A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses

Xiang Cai [1] Rishab Nithyanand [1] Tao Wang [2] Rob Johnson [1] Ian Goldberg [2]

[1]Stony Brook University
{xcai, rnithyanand, rob}@cs.stonybrook.edu

[2]University of Waterloo
{t55wang, iang}@cs.uwaterloo.ca

ABSTRACT

Fingerprinting attacks have emerged as a serious threat against privacy mechanisms, such as SSL, Tor, and encrypting tunnels. Researchers have proposed numerous attacks and defenses, and the Tor project now includes both network- and browser-level defenses against these attacks, but published defenses have high overhead, poor security, or both.

This paper (1) systematically analyzes existing attacks and defenses to understand which traffic features convey the most information (and therefore are most important for defenses to hide), (2) proves lower bounds on the bandwidth costs of any defense that achieves a given level of security, (3) presents a mathematical framework for evaluating performance of fingerprinting attacks and defenses in the open-world, given their closed-world performance, and (4) presents a new defense, Tamaraw, that achieves a better security/bandwidth trade-off than any previously proposed defense.

Our feature-based analysis provides clear directions to defense designers on which features need to be hidden. Our lower bounds on bandwidth costs help us understand the limits of fingerprinting defenses and to determine how close we are to "success". Our open-world/close-world connection enables researchers to perform simpler closed-world experiments and predict open-world performance. Tamaraw provides an "existence proof" for efficient, secure defenses.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General–Security and protection
; K.4.1 [**Computing Milieux**]: Computers and Society–Privacy

Keywords

Anonymity; Website fingerprinting attacks and defenses

1. INTRODUCTION

Website fingerprinting attacks enable an adversary to infer which website a victim is visiting, even if the victim uses an encrypting proxy, such as Tor. These privacy mechanisms encrypt the content transferred between the web server and client, but they do not effectively hide the size, timing, and direction of packets. A website fingerprinting attack uses these features to infer the web page being loaded by a client.

Website fingerprinting attacks have emerged as a serious threat against web browsing privacy mechanisms, such as SSL, Tor, and encrypting tunnels. At the 2012 Oakland conference, Dyer, et al. [5] showed that an attacker could infer, with a success rate over 80%, which of 128 pages a victim was visiting, even if the victim used network-level countermeasures. At CCS 2012, Cai et al. [3] described an attack that could achieve a greater than 75% success rate (out of 100 sites) against numerous network and application-level defenses.

Published and deployed defenses have high overhead, poor security, or both. The Tor project has incorporated network- and browser-level defenses against fingerprinting attacks into its Tor Browser Bundle [13], but Cai found that they provide no security benefit. Luo, et al. proposed HTTPOS [11], a collection of network- and HTTP-level defenses, but Cai's attack showed that it offered little security benefit. Wright, et al. proposed traffic morphing [19], but both Dyer and Cai showed that it provides little protection against fingerprinting attacks. In fact, Dyer and Cai surveyed numerous defenses and found them all ineffective. Dyer proposed BuFLO, which offers good security, but at a high bandwidth cost.

This paper addresses several challenges in designing, evaluating and comparing the performance of website fingerprinting attacks and defenses.

Most attack evaluations have used the artificial "closed-world" model, in which the victim selects one of n websites uniformly randomly and the attacker attempts to guess the chosen website based on the observed network traffic. This model has been criticized for being unrealistic because, in a real attack, the victim may visit any website in the world [14], potentially making the attacker's task much more difficult. Consequently, some researchers have suggested that website fingerprinting attacks are in fact a paper tiger [14].

In this paper we show how to compute the open-world performance of an attack based on its performance in a closed-world experiment. Thus, researchers can evaluate attacks and defenses in the simpler closed-world model and, using our method, compute open-world performance results. We use this method to compute open-world performance of our new defense, Tamaraw.

We also investigate the danger that fingerprinting attacks pose in the real world. We find that, without any defense whatsoever, fingerprinting attacks can pose a significant threat to visitors to popular web pages. For example, an ideal attacker against defenseless victims could recognize visits to the 100 most popular websites with a false discovery rate of less than 50%.

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2957-6/14/11 ...$15.00.
http://dx.doi.org/10.1145/2660267.2660362.

Defense	Security (ϵ)	BW Overhead	Overhead Ratio
Tamaraw	3.4%	199%	31.3
Tamaraw	0.4%	687%	6.9
BuFLO	41.5%	199%	1965
Tor	77.5%	25%	495.2

Table 1: Security and bandwidth overhead of defenses in a closed-world experiment with 800 websites. A defense is ϵ-secure if no attacker can guess the victim's website with probability more than ϵ. Note that this security evaluation is *attack independent*. An overhead ratio of r indicates that the defense incurred an overhead r times larger than the lower bound on overhead derived in Section 5.1.

Most defense evaluations have attempted to estimate the efficacy of the proposed defense by using the state-of-the-art attacks available at that time. This provides only a lower bound on the security of a defense — future attacks may demonstrate that it is insecure. This approach also makes it difficult to compare defenses that were evaluated using different attacks.

We solve both problems by evaluating defenses against an ideal attacker. In our ideal attack, two websites are distinguishable unless they generate the exact same sequence of network traffic observations. Thus evaluating a defense against an ideal attacker gives the lower bound on the security provided by the defense and the evaluation results are attack-independent — i.e., future researchers can compare different defenses using the same ideal attack.

Even when two defenses have been evaluated against the same attack, it can be difficult to compare them, since every defense offers a different trade-off between security and overhead. And even if one defense strictly dominates all other defenses in terms of security and efficiency, it is still not clear whether the attack is optimal. How efficient can a defense be while offering a given level of security?

To answer these questions, we develop an abstract model of website fingerprinting attacks and defenses and prove lower bounds on the bandwidth overhead of any defense that offers a given level of security. This enables us to compare defenses with different overhead/security trade-offs by comparing how close they are to the optimal trade-off curve. We can also bound how far defenses are from optimal. Our bounds suggest that, although defenses are getting better, there may still be significant room for improvement.

In order to design efficient and effective defenses, we need to know which traffic features leak the most information about the website being visited. Without this information, we may make the mistake of designing a defense that incurs high cost to hide a low-information feature. For example, most early defenses focused exclusively on packet sizes, but Cai, et al. showed that packet ordering contains at least as much information as packet sizes [3]. Therefore, we systematically analyze existing attacks and defenses to understand which traffic features convey the most information and therefore are most important for defenses to hide. Our analysis goes beyond the "black-box" approach of previous work that published overall attack success rates but did not investigate the reasons attacks and defenses succeed or fail.

Finally, we propose and evaluate a new, provably secure fingerprinting defense. Tamaraw extends and tunes BuFLO to hide the most significant traffic features uncovered by our feature-based analysis. In particular, we find that BuFLO unnecessarily wastes bandwidth hiding the number of upstream packets and does not adequately hide the total number of downstream packets.

We then use the new evaluation techniques described above to evaluate Tamaraw. We show that Tamaraw offers significantly better security than BuFLO in a closed-world setting. Our closed-world evaluation uses an ideal attacker and therefore bounds the success rate that any attacker can achieve against Tamaraw. Table 1 summarizes the evaluation results. We evaluated Tamaraw under two different configurations. One of them offers over 12× better security than BuFLO for the same bandwidth overhead. Table 1 also shows that all the defenses have a significant gap between their performance and the trade-off lower-bound curve, although Tamaraw comes closest by far.

Then, we show how to compute the open-world performance of an attack based on experimental results derived in the closed-world setting. Further, we show that even the optimal attacker, against defenseless victims, suffers from high false discovery rates. Finally, we evaluate Tamaraw in our open-world model and show that it performs significantly better than BuFLO, against the optimal attacker.

The rest of this paper is organized as follows. Section 2 reviews website fingerprinting attacks. Section 3 describes our feature-based defense comparison methodology. In section 4, we survey a set of six previously proposed defenses and four attack features. We then present the results of applying our feature-based comparative methodology to each of these defenses. Section 5 presents our model of fingerprinting attacks, the security/overhead trade-off lower-bound theorems, and the closed-world/open-world connection. Section 6 describes Tamaraw and presents our evaluation results. Section 8 discusses implications of our research.

2. WEBSITE FINGERPRINTING ATTACKS

In general, website fingerprinting (WF) attacks are a subset of traffic analysis attacks. A *WF attacker* is able to monitor the communications between a client's computer and a private web browsing proxy. The private browsing proxy may be an SSH proxy, VPN server, Tor, or other privacy service. The traffic between the user and proxy is encrypted, so the attacker can only see the timing, direction, and size of packets exchanged between the user and the proxy. Based on this information, the attacker attempts to infer the website(s) that the client is visiting via the proxy by examining various features of the observed traffic. Cai et al. [3] and Chen et al. [4] describe variants wherein an attacker aims to identify a web site instead of a web page. For consistency across the literature, however, we focus on the identification of single web pages. A *WF defense* is a set of countermeasures that protect the client against a *WF attack* by obfuscating, or *covering* features of the web traffic, making them less distinguishable across different web pages, thus reducing the accuracy of the attack.

