of DF continues to decline somewhat linearly as the padding budget increases. Though a higher value of P_{max} could be selected, a value of 2250 is chosen for the second configuration, BRO-2, to limit the bandwidth overhead; this gives a median bandwidth overhead of 25.4% vs. 23.6% for FT-2. Hence, for the final configurations of BRO-1 and BRO-2, P_{max} values of 1500 and 2250 are chosen. Note that the complete defense settings for BRO-1, BRO-2, FT-1, and FT-2 are available in Table 3.

5.5. Computational overhead of BRO

Note that the computational overhead of implementing BRO is minimal due to its comparatively small bandwidth overhead and the fact that it does not delay real traffic. The required overhead is dominated by the sampling of dummy packet timestamps—i.e., as the padding budget increases, so too does the computational overhead simply because there are more dummy packet timestamps to sample.

5.6. Analyzing why BRO outperforms FRONT

In the sections below, two characteristics of the padding applied by BRO and FRONT are explored. First is the spread of dummy packets—here, the spread is used to measure the clustering of timestamps for dummy packets applied by the defense. Measuring the spread of dummy packets will test the intuition that FRONT spreads its padding budget across a wider portion of the trace than BRO, and therefore may struggle to conceal trace features efficiently. For the dummy packet spread analysis, the timestamps applied to the trace by the client and proxy are considered independent—i.e., when gathering the timestamps of a defended trace, the padding applied by the client and proxy are collected as separate distributions.

The second characteristic is the location of the dummy packets within a defended trace. The location specifies where within a trace (the distance in seconds from the beginning) dummy packets are positioned. Determining the location of dummy packets in a defended trace will verify whether BRO applies padding deeper into the trace, which is crucial for obscuring features outside of the beginning of the trace. To ensure accurate results, BRO and FRONT are simulated 10 times and the statistics calculated in Sections 5.6.1 and 5.6.2 are averaged across all iterations.

Table 3
Defense Settings.

Defense	Settings
BRO-1	$W_{min} = 1$ s, $W_{max} = 14$ s $\alpha_{min} = \beta_{min} = 1, \alpha_{max} = \beta_{max} = 10$ $P_{max-c} = P_{max-s} = 1500$
BRO-2	$W_{min} = 1$ s, $W_{max} = 14$ s $\alpha_{min} = \beta_{min} = 1, \alpha_{max} = \beta_{max} = 10$ $P_{max-c} = P_{max-s} = 2250$
FT-1 [12]	$N_c = N_s = 1700$ $W_{min} = 1$ s, $W_{max} = 14$ s
FT-2 [12]	$N_c = N_s = 2500$ $W_{min} = 1$ s, $W_{max} = 14$ s
TT-2 [15]	Longest trace as reference sequence
TT-6 [15]	Longest trace as reference sequence
WT [17]	Longest trace as reference sequence

5.6.1. Dummy packet spread

The first characteristic is the spread of dummy packet timestamps. Using prior intuition on the shape of beta and Rayleigh distributions, it is hypothesized that the dummy packet timestamps throughout a *single trace* defended by FRONT will on average have a wider spread than that of a trace defended by BRO—i.e., BRO will deliver a clustered burst of dummy packets into the trace. This does not mean that FRONT provides padding deeper (further from the beginning) into the trace than BRO, but rather that to achieve coverage further from the beginning of the trace, FRONT must spread its padding budget across a much wider portion of the trace. In contrast, the beta distribution used in BRO provides coverage outside of the beginning of a trace while keeping the dummy packets clustered and targeted to a smaller portion of the trace. This is because the shape of the Rayleigh distribution employed by FRONT is directly tied to the size of the padding window; as the padding window grows in size, the spread of the Rayleigh distribution increases. This means that for a wider padding window in FRONT, dummy packets will be sparser and therefore less disruptive to trace features. In contrast, the beta distribution used by BRO is only *scaled* by the padding window—the distribution keeps the same shape, which clusters the dummy packets closer together. Furthermore, to be disruptive to the features of a trace further from the beginning, BRO only needs to generate beta distribution parameters that skew the distribution further to the right, which does not spread its padding budget thin.