A *WF attacker* is assumed to be local and passive: the attacker taps and observes from only one location, and he is not allowed to add, drop, or change packets. This means that the attacker is weak, but is also resource-light and essentially undetectable. The attacker can prepare for the attack by collecting information about websites in advance. For example, he can visit websites using the same privacy service as the client, collecting a set of website "fingerprints" as a training set, which he later uses to recognize the client site. These 'fingerprints' usually consist of packet sizes, directions, etc. observed between the client and the proxy. Although the packets themselves are encrypted, the packet sequence still carries information about the sizes of the objects in a web page and the order they are being requested, which is induced by the structure of the web page (the list of resources, their lengths, and which resources each resource requests). The interaction of multiple connections between a client and a web server may cause randomization in packet ordering but, as we will see later, a successful WF attack will tolerate these randomized differences while learning to distinguish different web pages.

We retain two assumptions that all previous works on WF have made of the attacker. First, the attacker is able to notice exactly when a new page is requested. In other words, the attacker knows which sequence of packets corresponds to a single page. This assumption is sometimes made in a stronger form — that the client's think time always dominates page load time — which implies that a distinct pause exists between any two packet sequences of different page loads. Second, any background activity the client may have will not interfere with the client's traffic. For example, a client will not download a file while visiting a web page; alternatively, these file download packets can be easily discarded by the attacker. These assumptions are used by all previous works on WF as they simplify the problem, though it should be noted that these assumptions are advantageous for the attacker.

3. FEATURES AND METHODOLOGY

A classifier succeeds at distinguishing between two classes when it is able to discover a consistent difference between them. This can be viewed as a difference between their features, which characterize a class. Implicitly or explicitly, classification techniques such as WF attacks extract features to classify. Conversely, a successful defense effectively hides these features. In this section, we describe our methodology and use it to evaluate the strengths and weaknesses of different WF defenses.

3.1 Packet Sequences and their Features

In general, packet sequences have four major features: unique packet lengths, packet length frequency, packet ordering, and interpacket timing. Therefore, a packet sequence P can be written as:

$$P = \langle (t_1, \ell_1), (t_2, \ell_2), ..., (t_n, \ell_n) \rangle$$

In the above, t_i is the difference in time observed between packets i and $i-1$ (interpacket timing), with $t_1 = 0$; ℓ_i is the byte length of packet i. The sequence length, $|P|$, is equal to n. We write P_t and P_ℓ as the sequences of only the interpacket times and only the packet lengths, respectively. We indicate the packet length as a positive value if the packet is outgoing and as a negative value if it is incoming.

Unique Packet Lengths: Packet lengths are a simple and strong feature of a web page. GET request lengths are partly determined by the length of the resource name. Incoming packets are almost always sent at the Maximum Transmission Unit (MTU), with the length of the last packet indicating the size of the resource (modulo the MTU).

Most packet lengths of a page are unlikely to change unless resource lengths change. WF attacks almost always consider packet lengths unless they are designed for the Tor scenario, in which case packet lengths are covered. When unspecified, we assume that packet lengths are not hidden from the attacker.

Mathematically, sequences P and P' are said to have different unique packet lengths iff their sets of packet lengths are different – i.e.,

$$(\exists L \in P_\ell | L \notin P'_\ell) \vee (\exists L \in P'_\ell | L \notin P_\ell)$$

Packet Length Frequency: Packet length frequency is the number of times each unique packet length occurs. The number of incoming packets at MTU size is a rough estimation of the total size of the page, which changes due to random advertisements and updated content. We will subsequently show that all current WF defenses fail to cover the total traffic size, as doing so is difficult and would necessarily incur a large traffic overhead. Therefore, the poor performance of most early WF attacks [6, 9, 10] may be attributed to the fact that they explicitly discard packet length frequencies.

Suppose $n_L(P_\ell)$ is the number of times packet length L appears in P_ℓ. P and P' are said be have different packet length frequencies iff their packet lengths occur at different frequencies while excluding packet lengths that are unique to either P or P' – i.e.,

$$\exists L | n_L(P_\ell) \neq n_L(P'_\ell) \wedge n_L(P_\ell) > 0 \wedge n_L(P'_\ell) > 0$$

Packet Ordering: The structure of a page induces a logical order in its packet sequence. As an example, a GET packet for a resource can only be sent once the reference to that resource is received by the client. An attacker may be able to infer information about the content of each packet from observing packet ordering. Further, packet ordering depends on network conditions: it may vary due to bandwidth and latency, and it may be affected by changing the parameters for persistent HTTP connections, pipelining, and so on. Tor developers have implemented a prototype defense based on packet ordering by randomizing request order (see Section 4.1).

We denote M_ℓ as the multiset of packet lengths in P_ℓ, without ordering. We say that two sequences P and P' have different packet ordering iff:

$$M_\ell = M'_\ell \wedge P_\ell \neq P'_\ell$$

Interpacket Timing: Interpacket times often reveal the logical relationship between packets. For example, viewing from the client's end, the outgoing server connection SYN and the incoming SYN-ACK will differ by a round-trip time; so will the GET request and the first packet of that resource. If the attacker's tap is near the client, then the attacker can infer that an outgoing packet and an incoming packet cannot be logically dependent if their interpacket time is less than one RTT.

Suppose P and P' have sequence lengths $|P|$ and $|P'|$. P and P' are said to have different interpacket timings iff their timings are different – i.e.,

$$\exists i, 1 \leq i \leq \min(|P|, |P'|) : (P_t)_i \neq \left(P'_t\right)_i$$

FACT 1. *If $P \neq P'$, then they must differ in at least one of the four features above.*

This fact demonstrates that our choice of features is, in some sense, complete, in that it represents any difference between two packet sequences. We can therefore claim that successful attacks should expose at least one of those four features between packet sequences, while defenses are perfectly effective if they can hide all four features.

FACT 2. *Given any packet sequence P and any subset of the above features, there exists a packet sequence P' such that P and P' only differ in this subset of features, with the exception that different packet ordering implies that unique packet lengths and packet length frequencies do not differ.*

This fact implies that our features are somewhat independent of each other (except packet ordering). It should therefore be possible to find generators that change one feature without affecting the others, allowing us to pinpoint which features various defenses attempt to cover. It also shows that the features considered in this paper are exhaustive, even for data-dependent features (For example, if the size of a response depends on the content of a cookie).

3.2 Comparative Methodology

To determine if a defense is able to hide a feature, we apply the defense to two classes, C and C', which differ only by that feature. Then, we say that a defense is successful in hiding the feature if after applying the defense, there is no discernible difference between C and C'.

We use a generator G to transform C into C' by causing a change in some feature of each packet sequence $P \in C$ and inserting the output into C'. We parameterize $G^{(v)}$ by v, a non-negative integer such that the greater the value, the more "different" P and $P' = G^{(v)}(P)$ will be; we require $G^{(0)}(P) = P$. We design each generator to modify only one specific feature. $G^{(v)}$ operates from the start of the packet sequence. Informally, $G^{(v)}$ is equivalent to $G^{(1)}$ repeated v times, possibly from different starting points in the packet sequence. For all of our generators, this interpretation ensures that v functions as a magnitude. The designed generators are not randomized and each generator may accept values of v up to 180 or $|P|/5$, whichever is lower. None of the generators produce a packet length greater than the MTU. We give a textual description of each generator below and mathematically define each generator $G_i^{(v)}$ in Table 2. More details of the generators can be found in our tech report. [17]

Small Packet-Length Changes (G_1): All packet lengths are increased by v, up to MTU.

Large Packet-Length Changes (G_2): v packet lengths are increased by 1000, up to MTU.

Packet-Length Diffusion (G_3): The lengths of v packet are increased by their position divided by 5, up to MTU.

Appended Incoming Packets (G_4): v incoming MTU packets are appended to the end.

Appended Outgoing Packets (G_5): v outgoing packets are appended to the end, their lengths being the lengths of the first outgoing packets of P.

Inserted Incoming Packets (G_6): v incoming MTU packets are added, one per 5 packets.

Adjacent Transpositions (G_7): v packets are transposed with the previous packet.

Short-Distance Transpositions (G_8): v packets are transposed with the packet 4 elements ago.

Long-Distance Transpositions (G_9): v packets are transposed with the packet 19 elements ago.

Delays (G_{10}): Each packet is delayed by a linearly increasing amount of time, multiplied by v.

3.3 Classification and Experimental Setup

In order to understand the significance of these traffic features for fingerprinting attacks, we focus on distinguishing between two classes:

$$C = \{P_1, P_2, ..., P_{400}\}$$

$$C' = \{G^{(v)}(P_1), G^{(v)}(P_2), ..., G^{(v)}(P_{400})\}$$

These are the original class (C) and the generator-modified class (C') with one feature changed. Since our generators operate on packet sequences, the elements of C and C' are packet sequences.

We construct C by connecting to `bbc.co.uk` 400 times with caching disabled. The reason we do so is that C should contain packet sequences of the same page rather than different pages, because WF attack classifiers are designed to tolerate the randomness within the same page while exposing the differences between different pages. A successful classifier should therefore be able to distinguish C and C' despite the randomness in C (and therefore C'). On the other hand, the elements of C need to differ from each other; this will allow us to measure how sensitive each defense is to the generators' operations. The page we chose has a suitable amount of randomness and has therefore been difficult to classify [18]. We use the first 200 elements of classes C and C' for training and the last 200 elements for testing our feature classifiers described below. The training and testing sets are denoted C_{train} and C_{test}, with the generator-modified sets being C'_{train} and C'_{test}. We use four different *feature classifiers,* each one specializing on differentiating one specific feature.

Unique Packet-Lengths (F_1): Given a unique packet length in P_{test}, if it is in any packet sequence $P \in C$ and not in any $P' \in C'$, add 1 to the score of class C and vice versa. This is a modified version of the early Jaccard co-efficient classifier introduced in [9].

Packet-Length Frequencies (F_2): For training, we count the total number of bytes contained in incoming and outgoing packets, as well as the total number of incoming and outgoing packets, and take the mean and standard deviation over the packet sequences in each class. Each testing packet sequence is then scored using the normal distribution kernel against those four values for each class, with the incoming and outgoing packets scored separately and then multiplied. This classifier is a simplified version of the Naïve Bayes classifier also described in [9].

Packet Ordering (F_3): For testing, each packet length in the sequence (discarding direction) is compared to the mean of all training packet lengths at that position. This classifier is derived from the Cross Correlation classifier described in [2].

Interpacket Timing (F_4): Classification here is based only on total elapsed time. We use this classifier because G_{10} is a delay. This reveals whether or not the total load time of a page would still be a useful feature after the defense is applied.

Since the objective is simply to find the difference between the two classes that differ only by a single feature, the above four single feature classifiers are sufficient.

The defense D is applied to each element of C_{train} and C'_{train} to produce $D(C_{train})$ and $D(C'_{train})$; the feature classifier is then trained to distinguish between them. Finally, D is applied to C_{test} and C'_{test} and its effectiveness is measured by the feature classifiers. The effectiveness of the defense D is measured by the value of v before the classifier is able to distinguish between the two classes with reasonable levels of accuracy (.55 and .75) – i.e., the magnitude of differences induced between C and C' before the classifier can distinguish between them.

Our experimental setup is as follows. We load pages over a 100 Mbps Ethernet connection with MTU set at 1500. For automated page loading, we used iMacros 9.00 on Firefox 23.0. We collected data with `tcpdump` and parsed the data into packet sequences with our code. We implemented all of the classifiers and simulated defenses in this paper in a combination of Python, C++, and C, and all of them are available upon request.

4. COMPARISON OF DEFENSES

In this section, we survey the current state-of-the-art of WF defenses and present the results of our comparative evaluation based on the methodology described in Section 3. We then analyze our findings and try to shed light on previously unexplained results. Our simulations of each of the listed defenses operate on packet sequences (packet lengths and timings, but no content) rather than raw packets; they do not directly modify packets during page loading. This allows us to observe if a defense is able to cover a feature modified by a generator.

4.1 Simulated Defenses

We simulate and compare the following network level fingerprinting defenses:

Feature type	Generator name	#	Transformation for $G_{\#}^{(v)}$
Unique packet sizes	Small packet size changes	1	For $0 < i \leq \lvert P \rvert$: $\ell_i \leftarrow d_i \min(\lvert \ell_i \rvert + v, 1500)$
	Large packet size changes	2	For $0 < i \leq v$: $\ell_{5i} \leftarrow d_{5i} \min(\lvert \ell_{5i} \rvert + 1000, 1500)$
	Diffusing packet sizes	3	For $0 < i \leq v$: $\ell_{5i} \leftarrow d_{5i} \min(\lvert \ell_{5i} \rvert + i, 1500)$
Packet size frequencies	Append inbound MTU packets	4	Repeat v times: $P \leftarrow A(P, N, (t_N, -1500))$
	Append outbound packets	5	For $0 < i \leq v$: $P \leftarrow A(P, N, (t_N, P_{out_i}))$
	Insert inbound MTU packets	6	For $0 < i \leq v$: $P \leftarrow A(P, 5i, (t_{5i}, -1500))$
Packet ordering	Adjacent transpositions	7	For $0 < i \leq v$: $T(P, 5i, 5i - 1)$
	Short distance transpositions	8	For $0 < i \leq v$: $T(P, 5i, 5i - 4)$
	Long distance transpositions	9	For $0 < i \leq \lfloor v/5 \rfloor, 0 < j \leq 5$: $T(P, 25i - j, 25i - 19 - j)$
Interpacket timing	Delays	10	$\forall p_i \in P, t_i \leftarrow t_i + v \cdot i \cdot 0.020\,\mathrm{ms}$

Table 2: Generators. Packet sequence $P = \{p_1, p_2, ..., p_N\}$ where $p_i = (t_i, \ell_i)$, $t_1 = 0$. P_{out} is the sequence of outgoing packets in P, and P_{out_i} is its i^{th} element (wrapping back to the beginning if $i > \lvert P_{out} \rvert$). d_i is the direction (1=outgoing, -1=incoming) of p_i. $A(P, i, p)$ appends packet p after p_i. $T(P, i, j)$ transposes the packet lengths of p_i and p_j.

Defense	Unique packet lengths									Packet length counts									Packet ordering									Timing		
	G_1			G_2			G_3			G_4			G_5			G_6			G_7			G_8			G_9			G_{10}		
PadM	*	*	*	*	*	*	*	*	*	F_2	16	57	F_2	3	9	F_2	16	57	*	*	*	*	*	*	*	*	*	F_4	23	47
PadE	F_2	28	91	F_3	1	3	F_3	59	*	F_3	2	27	F_2	2	9	F_3	1	1	F_3	29	*	F_3	2	3	F_3	4	4	F_4	23	47
Wr-Morph	F_2	13	74	F_2	29	180	*	*	*	F_2	43	72	F_2	2	11	F_2	32	68	*	*	*	*	*	*	*	*	*	F_4	23	47
HTTPOS	F_3	29	30	F_3	23	50	F_3	45	*	F_3	6	18	F_2	3	9	F_3	1	3	F_3	179	*	F_3	2	*	F_3	4	4	F_4	23	47
Pa-Decoy	F_1	1	68	F_1	48	178	F_1	34	*	F_3	93	*	F_2	38	151	F_3	1	*	*	*	*	*	*	*	F_3	14	*	F_4	111	146
BuFLO	F_2	147	*	*	*	*	*	*	*	F_2	23	85	F_2	78	*	F_2	20	94	*	*	*	*	*	*	*	*	*	*	*	*
Tamaraw	F_2	177	*	F_2	180	*	*	*	*	F_2	122	*	F_2	68	168	F_2	82	*	*	*	*	*	*	*	*	*	*	*	*	*

Table 3: Upper bounds on the quality of the defenses. Results are given in three columns: the first column is the feature classifier that was able to achieve the highest mean accuracy, the second is the value of v for which the feature classifier in column 1 had an accuracy greater than 0.55 for all greater values of v, and the third is similar, but for 0.75; asterisks indicate when these accuracies were not reached for $v \leq 180$. Asterisks and higher v values indicate that the corresponding defense is more successful in covering the specific feature. Tamaraw is our BuFLO-based defense presented in Section 6.

Maximum Packet Padding (PadM): Padding adds garbage data to the end of packets in order to obscure their true length. Packet padding schemes have been known in the literature; a large number of different padding schemes were analyzed by Dyer et al. [5] and shown to be ineffective against the attack classifiers described in [9] and [12]. Packet padding schemes are meant to obscure the unique packet length feature. In the PadM defense, all packet lengths are padded to the MTU. The effect of using Tor is similar to PadM, as Tor traffic is delivered in fixed-size cells.

Exponential Packet Padding (PadE): Here, packet lengths are padded upwards to the closest power of 2, but not exceeding the MTU. No extra packets are added in either of the two schemes. PadE is meant to be a bandwidth-cheaper version of PadM.

Traffic Morphing (Wr-Morph): Wright et al. [19] published a WF defense that is meant to obscure packet lengths, known as traffic morphing. Unique among the defenses, Wr-Morph is designed to allow the client to set a target page C_T and mimic the page's packet size distribution by modifying packet lengths.

In our implementation, we use `google.com` as C_T, our target morph page since it is reasonable for the client to attempt to hide her traffic features by using the most commonly accessed page as a decoy.

HTTP Obfuscation (HTTPOS): HTTPOS was presented by Luo et al. [11] as a platform for website fingerprinting defenses. Unlike many other defenses, HTTPOS is implemented entirely on the client-side, with no need for support from any proxy or the end server. It does so by using TCP advertised windows, HTTP pipelining, and HTTP ranges in order to control the sizes of both outgoing *and incoming* packets.

Our simulation is not done at the application layer; rather, we simply go through the packet sequence and split up each incoming packet of size less than the MTU. In the implementation of HTTPOS, this requires a new outgoing packet between the two splits which signals the end server that the client is ready to receive the second split, which costs one round-trip time. In our simulated defense, we assume this can be done without the signal and round-trip time, which is possible with the cooperation of a proxy or the end server.[1] We complete this defense by also padding all outgoing packets to a fixed size of 1500; the authors describe how outgoing packet padding can be done, but they are not clear as to what they implemented. Our choice gives maximum protection for unique packet lengths, at the cost of extra overhead.

Background Noise (Pa-Decoy): Background noise can be used to add randomness to each page load in order to make them less distinguishable. Tor has some background activity that makes fingerprinting more difficult, such as circuit construction, circuit measurement, and stream SENDMEs.