To test this, the distribution of dummy packet timestamps applied to each trace is collected and the mean, median, and standard deviation of the interquartile range (IQR) are computed. The interquartile range is

Table 4
Dummy Packet Spread. BRO applies dummy packets into a more finite portion of the trace.

Defense	Median IQR	Mean IQR	Std. Dev.
BRO-1	1.4	1.5	0.8
BRO-2	1.4	1.5	0.8
FT-1	5.9	5.1	1.8
FT-2	6.0	5.1	1.8

Table 5
Dummy Packet Location. The median timestamp between BRO and FRONT is similar; however, the mean timestamp indicates that BRO simultaneously applies padding deeper into the trace. The coefficient of variation proves that BRO has a higher degree of trace-to-trace randomness.

Defense	Median timestamp	Mean timestamp	Coefficient of variation
BRO-1	0.66	0.84	101.2%
BRO-2	0.61	0.90	101.3%
FT-1	0.30	0.37	80.6%
FT-2	0.39	0.49	76.8%

selected over the entire range of timestamps because it will not contain outliers and thus provide a better metric that is not sensitive to extreme cases. Table 4 summarizes the results. Observe that BRO has an IQR of dummy packet timestamps smaller than that of FRONT. This means that on average, BRO uses half of its padding budget within a 1.5-second window, whereas FRONT uses half of its padding budget within a 5.1-second window. Again, note the importance that this is not the location of dummy packets within a trace, but rather how dispersed from each other the dummy packets are.

For further analysis, the maximum interquartile range of dummy packet timestamps observed across all simulated iterations for each defense is examined; BRO-1 and BRO-2 experience a maximum of 6.55 and 6.32 s respectively, and FT-1 and FT-2 experience a maximum of 13.11 and 14.32 s respectively. Hence, the maximum interquartile range of dummy packet timestamps applied by BRO is much smaller than that of FRONT—further emphasizing the clustering of dummy packets in BRO. This verifies the intuition above that the dummy packets applied by FRONT have a wider spread and are therefore sparser throughout the trace than BRO; meaning that the padding applied by BRO is more comparable to an injection of dummy packets into a more finite portion of the trace. In other words, BRO delivers a highly concentrated injection of dummy packets to a random and finite portion of the trace whereas FRONT's coverage provides a blanket of dummy packets dispersed over a wider portion of the trace, spreading its padding budget thin.

5.6.2. Dummy packet location

The differences in the distribution of dummy packets between BRO and FRONT are further analyzed by calculating the mean, median, and coefficient of variation of dummy packet timestamps. The median and mean timestamp will provide insight into the location of padding within a defended trace. The coefficient of variation for the dummy packet timestamps applied by each defense provides further insight into the level of trace-to-trace randomness with respect to the location of padding within a defended trace. The results are given in Table 5.

The median dummy packet timestamps of BRO-1 and BRO-2 indicate that the defense applies a good measure of padding to the beginning of a trace, with half of all dummy packets being sent before the 1-second mark. Furthermore, BRO-1 and BRO-2 have median and mean timestamps greater than that of FT-1 and FT-2; this implies that BRO applies coverage to the beginning of the trace while simultaneously injecting dummy packets deeper into the trace than FRONT. This proves that despite the beta distribution used in BRO's ability to skew to the extreme right, BRO does not struggle to apply padding to the beginning of a trace.

Next, the coefficient of variation for dummy packet timestamps is examined. BRO-1 and BRO-2 have a coefficient of variation of 101.2% and 101.3% respectively and FT-1 and FT-2 have a coefficient of variation of 80.6% and 76.8%. This implies that BRO has a higher degree of randomness than FRONT regarding the location of dummy packets, while still ensuring that the beginning of a trace receives padding. It does this without spreading its padding budget thin across a large portion of the trace as was seen previously. This is likely due to the parameters of the beta distribution which allow it to skew to both the extreme left and extreme right.

6. Experimental setup

In this experiment, a new dataset of traces is collected. Traces of regular, unmodified traffic as well as 3 implementations of half-duplex traffic corresponding to entirely half-duplex traffic for Walkie-Talkie [17], and two Tail Time [15] configurations (TT-2, and TT-6). Specifics on the dataset are described in Section 6.2 below. Using the collected traces paired with simulated defenses, several website fingerprinting attacks are carried out to test the effectiveness of each defense.