Panchenko et al. [12] proposed a simple defense using background noise to defeat their own attack. Whenever a page is loaded, a decoy page is loaded in the background, using its own connections and connection limit so as not to interfere with the connections of the intended page. This defense has a high overhead. We

[1] This assumption is reasonable for us as many of the other defenses also require the cooperation of a proxy or the end server.

used a different page from Alexa's top 800 sites as background noise for each of the training and testing elements in order to simulate the intended effect that the attacker cannot predict which page the client picked. Our simulated defense assumes that the decoy page does not interfere at all with the page load of the true page.

Buffered Fixed Length Obfuscator (BuFLO): After Dyer et al. analyzed a large number of traffic analysis countermeasures and found that efficient defenses failed, they presented their own defense, BuFLO — a Buffered Fixed-Length Obfuscator, and demonstrated its relative success against the attacks of the day [5]. The BuFLO defense is an obfuscator with large bandwidth and time overhead that is applied to both the incoming and outgoing traffic. Packets are sent at fixed intervals with fixed length, and if no data needs to be sent, dummy packets are sent instead. The traffic must continue for a minimum amount of time, after which it may terminate if no more data needs to be sent.

BuFLO is parameterized by three values: d the fixed size of packets (set to 1500), ρ the interpacket timing (set to one packet/20 milliseconds), and τ the fixed minimum amount of time for which data must be sent (set to 10 seconds). These parameter settings were the strong (and more bandwidth-intensive) choice presented by Dyer et al., which was able to sharply decrease the accuracy of the Panchenko classifier [12].

4.2 Comparative Results

In this section we highlight the conclusions of our comparative evaluation. The full results are given in Table 3. For each defense, we only present the most successful feature classifier. We only use the interpacket timing classifier on G_{10}. We vary v from 1 to 180 and present the minimum value of v for which the feature classifier had a higher accuracy than 0.55 and 0.75 for all greater v. A lower v indicates that the defense was less successful in covering the feature against our classifiers, whereas an asterisk indicates that our feature classifiers could not distinguish the classes with the target accuracy (0.55 or 0.75) for any value of v that we tested. This table also includes our new proposed defense, Tamaraw (presented in Section 6).

Our evaluation shows that PadM covers unique packet length as well as packet orderings. In PadE, changing packet length can cause the lengths to be padded to different powers of 2, which could affect the set of unique packet lengths. However, the number of distinct lengths with PadE is small, so this is unlikely. Consequently, the unique packet length classifier did not detect a difference. Our packet ordering classifier, however, considered the changes in packet lengths to be re-ordering, and therefore managed to defeat PadE. PadM and PadE are ineffective against attacks that use packet length frequencies.

HTTPOS has been shown to be effective against attack classifiers that are strongly dependent on unique packet lengths [9, 11]. However, Cai, et al.'s attack succeeds against HTTPOS [3] . This is because HTTPOS attempts to remove unique packet lengths as a feature, but Cai's attack primarily uses packet ordering.

Pa-Decoy and BuFLO are effective against the Panchenko classifier and attacks that are dependent on packet length frequencies [5, 12] – i.e., these defenses partly cover packet length frequencies.

The table exposes several flaws. PadM, PadE, Wr-Morph and HTTPOS are not designed to cover total transmission size (packet length frequencies), so they would be ineffective against the attacks that leverage them. Pa-Decoy fails to completely cover interpacket timing because it only covers the total transmission time roughly half the time (i.e., when the decoy page takes longer to load than the desired page), which may leak the total page size, a powerful feature. Similarly, BuFLO does not cover total transmission time if it is over 10 seconds at $\rho = 0.020$ s/packet, which happened quite often. Trying to cover packet length frequency on $G_4^{(v)}$ becomes a race between v and the overhead of BuFLO; a larger v requires a larger setting of minimum time τ to cover it.

Our results are upper bounds on the quality of the defense, as more complicated classifiers could reveal more information. For instance, HTTPOS performs well against F_1, but if C' has larger unique packet lengths than C, then $D(C')$ will also have larger unique packet lengths than $D(C)$ under HTTPOS. We wrote a classifier specifically to find this difference and achieved an accuracy of 0.99 with $G_1^{(100)}$ compared to 0.5 for F_1.

Tor developers want to understand what WF defenses work with Tor [15]. As Tor already covers unique packet length, PadM, PadE, Wr-Morph, and HTTPOS are not meaningful on Tor, as all of these defenses are focused on covering unique packet lengths (although only PadM truly does so). We note in particular that HTTPOS is a platform valuable for its client-only implementation (requiring no cooperation from a proxy or the end server), but Tor bridges can be made to cooperate by implementing a WF defense as a pluggable transport. Pa-Decoy and Dy-BuFLO achieve only limited success at covering packet length frequencies. In short, none of these defenses can be considered a perfect fit for Tor.

5. THEORETICAL FOUNDATIONS

In this section we develop a model of website fingerprinting attacks and defenses, derive lower bounds on the bandwidth overhead of any defense that achieves a given level of security, and show how to derive open-world performance from closed-world experimental results.

5.1 Security vs. Overhead Trade-Off

We focus on understanding the relationship between bandwidth overhead and security guarantees. The overhead required by a fingerprinting defense depends on the set of web sites to be protected – a set of similar websites can be protected with little overhead, a set of dissimilar websites requires more overhead. To derive lower bounds of bandwidth costs, we consider an *offline* version of the website fingerprinting defense problem, i.e. the defense system knows, in advance, the set of websites that the user may visit and the packet traces that each website may generate. We develop an efficient dynamic program to compute a lower bound on the bandwidth overhead of any fingerprinting defense scheme in the closed-world setting. We will use this algorithm to compute lower bounds on overhead for the websites used in our evaluation (in Section 6.2).

5.1.1 Definitions

In a website fingerprinting attack, the defender selects a website, w, and uses the defense mechanism to load the website, producing a packet trace, t, that is observed by the attacker. The attacker then attempts to guess w.

Let W be a random variable representing the URL of the website selected by the defender. The probability distribution of W reflects the probability that the defender visits each website. For each website, w, let T_w^D and T_w be the random variables representing the packet trace generated by loading w with and without defense system D, respectively. Packet traces include the time, direction, and content of each packet. Since cryptographic attacks are out of scope for this paper, we assume any encryption functions used by the defense scheme are information-theoretically secure. The probability distribution of T_w^D captures variations in network conditions, changes in dynamically-generated web pages, randomness in the browser, and randomness in the defense system. We assume the attacker knows the distribution of W and T_w^D for every w.

In a closed world setting, the attacker's goal is to infer W from T_W^D. The optimal closed-world attacker, A, upon observing trace t, outputs

$$A(t) = \operatorname*{argmax}_w \Pr[W = w] \Pr\left[T_w^D = t\right]$$

If more than one w attains the maximum, then the attacker chooses randomly among them.

Some privacy applications require good worst-case performance, and some only require good average-case performance. This leads to two security definitions for website fingerprinting defenses:

DEFINITION 1. *A fingerprinting defense D is **non-uniformly ϵ-secure** for W iff* $\Pr\left[A(T_W^D) = W\right] \leq \epsilon$. *Defense D is **uniformly ϵ-secure** for W if* $\max_w \Pr\left[A(T_w^D) = w\right] \leq \epsilon$.

These are information-theoretic security definitions – A is the optimal attacker described above. The first definition says that A's average success rate is less than ϵ, but it does not require that every website be difficult to recognize. The second definition requires all websites to be at least ϵ difficult to recognize. All previous papers on website fingerprinting attacks and defenses have reported average attack success rates in the closed-world model, i.e. they have reported non-uniform security measurements. We will do the same.

To define the bandwidth overhead of a defense system, let $B(t)$ be the total number of bytes transmitted in trace t. We define the **bandwidth ratio** of defense D as

$$\mathrm{BWRatio}_D(W) = \frac{E\left[B\left(T_W^D\right)\right]}{E\left[B\left(T_W\right)\right]}$$

This definition captures the overall bandwidth ratio between a user surfing the web while using defense D and a user visiting the same websites with no defense.

5.1.2 Bandwidth Lower Bounds

In this section we derive an algorithm to compute, given websites $w_1, \dots, w_n$, a lower bound for the bandwidth that any non-uniformly ϵ-secure fingerprinting defense can use in a closed-world experiment using $w_1, \dots, w_n$.

To compute a lower bound on bandwidth, we consider an adversary that looks only at the total number of bytes in a packet trace, i.e. an attacker A_S that always guesses

$$A_S(t) = \operatorname*{argmax}_w \Pr\left[B(T_w^D) = B(t)\right]$$

Any defense that is ϵ-secure against an arbitrary attacker must also be at least ϵ-secure against A_S. If we can derive a lower bound on defenses that are ϵ-secure against A_S, that lower bound will apply to any ϵ-secure defense.

We make two simplifying assumptions in order to obtain an efficient algorithm for computing lower bounds. First, we assume that each website has a unique fixed size, s_i. In our closed world experiments, we found that, for just over half the web pages in our dataset, their size had a normalized standard deviation of less than 0.11 across 20 loads, so we do not believe this assumption will significantly impact the results of our analysis. Second, we assume that the defense mechanism does not compress or truncate the website.

We prove the following theorem in Appendix A:

THEOREM 1. *Suppose ϵn is an integer. Let W be a random variable uniformly distributed over $w_1, \dots, w_n$, i.e. W represents a closed-world experiment. Suppose D is a defense that is ϵ-non-uniformly-secure against A_S on distribution W. Then there exists a monotonically increasing function f from $S = \{s_1, \dots, s_n\}$ to itself such that*

- $|f(S)| \leq \epsilon n$.
- $\sum_{i=1}^n f(s_i) / \sum_{i=1}^n s_i \leq \mathrm{BWRatio}_D(W)$.

Intuitively, f represents a mapping from each website's original size (s_i) to the number of bytes that D transmits when loading website w_i.

This theorem enables us to efficiently compute a lower bound on the overhead of any defense that is ϵ uniformly or non-uniformly secure in a closed world experiment on $w_1, \dots, w_n$. To get a lower bound for non-uniformly ϵ-secure defenses, we just need to find a monotonically increasing function $f : S \to S$ that satisfies $|f(S)| \leq \epsilon n$ and minimizes $\sum_{i=1}^n f(s_i)$.

Such an f is equivalent to a partition $S_1, \dots, S_k$ of S satisfying $k \leq \epsilon n$ and minimizing $\sum_{i=1}^k |S_i| \max_{s \in S_i} s$. These partitions satisfy a recurrence relation. If $S_1, \dots, S_k$ is an optimal non-uniformly $\frac{k}{n}$-secure partition, then $S_1, \dots, S_{k-1}$ is an optimal non-uniformly $\frac{k-1}{n-|S_k|}$-secure partition of $S_1 \cup \dots \cup S_{k-1}$. Therefore the cost, $C(\frac{k}{n}, n)$, of the optimal f satisfies the recurrence

$$C(\frac{k}{n}, n) = \begin{cases} n s_n & \text{if } k = 1 \\ \displaystyle\min_{1 \leq j \leq n-1} C(\frac{k-1}{n-j}, n-j) + j s_n & \text{otherwise.} \end{cases}$$

We can obtain a similar bound for uniformly ϵ-secure deterministic defenses. We say a defense is deterministic if, on each load of website w_i, it always transmits b_i bytes. The following theorem is proven in Appendix A.

THEOREM 2. *Let W be uniformly distributed over $w_1, \dots, w_n$, i.e. W represents a closed-world experiment. Suppose D is a deterministic defense that is uniformly ϵ-secure against A_S on distribution W. Then there exists a monotonically increasing function f from $S = \{s_1, \dots, s_n\}$ to itself such that*

- $\min_i |f^{-1}(s_i)| \geq 1/\epsilon$.
- $\sum_{i=1}^n f(s_i) / \sum_{i=1}^n s_i \leq \mathrm{BWRatio}_D(W)$.

As with the lower bound on non-uniformly secure defenses, such an f corresponds to a partition $S_1, \dots, S_k$ of S satisfying $\min_i$ $|S_i| \geq 1/\epsilon$ and minimizing $\sum_{i=1}^k |S_i| \max_{s \in S_i} s$. These partitions satisfy a slightly different recurrence. If $S_1, \dots, S_k$ is is an optimal uniformly ϵ-secure partition of S, then $S_1, \dots, S_{k-1}$ is an optimal uniformly ϵ-secure partition on $S_1 \cup \dots \cup S_{k-1}$. Thus the cost, $C(\epsilon, n)$ of the optimal uniformly ϵ-secure partition satisfies the recurrence relation:

$$C'(\epsilon, n) = \begin{cases} \infty & \text{if } n < 1/\epsilon \\ n s_n & \text{if } n \in \left[\frac{1}{\epsilon}, \frac{2}{\epsilon}\right) \\ \displaystyle\min_{1 \leq j \leq \frac{n-1}{\epsilon}} C'(\epsilon, n-j) + j s_n & \text{otherwise.} \end{cases}$$

Algorithm 1 shows a dynamic program for computing a lower bound on the bandwidth of any defense that can achieve ϵ non-uniform security in a closed-world experiment on static websites with sizes $s_1, \dots, s_n$ in time $O(n^2 \epsilon)$. We use this algorithm to compute the lower bounds reported in Section 6.2. The dynamic program for computing uniform security lower bounds is similar.

5.2 From Closed to Open World

In this section, we show how to use closed-world experimental results to compute open-world security of defenses and open-world performance of attacks. This makes attack and defense evaluation simpler: researchers need only perform closed-world experiments to predict open-world performance.

In an open-world attack, the defender selects a website, W, according to some probability distribution and generates a trace, T_W^D, corresponding to a visit to that website using some defense, D.

The attacker's goal is to determine whether $W = w^*$, where w^* is a particular website of interest. (It is easy to generalize this definition to situations with multiple websites of interest).

In the open-world setting, the distribution of the random variable W corresponds to the popularity of different websites among the population of users being monitored in the attack. So, for example, if the fingerprinting attacker is a government monitoring citizens Tor usage, then W would be distributed according to the popularity of websites among that nation's Tor users.

Any closed-world attack can be used to construct an open-world attack by selecting websites $w_2, \ldots, w_n$ and building a closed-world classifier, A, on $w^*, w_2, \ldots, w_n$. The open-world classifier is defined as $C(t) = 1$ iff $A(t) = w^*$.

We can compute the false positive rate of this open-world attack as follows. Let $p^* = \Pr[W = w^*]$ and $p_i = \Pr[W = w_i]$ for $i = 2, \ldots, n$. We can obtain estimates for $p^*, p_2, \ldots, p_n$ from public sources, such as the Alexa Page-Views per Million database [1]. Let R_n be the average success rate of A in the closed world, i.e.

$$R_n = \frac{\Pr[A(T^D_{w^*}) = w^*] + \sum\limits_{i=2}^{n} \Pr[A(T^D_{w_i}) = w_i]}{n}$$

Note that R_n is the standard performance metric used in closed-world evaluations. For simplicity, we assume that $\Pr[A(T^D_{w^*}) = w^*] = R_n$. We also assume that, whenever A misclassifies a trace, there is a $1/n$ chance that it misclassifies the trace as w^*, i.e. that $\Pr[A(T^D_W) = w^* | W \neq w^* \wedge A(T^D_W) \neq W] = 1/n$. Essentially, these two assumptions are equivalent to assuming that w^* is not particularly difficult or easy for A to recognize. With these assumptions, we can compute C's false-positive rate:

$$\begin{aligned}
\mathrm{FPR}(C) &= \Pr[C(T^D_W) = 1 | W \neq w^*] \\
&= \sum_{w \neq w^*} \frac{\Pr[W = w] \Pr[C(T^D_w) = 1]}{1 - p^*} \\
&= \sum_{w \neq w^*} \frac{\Pr[W = w] \Pr[A(T^D_w) = w^*]}{1 - p^*} \\
&= \sum_{i=2}^{n} \frac{\Pr[W = w_i] \Pr[A(T^D_{w_i}) = w^*]}{1 - p^*} \\
&+ \left(1 - \sum_{i=2}^{n} \Pr[W = w_i]\right) \frac{1}{n(1 - p^*)} \\
&= \frac{1 - R_n}{n(1 - p^*)} \sum_{i=2}^{n} p_i + \frac{1}{n(1 - p^*)} \left(1 - \sum_{i=2}^{n} p_i\right)
\end{aligned}$$

With the same assumptions, the true positive rate of C is

$$\mathrm{TPR}(C) = \Pr[C(T^D_W) = 1 | W = w^*] = R_n$$

The choice of the websites $w_2, \ldots, w_n$ used to build A will affect the performance of C in the open world. The choice of websites affects the false-positive rate in two ways: (1) choosing less popular websites tends to increase the false-positive rate since it decreases $\sum\limits_{i=2}^{n} p_i$, and (2) choosing more similar websites increases the false-positive rate by reducing R_n. The choice of websites affects the true-positive rate only through R_n. Cai, et al., showed that the Alexa top 100 websites were about as similar as 100 randomly chosen websites [3], i.e. that the most popular websites are not particularly similar to eachother. Thus it is generally a good strategy to choose $w_2, \ldots, w_n$ to be the most popular websites other than w^*.

Similarly, the number, n, of websites used to build A affects the false-positive rate in two ways: (1) increasing n tends to increase the false positive rate by lowering R_n, and (2) increasing n tends to decrease the false-positive rate since it increases $\sum\limits_{i=2}^{n} p_i$. Increasing n can only decrease the true-positive rate.

Thus we can tune the false-positive and true-positive rates of C by varying n. Small n will have large true- and false-positive rates. Increasing n will reduce both the false- and true-positive rates. By varying n, we can generate the receiver operating curve (ROC) of C.