6.1. Half-duplex implementation

Half-duplex website loading is implemented using the webRequest API [23] from Firefox WebExtensions [24]. Using the webRequest API allows for each configuration to be loaded as a browser extension without requiring modifications to the Firefox source code. This ensures ease of use and long-term compatibility. There are 3 configurations used in this experiment that implement Walkie-Talkie [17], and the two best-performing Tail Time configurations against each attack in [15] with tail timeout values of 2 and 6 s.

6.2. Dataset

In this experiment, the closed-world scenario is used to evaluate defense effectiveness. The top 100 websites from the Alexa Top Sites list [25] are used as the initial dataset. As a first round of elimination, the methodology of [15] is followed and subdomains and localized domains are removed. In the second round of elimination, each website is loaded through Tor a single time, and each website that fails to load is removed from the dataset. After each elimination round, the dataset is replenished with websites ranking further down the list. The result is a set of 100 websites without duplicates and subdomains that are less likely to experience failed page loads. Each website is accessed 1000 times for a total of 100,000 traces.

6.3. Data collection

A controller synchronizes and carries out all subcomponents of trace capture. The components include a modified Tor proxy, a cell logger, a script that verifies connectivity, and a script that visits the desired website. The controller is a simple shell script that starts and stops each component when needed and keeps track of which instance of which website is being captured.

The Tor source code is modified to directly emit cell sequences at a layer that excludes SENDME cells. SENDME cells are used for control within Tor but provide no useful information for classification [26]. The information emitted includes the timestamp and cell direction. The modified version of Tor is compiled and run in a Docker container so that it can be started and stopped at each page load, completely resetting Tor and the existing circuits. To minimize additional overhead to the modified instance of Tor, the cells are emitted to a cell logger. The cell logger listens for information from Tor and logs it to a file for the corresponding trace. Cell information is emitted from Tor to the cell logger using UDP packets.

In this experiment, it is desired to have the fullest and most accurate traces. Without checking for connectivity before starting a trace, valuable capture time is wasted if the initial website request must first wait for connectivity. This would make such traces smaller than traces from the same website where connectivity was not a problem—therefore increasing variation between traces from the same website. An increased variation could make classification more difficult and not provide a fair experiment. Thus, before visiting a website, the controller verifies that it can make a connection through Tor. It issues a request to http://connectivitycheck.gstatic.com/generate_204 with a timeout value of 15 s. If a connection is unable to be made, rather than skipping to the next capture, the capture components are torn down and the capture is restarted until a connection can be made.

The controller visits each website using Firefox, which is controlled by Selenium [27] through GeckoDriver [28]. This is the point at which extensions (if any) are loaded into the Firefox profile. The Firefox profile is configured to use the modified instance of Tor as a proxy. Note that for each website access, a new Tor circuit is used—i.e., each time the controller starts a capture, a new Tor circuit is created. The controller terminates each website capture after 15 s first by stopping the Docker container and then by sending a special packet to the cell logger telling it to save the trace. A 15-second trace capture is adequate due to the use of a connectivity check before browsing a website. This is demonstrated by the trace length in each dataset—the 90th percentile of trace lengths in DS−14 [8] and DS−19 [12] are 4947 and 9862 packets, respectively. The 90th percentile of traces in the dataset collected in this work is 18238 packets.

To further ensure the quality of traces, the 1000 captures for each website are split into 40 rounds. That is, for each of the 100 websites, there are 25 successive captures before moving on to the next website. This is done 40 times for a total of 1000 traces per website. This is to decrease the likelihood of being temporarily IP banned by the website's server due to repeated visits, hence providing quality traces.

The dataset is collected using 8 machines with a wired connection to the university network. Two machines are paired, and each pair collects traces for a different configuration. Among a pair of machines, each machine collects 500 traces per website. One pair collects unmodified, full-duplex traffic and the other three pairs use one of the previously mentioned extensions implementing half-duplex, burst traffic for Walkie-Talkie, TT-2, and TT-6. Furthermore, trace collection is stagger-started with 2 h between each machine to guarantee they are not all accessing the same website at once. Note that there are no ethical concerns with this experiment because each website visit is done with an automated script and there are no real users involved in the data collection.