In the real world, visits to w^* may be rare. In this case, false-positive rate can be a misleading metric. A classifier with a low false-positive rate may still be useless if true positives are so rare that they are overwhelmed by false positives. Therefore, we also report true-discovery rates for the open-world attack and defense evaluations in this paper. Given an open-world classifier, C, its true-discovery rate is defined as

$$\mathrm{TDR}(C) = \Pr[W = w^* | C(T^D_W) = 1].$$

Intuitively, the true-discovery rate is the fraction of alarms that are true alarms. The true-discovery rate can be computed from the false-positive and true-positive rates as follows:

$$\begin{aligned}
\mathrm{TDR}(C) &= \frac{\Pr[W = w^*] \,\mathrm{TPR}(C)}{\Pr[W = w^*] \,\mathrm{TPR}(C) + \Pr[W \neq w^*] \,\mathrm{FPR}(C)} \\
&= \frac{p^* R_n}{p^* R_n + \frac{1 - R_n}{n} \sum\limits_{i=2}^{n} p_i + (1 - \sum_{i=2}^{n} p_i) \frac{1}{n}}
\end{aligned}$$

Algorithm 1 Algorithm to compute a lower bound on the bandwidth of any offline non-uniformly ϵ secure fingerprinting defense against A_S attackers.

```
function $A_S$-MIN-COST($n$, $\epsilon$, $\{s_1, \ldots, s_n\}$)
    Array $C[0 \ldots n\epsilon, 0 \ldots n]$
    for $i = 0, \ldots, n\epsilon$ do
        $C[i, 0] \leftarrow 0$
    end for
    for $i = 0, \ldots, n$ do
        $C[0, i] \leftarrow \infty$
    end for
    for $i = 1 \to n$ do
        for $j = 1 \to n\epsilon$ do
            $C[j, i] = \min_{1 \leq \ell \leq i-1} [(i - \ell) s_i + C[j - 1, \ell]]$
        end for
    end for
    return $C[n\epsilon, n]$
end function
```

6. TAMARAW: A NEW DEFENSE

In this section we present a prototype of a new defense, Tamaraw[2], that can be considered a theoretically provable version of BuFLO.

6.1 Design

Based on the results of our comparative and theoretical study of website fingerprinting defenses, Tamaraw is designed with three guiding principles in mind:

1. **Strong Theoretical Foundations:** The security of the Tamaraw defense is based on an extension of the concept of optimal partitioning and feature hiding demonstrated in Section 5.1 (against A_S attackers). The relation is seen in section Section 6.2.1.

[2] A Tamaraw is a lightweight cousin of the Buffalo.

2. **Feature coverage:** While Section 5.1 aims to hide only the total transmission size, Tamaraw attempts to hide all features. In fact, Table 3 shows that Tamaraw effectively hides all the features studied in Section 3, with the exception of total downstream transmission size. As in Section 5.1, optimal partitioning requires different sites to be padded to different transmission sizes.

3. **Reducing Overhead Costs:** BuFLO faces the dilemma that increasing τ will increase its defensive coverage, but also increase its overhead. We address this dilemma and we are able to find ways to significantly reduce the overhead of BuFLO in both bandwidth and time.

Tamaraw works as follows. As in BuFLO, traffic is still sent in fixed size packets and at fixed intervals; however, the packet size is set at 750 bytes rather than the MTU. This is done since most outgoing packets are covered by this size (see Appendix: Figure 3) while not incurring unwanted overhead. Further, incoming and outgoing traffic are treated differently. Outgoing traffic is fixed at a higher packet interval, which saves overhead as outgoing traffic is much less frequent. We denote the packet intervals as ρ_{out} and ρ_{in} (measured in s/packet). We use experimentation to choose these values of ρ in Section 6.2.

Additionally, BuFLO only attempts to cover total transmission size if the total amount of time for one page is less than τ. This makes the choice of τ especially awkward: increasing τ increases the number of web pages covered by BuFLO, but it also increases the overhead. In Tamaraw, however, the number of packets sent in both directions are always padded to multiples of a padding parameter, L.[3] This means that if L is large enough, then it is likely that for each web page there exists some other web page that is mapped to the same multiple of L. Suppose the total number of incoming packets is I, where $AL < I \leq (A+1)L$, then we pad to $(A+1)L$ at the rate ρ_{in}. We do this separately for outgoing packets as well. Compared to the optimal defense in Section 5.1, the partitions produced are fixed, independent of the dataset.

Even though the differences between BuFLO and Tamaraw are not very large, the impact on security is tremendous: Tamaraw offers a maximum attack-accuracy guarantee, BuFLO does not.

6.2 Experimental Results

In BuFLO, ρ_{out} and ρ_{in} were both 0.02, but it is expected that ρ_{out} should not have to be as large as ρ_{in}. As the distinguishability of two different web pages is controlled by the padding parameter L, our objective in the choice of ρ_{in} and ρ_{out} is to minimize overhead. We test the bandwidth and time overhead of Tamaraw on Alexa's top 100 pages, loaded 70 times each. We vary ρ_{out} and ρ_{in} from 0.005 to 0.16 seconds/packet while using Tamaraw with MTU packet sizes. We present a pareto curve showing all values for which no other choice of parameters had both a lower time and bandwidth overhead in the Appendix: Figure 4.

Generally, as ρ_{in} and ρ_{out} increased, size overhead decreased while time overhead increased. Here we set L to 100, for reasons discussed below. With MTU packets, $\rho_{out} = 0.04$ and $\rho_{in} = 0.012$, we achieve a 17% decrease of total transmission size overhead from BuFLO's 149 ± 6% to 123 ± 10%, with the time overhead roughly the same, changing from 330 ± 80% to 320 ± 70%. In order to achieve these same benefits when we use Tamaraw with 750 byte packet sizes, we maintain the same transmission rate. This is achieved by halving ρ_{in} and ρ_{out} – i.e., we set $\rho_{out} = 0.02$ and $\rho_{in} = 0.006$.

In our implementations of BuFLO and Tamaraw, we pessimistically required that the original logical ordering of the real packets must be maintained. For example, if the defense allowed an outgoing packet to be sent, but the next real packet to be sent is an incoming packet, then a dummy outgoing packet is sent, even if there are other outgoing packets waiting after the incoming packet. This is to guarantee that the causal order is preserved: it could be that the subsequent outgoing packets *depend* on the incoming packet. This rule has a large effect on the bandwidth. A practical implementation could achieve a lower size and time overhead as re-ordering is possible for both defenses when subsequence is not consequence; our simulations are therefore pessimistic on the overhead but necessary to guarantee correctness.

We apply the same defense methodology in Section 4.2, and give the results in Table 3, presented earlier. We can see that at $\rho_{out} = 0.02$, $\rho_{in} = 0.006$, and $L = 100$, Tamaraw is much more successful at protecting all features than other defenses, despite that it cannot achieve a perfect cover of total packet length or frequency either. Looking more closely at the table, we see that Tamaraw is not perfectly successful against generators that significantly change the total transmission size (including the unique packet length generators G_1 and G_2). As BuFLO sends more dummy outgoing packets than Tamaraw, BuFLO is more able to cover changes in outgoing transmission size (G_2, G_5), but it is less able to cover changes in incoming transmission size (G_1, G_4, G_6). We next show why Tamaraw is a more effective defense than BuFLO.

6.2.1 An Ideal Attacker

In order to produce an *upper bound* on the attack accuracy of any classifier on Tamaraw, we evaluate the partitions produced by Tamaraw (partitions were introduced in Section 5.1). The number of partitions is directly linked to the maximum classification accuracy. For a partition of size $|S|$, the attacker can at best achieve an accuracy of $1/|S|$ on each site in the partition.

For Tamaraw, the *partition* is calculated as follows. Let D be Tamaraw where L is set to 0 (no dummy packets appended to the end). Suppose the number of incoming packets for defended packet sequence $D(P)$ is $|D(P)_{inc}|$ and the number of outgoing packets is $|D(P)_{out}|$. Two packet sequences $D(P)$ and $D(P')$ are the same under Tamaraw if they satisfy:

$$\left\lfloor \frac{|D(P)_{inc}|}{L} \right\rfloor = \left\lfloor \frac{|D(P')_{inc}|}{L} \right\rfloor, \left\lfloor \frac{|D(P)_{out}|}{L} \right\rfloor = \left\lfloor \frac{|D(P')_{out}|}{L} \right\rfloor.$$

This is the case even if the attacker is able to observe those packet sequences multiple times with the knowledge that they belong to two pages. We experiment on Alexa's top 800 sites. We only load one instance of each web page and reset the browser state between each page load. By doing this, we eliminate the network variability and make the defense system deterministic, which, as shown in the Appendix, does not reduce the security of the defense. Thus we can soundly use this technique to obtain an upper bound on the success rate of an ideal attacker against this defense. For BuFLO, we consider two packet sequences to belong to the same partition if the total transmission size is the same, as total transmission size is the only observable difference.

6.2.2 Closed-world Performance

Figure 1 shows the non-uniform security provided by Tamaraw and BuFLO against their corresponding bandwidth overheads. The BuFLO points correspond to the BuFLO configurations evaluated by Dyer, et al. [5]. For reference, Figure 1 also includes a point

[3] It is non-trivial for the proxy to know when to pad, as it does not know when the data stream has ended. One way for the proxy to know this is to set a parameter K, such that if the last K packets were dummy packets, then the traffic is determined to have ended. In our analysis we assume that the client and proxy know when to pad.