6.4. Attacking experiment

In order to evaluate each defense, the best-performing attacks from [12] are chosen; they are Deep Fingerprinting[1] [6], k-Fingerprinting[2,3] [4], and CUMUL[4,5] [3]. This enables BRO to be evaluated against both deep learning and machine learning attacks, verifying that it is effective at defending against both. For each attack, the default settings are used and k-fold cross-validation is done as specified in their original work [3,4,6] Note that this experiment assumes the attacker knows which defense is being used, and therefore trains their classifier using defended traces—i.e., when simulating an attack on a defense, the defended dataset is split into a training and testing set. Defenses are evaluated in a closed-world scenario.

[1] Source code for the DF attack used in this work: https://github.com/deep-fingerprinting/df.

[2] Source code for the k-FP attack used in the closed-world scenario in this work: https://github.com/jhayes14/k-FP.

[3] Source code for the k-FP attack used in the open-world scenario in this work: https://github.com/websitefingerprinting/WebsiteFingerprinting.

[4] Source code for the CUMUL attack used in the closed-world scenario in this work: https://www.cs.sfu.ca/~taowang/wf/attacks/.

[5] Source code for the CUMUL attack used in the open-world scenario in this work: https://github.com/websitefingerprinting/WebsiteFingerprinting.

Table 6
Bandwidth Overhead. TT and WT cause a high bandwidth overhead due to the choice of the longest trace as the reference sequence.

Defense	Median bandwidth overhead	Mean bandwidth overhead
BRO-1	16.8%	24.2%
BRO-2	25.4%	36.7%
FT-1	16.1%	23.6%
FT-2	23.6%	34.5%
TT-2	607.4%	855.2%
TT-6	592.9%	835.8%
WT	570.3%	805.6%

7. Defense evaluation

To provide a comparison, three defenses, FRONT [12], Tail Time (TT) [15], and Walkie-Talkie (WT) [17] are selected. FRONT is another lightweight defense and Walkie-Talkie represents a defense that introduces delay to the trace. Tail Time is selected because it is an improved version of Walkie-Talkie that decreases page loading time. For both Tail Time and Walkie-Talkie, the padding method of [15] is followed where the longest trace is selected as the reference sequence. BRO is simulated in two configurations (BRO-1 and BRO-2) using the parameters detailed in Section 5, and FRONT is simulated with the parameters of FT-1 and FT-2 as outlined in [12]. Table 3 summarizes the configuration for each defense.

7.1. Bandwidth overhead

BRO aims to provide an improved defense while maintaining comparatively low bandwidth overhead. BRO's bandwidth overhead is similar to FRONT's (see Table 6). The slightly higher bandwidth overhead of BRO may be explained by the increased degree of dummy packet clustering relative to FRONT; both defenses cancel all dummy packets scheduled to be sent after the last real packet, and the wider spread of dummy packets in FRONT could allow more dummy packets to be canceled, resulting in a 1%–2% higher bandwidth overhead in BRO. However, it is important to note that BRO-1 and BRO-2 have smaller padding budgets (1500 and 2250) than FT-1 and FT-2 (1700 and 2500), respectively. This means that if either defense were to utilize the entire padding budget on a single trace, BRO would introduce fewer dummy packets to the network, lessening the burden on the network over FRONT.

Observe that in all cases, the median bandwidth overhead is much lower than the mean bandwidth overhead—indicating positive skew. This suggests that several smaller traces within the dataset pull the mean bandwidth overhead higher than the median.

7.2. Latency overhead

Recall that dummy packets sent before the last real packet do not contribute to latency overhead. This means that BRO and FRONT have 0% latency overhead. The mean and median latency overhead for each defense is shown in Table 7. Notice that Tail Time and Walkie-Talkie incur high latency overheads due to their use of half-duplex communication.

Note that TT experiences a higher latency overhead than WT. It is believed that this does not contradict TT's goal of decreasing page load times over WT and rather that the half-duplex bursts in TT are smaller and more irregular than WT, leading to a backlog of bursts from the reference sequence that still must be sent despite all real packets already being received. Since this is not relevant to the evaluation of BRO, it is not further explored and is left for future work.

Table 7
Latency Overhead. BRO and FRONT incur no latency overhead because they do not delay real traffic.

Defense	Median latency overhead	Mean latency overhead
BRO-1	0.0%	0.0%
BRO-2	0.0%	0.0%
FT-1	0.0%	0.0%
FT-2	0.0%	0.0%
TT-2	53.1%	52.4%
TT-6	59.7%	59.2%
WT	48.5%	48.6%

Table 8
Attack Accuracy (Note that a lower accuracy means a better defense.).

Defense	Accuracy		
	DF	k-FP	CUMUL
Undefended	72.2%	61.3%	62.5%
BRO-1	52.7%	42.0%	53.3%
BRO-2	43.0%	39.5%	51.9%
FT-1	58.8%	42.6%	57.2%
FT-2	54.2%	38.9%	51.1%
TT-2	60.0%	55.3%	27.7%
TT-6	57.9%	47.2%	27.2%
WT	53.7%	44.9%	25.1%

7.3. Evaluation against attacks

Each defense is evaluated against the three attacks chosen above: DF, k-FP, and CUMUL. Table 8 summarizes the accuracy of the attacks on each defense. BRO performs nearly equal to FRONT against the machine learning attacks (k-FP and CUMUL), suggesting that BRO is not any more susceptible to these attacks. However, BRO-1 and BRO-2 outperform FT-1 and FT-2 against DF with comparable bandwidth overhead, making BRO the superior defense. BRO-1 and BRO-2 reduce the accuracy of DF down to 52.7% and 43.0% respectively compared to 58.8% and 54.2% for FT-1 and FT-2. BRO and FRONT outperform TT and WT against DF and k-FP, with WT providing the best protection against CUMUL. Note, however, that the extra protection by WT in this case comes at the cost of an increased bandwidth and latency overhead.

A decrease in accuracy for all attacks against undefended traffic is observed compared to their original research. In line with [15], it is hypothesized that this is due to the trace collection method used in this experiment, which provides more accurate end-to-end trace data by logging cell sequences directly from Tor as opposed to reconstructing them from a tcpdump. Additionally, the trace collection method utilizes UDP packets, which may allow for some packet loss, though this loss is expected to be small.

7.4. The extent to which BRO outperforms FRONT

It is essential to understand the extent to which BRO outperforms FRONT and how much FRONT must increase its padding budget to perform on par with BRO. To examine this, BRO and FRONT are evaluated against DF by increasing the padding budget at increments of 250 packets starting at 250 through 3000, similar to the experiment above which tuned the padding budget of BRO. Here, the padding budget refers to the value given to both the client and proxy. DF is chosen as the only attack due to its reputation as being accurate even against modest defenses [6,12], which makes it ideal to demonstrate how effectively BRO and FRONT utilize their padding budgets. The results are shown in Fig. 4; note that a lower accuracy means a better defense. Observe that BRO outperforms FRONT immediately with a padding budget as low as 250 packets. BRO reduces the accuracy of DF to below 50% with a padding budget of just 1750 packets. At a padding budget of 3000 — nearly double the requirement of BRO — FRONT is not able to bring the accuracy of DF below 50%. At the most effective configuration with a padding budget of 3000 dummy packets, the accuracy of DF on BRO is reduced to 33.8% whereas FRONT reduces the accuracy to just 51.1%. Furthermore, with a padding budget equal to or greater than 1000, DF experiences on average 10.9% lower accuracy on BRO than it does on FRONT, and the gap in the protection provided by each defense continues to widen as the padding budget increases. This suggests that DF is more sensitive to the highly concentrated and randomized injection of dummy packets administered by BRO.

Table 9
BRO vs. FRONT on DS−19 (Note that TPR is more appropriate than accuracy for an open-world dataset as discussed in Section 4, and that a lower TPR means a better defense. Note also that a larger padding window size increases the effectiveness of BRO.).