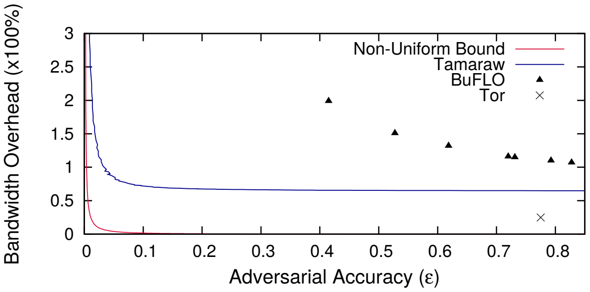

Figure 1: Non-uniform security (ϵ) against transmission size overhead for BuFLO, Tamaraw with $L = 100$, and Tor.

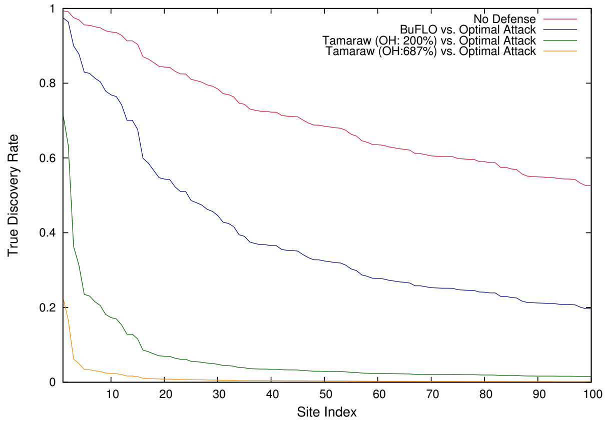

Figure 2: TDR for the Alexa top 100 sites in the open-world when using various defenses against the ideal attacker.

for Tor, for which we use overhead and security measurements reported by Cai, et al. against their Ca-DLevenshtein attack [3]. The results show that Tamaraw offers a significantly better security/efficiency trade-off than BuFLO. For reference, at a size overhead of 130%, there are 553 partitions (non-uniform security of 69%) in BuFLO ($\tau = 9$) and 18 partitions (non-uniform security of 2.25%) in Tamaraw. This shows that a design that adheres to the principles of provable lower bounds in Section 5.1 is more suitable for clients.

Table 1 shows how close different defenses are to the optimal lower bound curve derived in Section 5.1. The *Overhead Ratio* of a defense is the ratio between the defense's bandwidth overhead and the lower bound on overhead. Table 1 shows the best overhead ratios that Tamaraw and BuFLO achieved in our experiments. For reference, we also give the overhead ratios for Tor. Tamaraw achieves its best overhead ratio in a high-security/high-overhead configuration that may not be practical for most users. Therefore, we also report Tamaraw's performance in a more feasible configuration with overhead comparable to the best BuFLO configuration in our experiments. Even with this restriction, Tamaraw has an overhead ratio less than a sixtieth of BuFLO.

6.2.3 Open-world Performance

Figure 2 shows the TDR in the open-world for the Alexa top 100 sites when they face the ideal adversary and are defended by BuFLO, Tamaraw (at 200% and 687% overhead), or no defense, respectively. To compute these curves, we build an ideal closed-world classifier on the Alexa top 800 sites. The ideal attacker is based on the ambiguity sets described above. We then compute the TDR of the corresponding open-world classifier where the the website of interest is w_i, for $i = 1, \ldots, 100$. Note that, even though the open-world is the entire internet, our experiment only considers open-world attacks that attempt to recognize visits to one of the 100 most popular websites. The reason is because the TDR of an open-world attack on an unpopular website will be lower than that of an attack on a more popular website. By showing that the TDR becomes extremely low when attacking Tamaraw, even for the first 100 websites, we show that it's extremely low for all websites. The popularity of each website was taken from the Alexa *estimated page views per million* database [1]. We only need popularity information for the sites used to construct the closed-world classifier; the rest of the sites on the internet are treated as being randomly classified and have, in aggregate, $10^6 - (p_1 + p_2 + ... + p_{800})$ page-views per million, where p_i is the page-views-per-million of the i^{th} most popular site. For comparison, for the 100^{th} most popular site, in the open world, Tamaraw (with 200% overhead) has a TDR approximately $\frac{1}{13}^{th}$ the TDR of BuFLO and Tamaraw (with overhead 687%) has a TDR approximately $\frac{1}{110}^{th}$ the TDR of BuFLO. These results show that Tamaraw is a significant improvement over BuFLO in the open- and closed-world settings.

7. CODE AND DATA RELEASE

To ensure reproducibility and correctness, all code and data used in this paper are publicly available[4]. This includes: traces of websites loaded, code for all generators, code for all feature based attackers, and code for all defenses tested (including Tamaraw).

8. CONCLUSIONS

In this paper, we developed and tested a new feature-based comparative methodology that classifies and qualifies WF defenses. This methodology allows us to understand which defenses are able to successfully hide which features – thereby upper-bounding their success rates in defending against attackers that rely heavily on that feature. This methodology also exposes some flaws of previous defenses.

Our theoretical model clarifies the limits of website fingerprinting defenses. It establishes efficiency bounds that no defense can cross, giving an absolute benchmark for evaluating the efficiency of defenses. The lower bounds of bandwidth costs are surprisingly low, suggesting that it may be possible to build very efficient defenses. We also show that, in some contexts, randomized defenses offer no security or overhead advantage compared to deterministic defenses. This theoretical foundation also provides a framework for comparing schemes which offer different overhead and security trade-offs. Further, it allows conclusions to be drawn about open-world performance of attacks and defenses, based on their closed-world results. This greatly simplifies the experimental setup required to estimate open-world performance of attacks and defenses.

While previous work has shown that current WF defenses are either ineffective or inefficient, and while our work has explained these results using a systematic methodology, we argue that the situation is not hopeless for web browsing clients who desire privacy. We propose a new defense, Tamaraw, that is able to reduce the overhead of BuFLO significantly. Using our methodology, we show that Tamaraw provides better protection against website fingerprinting attacks than all previous defenses, in both, the open and closed-world models.

[4] `https://crysp.uwaterloo.ca/software/webfingerprint/`

9. ACKNOWLEDGMENTS

We would like to thank Scott E. Coull, Andriy Panchenko, and Kevin P. Dyer for their correspondence with us, which helped us improve the paper. We thank NSERC, ORF, and The Tor Project for funding this project. This work was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET: `www.sharcnet.ca`) and Compute/Calcul Canada.

10. REFERENCES

[1] Alexa — The Web Information Company. www.alexa.com.

[2] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Privacy Enhancing Technologies*, pages 1–11. Springer, 2006.

[3] X. Cai, X. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 605–616, 2012.

[4] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 191–206. IEEE, 2010.

[5] K. Dyer, S. Coull, T. Ristenpart, and 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*, pages 332–346, 2012.

[6] D. Herrmann, R. Wendolsky, and H. Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 31–42, 2009.

[7] A. J. Hoffman and J. B. Kruskal. Integral boundary points of convex polyhedra. In M. JÃijnger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 49–76. Springer Berlin Heidelberg, 2010.

[8] I. Keller and C. Tompkins. An Extension of a Theorem of Dantzig's. *Linear Inequalities and Related Systems, Annals of Mathematics Studies*, 38:247–254, 1956.

[9] M. Liberatore and B. Levine. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 255–263, 2006.

[10] L. Lu, E.-C. Chang, and M. C. Chan. Website Fingerprinting and Identification Using Ordered Feature Sequences. In *Computer Security–ESORICS 2010*, pages 199–214. Springer, 2010.

[11] X. Luo, P. Zhou, E. W. Chan, W. Lee, R. K. Chang, and R. Perdisci. HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. In *NDSS*, 2011.

[12] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Proceedings of the 10th ACM Workshop on Privacy in the Electronic Society*, pages 103–114, 2011.

[13] M. Perry. Experimental Defense for Website Traffic Fingerprinting. `https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting`, September 2011. Accessed Feb. 2013.

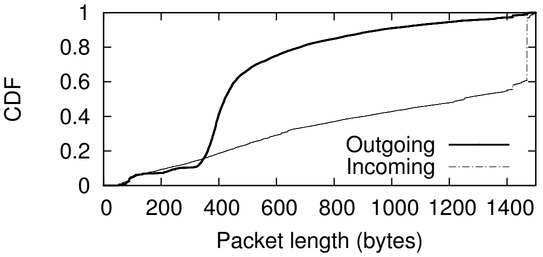

Figure 3: Packet lengths observed when loading one instance of each of Alexa's top 800 sites. Packets sized 1500 bytes are discarded.

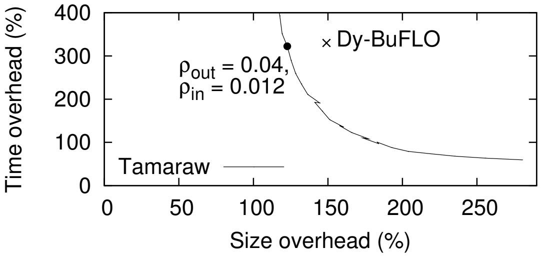

Figure 4: The lower-left boundary of the two-dimensional feasibility region of size and time overhead for Tamaraw when varying ρ_{out} and ρ_{in}. Our chosen parameters and the overhead of BuFLO on the same data set are marked. The overhead includes the padding mandated by $L = 100$.

[14] M. Perry. A critique of website fingerprinting attacks. `https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks`, November 2013.