Defense	TPR		
	DF	k-FP	CUMUL
BRO-1 (14 s)	57.2%	62.0%	34.2%
BRO-2 (14 s)	46.5%	57.8%	26.3%
BRO-1 (30 s)	58.4%	46.1%	27.0%
BRO-2 (30 s)	38.6%	39.0%	19.4%
FT-1	76.4%	48.0%	40.5%
FT-2	63.0%	35.6%	25.8%

7.5. BRO vs. FRONT on DS−19

A comparison of BRO and FRONT was done on DS−19, an open-world dataset outlined in [12]. The reason for this comparison is twofold: (1) it tests the performance of BRO in an open-world setting; and (2) it enhances the comparison of the performance of BRO and FRONT by analyzing both defenses on the dataset FRONT was developed on. For each attack, the TPR is calculated, which is more appropriate for an open-world dataset as discussed in Section 4. The results are given in Table 9.

Observe that the same overall trends from Section 7.3 are maintained — i.e., that BRO outperforms FRONT, particularly against DF — except for that BRO performs worse than FRONT against k-FP. The last packet in traces from DS−19 has an average timestamp of 27.3 s, and the longest trace (by order of time) has a final timestamp of 93 s. By contrast, the dataset collected in this work has an average final packet timestamp of 17.3 s and a maximum of 20.2 s. Somewhat arbitrarily, a new padding window size of 30 s is selected for BRO (denoted as BRO-1 (30 s) and BRO-2 (30 s) in Table 9) to encompass a larger part of the trace in DS−19. Note that increasing the padding window size decreases the bandwidth overhead due to the increase in potential for dummy packets to be scheduled beyond the last real packet in a trace, and therefore canceled. Increasing the padding window to 30 s decreases the performance of k-FP such that BRO-1 outperforms FT-1 but FT-2 is marginally better than BRO-2 against k-FP. Further increasing the padding window size to 45 s decreases the TPR of k-FP to 41.4% against BRO-1 and 32.4% against BRO-2. In this work, a padding window size of 14 s was chosen for a comparison with FRONT, and these results reiterate the need to fine-tune the padding window size in BRO to create an optimal defense. This is left as future work.

8. Conclusion and future work

8.1. Conclusion

This work presents a new lightweight website fingerprinting defense called Beta Randomized Obfuscation (BRO) that introduces no delay to real browsing traffic. BRO utilizes a randomized beta distribution and padding window for sampling dummy packet timestamps. The padding distribution always starts at time 0, but the beta distribution's ability to skew to the extreme right allows for easy obfuscation of important features further from the beginning of the trace.

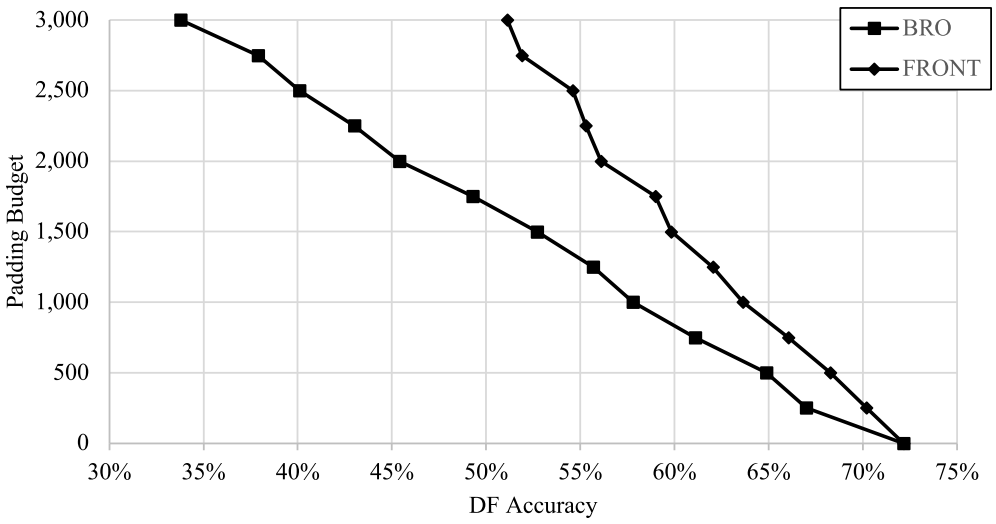

Fig. 4. Deep Fingerprinting Accuracy vs. Padding Budget. At all padding budget increments, BRO provides better protection than FRONT against DF. In order to restrict the accuracy of DF below 50%, FRONT requires a padding budget of over 3000 whereas BRO only requires 1750.

The results are verified by the experimental setup used which is consistent with related work. Furthermore, the dataset size mirrors that of the original DF work, and the evaluation techniques used to compare BRO to other recent website fingerprinting defenses are relevant, reinforcing the authority of the results.

BRO is demonstrated to be effective at concealing website traffic against several state-of-the-art website fingerprinting attacks. Analysis of dummy packet timestamps shows that BRO delivers a more concentrated injection of padding to a finite section of the trace than does FRONT, another lightweight website fingerprinting defense. Furthermore, BRO has a higher rate of trace-to-trace randomness than FRONT regarding the location of dummy packets within a trace. This means that BRO more efficiently utilizes its padding budget against attacks as demonstrated by the experimental results.

Against attacks, BRO proves to diminish the accuracy of Deep Fingerprinting to the point that it performs worse than the other attacks used in this experiment despite being more accurate on undefended traffic. BRO provides better protection against Deep Fingerprinting than other state-of-the-art defenses.

Analysis of how increasing the padding budget of BRO and FRONT affects the accuracy of Deep Fingerprinting shows that FRONT must increase its padding budget (therefore increasing the bandwidth overhead) over that of BRO to achieve the same level of protection; in some cases, FRONT must nearly double its padding budget. In fact, with a padding budget of 1000 or greater, Deep Fingerprinting is on average 10.9% less accurate against BRO than it is against FRONT.

8.2. Future work

This work presents five configurations for experimenting with beta distribution parameters. Future work may include a better fine-tuning of these parameters to find the best defense. This includes testing to see if the range of possible parameters should differ between the client and proxy. Experimentation with the maximum padding window size is also warranted. This work uses a maximum padding window size of 14 s for comparison with FRONT; a more optimal value may exist.

Another area of future work is testing BRO on the QUIC protocol. This would involve creating a QUIC-compatible implementation of BRO to analyze both its implementation as a client-side-only defense and the protection it provides on the QUIC protocol as was done with FRONT in [22].

This work is limited in that only a single deep learning attack is simulated against BRO. To gain a better sense of BRO's capabilities and the generalizability of its performance against deep learning attacks, it is necessary to experiment with others such as [7,9].

Availability

The source code for this experiment is made available to the research community to replicate results.

BRO

https://github.com/csmcguan/bro

Tail time and walkie-talkie

https://github.com/csmcguan/tail-time

Tor cell capture

https://github.com/csmcguan/tor-capture

CRediT authorship contribution statement

Colman McGuan: Writing – original draft, Validation, Software, Investigation, Formal analysis, Conceptualization. **Chansu Yu:** Writing – review & editing, Supervision, Funding acquisition, Formal analysis, Conceptualization. **Kyoungwon Suh:** Writing – review & editing, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Links to the code used in this work are provided in the manuscript.

References

[1] J.-P. Verkamp, M. Gupta, Inferring mechanics of web censorship around the world, in: Proceedings of the 2nd USENIX Workship on Free and Open Communications on the Internet, 2012.

[2] Z. Wang, Y. Cao, Z. Qian, C. Song, S.V. Krishnamurthy, Your state is not mine: a closer look at evading stateful internet censorship, in: Proceedings of the 2017 Internet Measurement Conference, 2017, pp. 114–127, http://dx.doi.org/10.1145/3131365.3131374.

[3] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, T. Engel, Website fingerprinting at internet scale, in: Proceedings of the 2016 Network and Distributed System Security (NDSS) Symposium, 2016, http://dx.doi.org/10.14722/ndss.2016.23477.

[4] J. Hayes, G. Danezis, k-fingerprinting: a robust scalable website fingerprinting technique, in: Proceedings of the 25th USENIX Security Symposium, 2016, pp. 1187–1203.

[5] M. Shen, Y. Liu, L. Zhu, X. Du, J. Hu, Fine-grained webpage fingerprinting using only packet length information of encrypted traffic, IEEE Trans. Inf. Forensics Secur. (2021) 2046–2059, http://dx.doi.org/10.1109/TIFS.2020.3046876.

[6] P. Sirinam, M. Imani, M. Juarez, M. Wright, Deep fingerprinting: undermining website fingerprinting defenses with deep learning, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1928–1943, http://dx.doi.org/10.1145/3243734.3243768.