[15] M. Perry, E. Clark, and S. Murdoch. The Design and Implementation of the Tor Browser [DRAFT]. `https://www.torproject.org/projects/torbrowser/design/`. Accessed Oct. 2013.

[16] P. Seymour. Decomposition of regular matroids. *Journal of Combinatorial Theory, Series B*, 28:305–359, 1980.

[17] T. Wang and I. Goldberg. Comparing website fingerprinting attacks and defenses. Technical Report 2013-30, CACR, 2013. `http://cacr.uwaterloo.ca/techreports/2013/cacr2013-30.pdf`.

[18] T. Wang and I. Goldberg. Improved Website Fingerprinting on Tor. In *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society*, 2013.

[19] C. Wright, S. Coull, and F. Monrose. Traffic Morphing: An Efficient Defense against Statistical Traffic Analysis. In *Proceedings of the 16th Network and Distributed Security Symposium*, pages 237–250, 2009.

APPENDIX

A. LOWER BOUND PROOFS

Suppose websites $w_1, \ldots, w_n$ have sizes $s_1 < s_2 < \ldots < s_n$. Let $S = \{s_1, \ldots, s_n\}$. For any defense, D, let p_{ij} be the probability that D transmits j bytes during a load of website w_i.

Since, in a closed-world experiment, each website occurs with probability $1/n$, the bandwidth cost of D is

$$\sum_{j=1}^{\infty} \sum_{i=1}^{n} \frac{1}{n} j p_{ij}$$

and the non-uniform success probability of A_S is

$$\sum_{j=1}^{\infty} \frac{\max_i p_{ij}}{\sum_i p_{ij}} \cdot \frac{\sum_i p_{ij}}{n} = \sum_{j=1}^{\infty} \frac{\max_i p_{ij}}{n}$$

We derive lower bounds on the bandwidth cost of D by computing the matrix of p_{ij} values that minimize the above bandwidth cost function while still satisfying the above security constraint. Recall that, since D is assumed not to compress or truncate web pages, $p_{ij} = 0$ for $j < s_i$.

The overall structure of the proof for non-uniform security is

- Constrain the structure of the optimal p_{ij} so that we can formulate the optimization problem as a linear program. (see Lemma 1).
- Prove that the linear program has an integral solution, so that the optimal solution is equivalent to a function $f : S \to S$ satisfying certain constraints (see Lemma 2).
- Prove that f is monotonically increasing (see Lemma 3).

The lower bound for uniform security is similar. We first prove that there exists a similar function f for any deterministic uniformly secure defense, and then apply Lemma 3.

LEMMA 1. *Let p_{ij} be the probabilities that minimize the bandwidth cost while meeting the security requirement.*

1. *$p_{ij} = 0$ unless $j \in \{s_1, \ldots, s_n\}$.*
2. *If $p_{ij} < \max_k p_{kj}$, then for all $j' > j$, $p_{ij'} = 0$.*
3. *For all j, $p_{kj} \leq p_{k+1,j}$ for $k \in [1, i]$, where $s_i \leq j < s_{i+1}$.*

PROOF. 1. Suppose $p_{\ell j} \neq 0$ where $s_k < j < s_{k+1}$. Then we can make a more efficient and no less secure by replacing p_{is_k} with $p_{is_k} + p_{ij}$ for all i and setting $p_{ij} = 0$ for all i. This will have lower bandwidth cost because $s_k < j$. This will not violate the constraint that, for all i and $j' < s_i$, $p_{ij'} = 0$, because, if $p_{ij} \neq 0$ before the change, then $s_i \leq j$, so $s_i \leq s_k$. This will not worsen security because $\max_i (p_{is_k} + p_{ij}) \leq \max_i p_{is_k} + \max_i p_{ij}$.

2. Suppose otherwise. Let $t = \min(\max_k p_{kj} - p_{ij}, p_{ij'})$. Note $t \neq 0$. Thus we can construct a more efficient and no less secure defense by replacing p_{ij} with $p_{ij} + t$ and $p_{ij'}$ with $p_{ij'} - t$.

3. Suppose $p_{kj} > p_{k+1,j}$ for some $k \in [1, i]$, where $s_i \leq j < s_{i+1}$. This implies that $p_{k+1,j} < \max_i p_{ij}$. By Item 2, $p_{k+1,j'} = 0$ for all $j' > j$.

   Thus we must have that: $\sum_{j'=s_{k+1}}^{j} p_{k+1,j'} = 1$.

   This also implies that $p_{kj} \neq 0$. Thus, by Item 2, $p_{kj'} = \max_i p_{ij'}$ for all $j' \in \{s_k, \ldots, j-1\}$. This implies that $\sum_{j'=s_k}^{j} p_{kj'} > \sum_{j'=s_{k+1}}^{j} p_{k+1,j'} = 1$, a contradicition.

Since p_{ij} is non-zero only if $j \in \{s_1, \ldots, s_n\}$, we can relabel the p_{ij} to be the probability that the defense transmits s_j bytes during a load of website w_i.

Lemma 1, Item 3 implies that $\max_i p_{ij} = p_{ii}$, so the security constraint can be re-written as $\sum_{i=1}^{n} p_{ii} \leq \epsilon n$.

Now that the security constraint is a linear function of the p_{ij} variables, we can formulate a linear program for computing the optimal p_{ij} values:

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij} s_j \qquad \text{(the bandwidth cost)}$$

subject to the constraints

$$\begin{array}{lll} (a) & \sum_{i=1}^{n} p_{ii} \leq \epsilon n & (\epsilon \text{ non-uniform security}) \\ (b) & \sum_{j=i}^{n} p_{ij} = 1 & (p_{ij} \text{ are probabilities}) \\ (c) & 0 \leq p_{ij} \leq 1 & \end{array}$$

LEMMA 2. *The above linear program has an integral solution.*

PROOF. Linear programs with Totally Unimodular (TU) constraint matrices and integral objective functions have integral solutions [7]. We prove that the constraint matrix, A (derived by the constraints (a), (b), and (c) of the above LP), is TU. To prove TU-ness of A, it is sufficient to prove the following [8]: (i) Every column contains at-most 2 non-zero entries, (ii) Every entry is 0, 1, or -1, (iii) If two non-zero entries in any column of A have the same sign, then the row of each belongs in two disjoint partitions of A.

Since the set of TU matrices is closed under the operation of adding a row or column with at-most one non-zero entry [16], we may delete the $2n$ rows of A corresponding to constraint (c) and prove that the remaining constraint matrix A' satisfies the TU conditions (i) - (iii).

Observe the following properties of A':

- There are n rows (WLOG, rows 1 to n) induced by the constraint (a). These are such that: $A_{i,(i-1)n}, \ldots, A_{i,in-1} = 1$, $\forall i \in \{1, \ldots, n\}$ and 0 for all other entries. Therefore, each column of the partition B composed of these n rows contains only a single non-zero entry (i.e., +1).
- There is only 1 row (WLOG, row $n+1$) induced by the constraint (b). This row has the form: $A_{n+1,j} = 1, \forall j \in \{1^2, \ldots, n^2\}$ and 0 for all other entries. Each column of the partition C composed of this single vector may contain only a single non-zero entry (i.e., +1).

From the above properties, it is clear that matrix A' is TU since: Each column contains at-most 2 non-zero entries (+1) and it may be partitioned into matrices B and C such that condition (iii) is satisfied. Therefore, the matrices A' and A are TU and the LP describing A has only integral optima.

In an integral solution of the linear program, all the probabilities are 0 or 1, so the solution is equivalent to a function $f : S \to S$ satisfying

- $|f(S)| \leq \epsilon n$.
- $\sum_{i=1}^{n} f(s_i) / \sum_{i=1}^{n} s_i \leq \text{BWRatio}_D(W)$.

We now show there is a similar function for any deterministic uniformly secure defense D. Set $f(s_i) = b_i$ where b_i is the number of bytes transmitted when the defense D loads website w_i. Since D does not compress or truncate websites, we must have $b_i \geq \max_{s \in f^{-1}(b_i)} s$ for all i. Observe that we can assume $b_i = \max_{s \in f^{-1}(b_i)} s$ without harming security or efficiency, so that $f : S \to S$. Thus f satisfies the security constraint $\min_i |f^{-1}(s_i)| \geq 1/\epsilon$, and $\sum_{i=1}^{n} f(s_i) / \sum_{i=1}^{n} s_i \leq \text{BWRatio}_D(W)$.

LEMMA 3. *The mapping function f corresponding to an optimal non-uniformly ϵ-secure defense, or a deterministic uniformly ϵ-secure defense, is monotonic.*

PROOF. Consider any partition of $\{s_1, \ldots, s_n\}$ into sets S_1, $\ldots$, S_k. Let $m_i = \max_{s \in S_i} S_i$. Without loss of generality, assume $m_1 \leq m_2 \leq \cdots \leq m_k$. Now consider the monotonic allocation of traces into sets $S_1^*, \ldots, S_k^*$ where $|S_i^*| = |S_i|$. Let $m_i^* = \max_{s \in S_i^*} s$. Observe that $m_i^* \leq m_i$ for all i, i.e. the new allocation has lower bandwidth.

Since the number of sets in the partition and the sizes of those sets are unchanged, this new allocation has the same uniform and non-uniform security as the original, but lower bandwidth. Hence the optimal f must be monotonic.