[7] S.E. Oh, S. Sunkam, N. Hopper, p^1-fp: Extraction, classification, and prediction of website fingerprints with deep learning, in: Proceedings on Privacy Enhancing Technologies Symposium, 2019.

[8] T. Wang, X. Cai, R. Nithyanand, R. Johnson, I. Goldberg, Effective attacks and provable defenses for website fingerprinting, in: Proceedings of the 23rd USENIX Security Symposium, 2014, pp. 143–157.

[9] V. Rimmer, D. Preuveneers, M. Juarez, T.V. Goethem, W. Joosen, Automated website fingerprinting through deep learning, in: Proceedings of the 2018 Network and Distributed System Security (NDSS) Symposium, 2018, http://dx.doi.org/10.14722/ndss.2018.23105.

[10] K.P. Dyer, S.E. Coull, T. Ristenpart, T. 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, 2012, pp. 332–346, http://dx.doi.org/10.1109/SP.2012.28.

[11] P. Sirinam, N. Matthews, M.S. Rahman, M. Wright, Triplet fingerprinting: more practical and portable website fingerprinting with N-shot learning, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1131–1148, http://dx.doi.org/10.1145/3319535.3354217.

[12] J. Gong, T. Wang, Zero-delay lightweight defenses against website fingerprinting, in: Proceedings of the 29th USENIX Security Symposium, 2020, pp. 717–734.

[13] D. Lu, S. Bhat, A. Kwon, S. Devadas, DynaFlow: an efficient website fingerprinting defense based on dynamically-adjusting flows, in: Proceedings of the 2018 Workshop on Privacy in the Electronic Society, 2018, pp. 109–113, http://dx.doi.org/10.1145/3267323.3268960.

[14] W.D. la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engle, K. Wehrle, A. Panchenko, TrafficSliver: fighting website fingerprinting attacks with traffic splitting, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 1971–1985, http://dx.doi.org/10.1145/3372297.3423351.

[15] J. Liang, C. Yu, K. Suh, H. Han, Tail time defense against website fingerprinting attacks, IEEE Access (2022) 18516–18525, http://dx.doi.org/10.1109/ACCESS.2022.3146236.

[16] M. Juarez, M. Imani, M. Perry, C. Diaz, M. Wright, Toward an efficient website fingerprinting defense, in: Proceedings of the 21st European Symposium on Research in Computer Security, 2016, pp. 27–46.

[17] T. Wang, I. Goldberg, Walkie-talkie: an efficient defense against passive website fingerprinting attacks, in: Proceedings of the 26th USENIX Security Symposium, 2017, pp. 1375–1390.

[18] X. Cai, R. Nithyanand, T. Wang, R. Johnson, I. Goldberg, A systematic approach to developing and evaluating website fingerprinting defenses, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 227–238, http://dx.doi.org/10.1145/2660267.2660362.

[19] D. Wagner, B. Schneier, Analysis of the SSL 3.0 protocol, in: Proceedings of the Second USENIX Workshop on Electronic Commerce, 1996.

[20] H. Cheng, R. Avnur, Traffic analysis of SSL encrypted web browsing, 1998.

[21] J.-P. Smith, P. Mittal, A. Perrig, Website fingerprinting in the age of QUIC, in: Proceedings on Privacy Enhancing Technologies, 2021, pp. 1–22.

[22] J.-P. Smith, L. Dolfi, P. Mittal, A. Perrig, QCSD: A QUIC client-side website-fingerprinting defense framework, in: Proceedings of the 32st USENIX Security Symposium, 2022, pp. 771–789.

[23] MDN Web Docs Contributors, JavaScript APIs: webRequest, MDN web docs, 2022, URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest.

[24] MDN Web Docs Contributors, Browser extensions, MDN web docs, 2022, URL https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions.

[25] Alexa, URL http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[26] T. Wang, I. Goldberg, Improved website fingerprinting on tor, in: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, 2013, pp. 201–212, http://dx.doi.org/10.1145/2517840.2517851.

[27] Software Freedom Conservancy, Selenium, URL https://www.selenium.dev/.

[28] Mozilla Foundation, GeckoDriver, URL https://firefox-source-docs.mozilla.org/testing/geckodriver/index.html.