Website Fingerprinting and Identification Using Ordered Feature Sequences

Liming Lu, Ee-Chien Chang, and Mun Choon Chan*

Department of Computer Science, School of Computing
National University of Singapore
{luliming,changec,chanmc}@comp.nus.edu.sg

Abstract. We consider website fingerprinting over encrypted and proxied channel. It has been shown that information on packet sizes is sufficient to achieve good identification accuracy. Recently, traffic morphing [1] was proposed to thwart website fingerprinting by changing the packet size distribution so as to mimic some other website, while minimizing bandwidth overhead. In this paper, we point out that packet ordering information, though noisy, can be utilized to enhance website fingerprinting. In addition, traces of the ordering information remain even under traffic morphing and they can be extracted for identification. When web access is performed over OpenSSH and 2000 profiled websites, the identification accuracy of our scheme reaches 81%, which is 11% better than Liberatore and Levine's scheme presented in CCS'06 [2]. We are able to identify 78% of the morphed traffic among 2000 websites while Liberatore and Levine's scheme identifies only 52%. Our analysis suggests that an effective countermeasure to website fingerprinting should not only hide the packet size distribution, but also aggressively remove the ordering information.

Keywords: Traffic analysis, side channel attack, edit distance, privacy, anonymity.

1 Introduction

Website fingerprinting aims to identify the website accessed in some low latency, encrypted tunnel. The "fingerprint" of a website is typically profiled on side channel features observed from the stream of packets sent and received to access the website. Let us call such stream the HTTP stream. From the perspective of web clients, website fingerprinting raises the privacy concern and indicates the need for further anonymization. On the other hand, such technique aids legitimate wardens track web accesses over encrypted channel.

Since neither the IP address nor domain name of the website is available due to encryption and proxies, it is natural to fingerprint websites using packet size related features [2] [3] [4]. Simple defenses to website fingerprinting have been proposed in previous works, including variations of packet padding. However,

* This work is partially supported by Grant R-252-000-413-232/422/592 from TDSI.

the large bandwidth overhead they cost leads to insufficient incentives for deployment. “Traffic morphing” [1] is designed as a scheme to evade detection by warden, and at the same time, incurs small bandwidth overhead. Traffic morphing alters the packet size distribution of an HTTP stream, to make it appear as if it is from another website. A morphing matrix is pre-computed to transform a source distribution to the target, minimizing the total increase in packet sizes.

In this paper, we propose a website fingerprinting scheme that exploits information on packet ordering as well as on packet sizes. Our scheme is motivated by an observation on the regularity of browsers’ behavior in retrieving a webpage through HTTP, which leads to a robust ordering of certain packets transmitted. Experimental results show that sufficient ordering information is retained for accurate website fingerprinting under reasonable amount of noise.

Our scheme is evaluated on a classification problem that identifies an HTTP stream among a list of profiled websites, as well as a detection problem in which it is uncertain whether the test website belongs to the profiled set. We tested our scheme on datasets containing up to 2000 websites, with traces collected over two months. For the classification problem, we test our scheme with full-length OpenVPN test streams, i.e., 30-second long encrypted HTTP streams tunneled through a VPN server, and the fingerprint identification accuracy is 97% among 1000 websites. We also test with partial SSH traces that capture only the first few seconds of encrypted communication, the accuracy is 81% among 2000 websites, still over 10% better than the previous scheme. For the detection problem, our scheme presents an equal error rate at 7%, which is significantly better than the previous scheme at 20%. Hence, for both types of evaluations, our scheme yields superior identification accuracy compared to previous schemes.

Furthermore, our scheme is resilient to the traffic morphing technique by Wright *et al.* [1]. When traffic is morphed according to unigram (i.e. single packet size) distribution, our scheme distinguishes the morphed traffic from its target with a success rate of 99% versus 25% for the previous scheme. When traffic is morphed according to bigram (i.e. two adjacent packet sizes) distribution, our scheme distinguishes 98.7% versus 22.5% for the previous scheme. When bigram morphed traffic are mixed with traces of 2000 other websites, our scheme correctly identifies 78% of the morphed traffic, while the previous scheme identifies 52%. Of course, our scheme may not improve the fingerprinting accuracy if packets are padded or randomly morphed without concern of the bandwidth overhead. For example, padding all packets to the path maximum transmission unit will remove the ordering information, thus no improvement will be made.

In addition, we analyze the consistency of our website fingerprints, with respect to static and dynamic websites, and different HTTP pipelining configurations. We find that only 6% of the websites need reprofiling after a month. The results verify that our feature selection generates consistent fingerprints.

Our analysis suggests that both size and ordering features should be removed in countermeasures to website fingerprinting. We propose some countermeasures that engender randomization in packets or HTTP requests. The countermeasures are low in bandwidth overhead, though they may trade off the response time.

Here we summarize the main contributions of our paper. (*i*) We provide a concrete and efficient scheme that utilizes packet ordering information for website fingerprinting. It improves the identification accuracy over previous schemes. (*ii*) The website fingerprinting scheme we propose is able to withstand traffic morphing that considers little packet ordering information.

The rest of this paper is organized as follows. The related work is reviewed in Sect. 2. Our website fingerprinting model is presented in Sect. 3. Our method to handle traffic morphing is described in Sect. 4. Experiments and analysis on the effectiveness and robustness of our scheme are presented in Sect. 5. We suggest some countermeasures in Sect. 6 and conclude this paper in Sect. 7.

2 Related Work

Side channel information has been utilized in a wide range of traffic analysis applications. For example, keystroke intervals are used for password guessing over SSH [5]; the sequence of bit rates is used to identify the streaming video encoded with variable bit rate schemes [6]; packet sizes and rates are used to reveal the presence of Skype traffic [7]; packet size, timing and direction are used to infer the application layer protocol [8].

Several recent works [2] [3] [4] [9] have looked at the issue of using traffic analysis to identify websites accessed. Yet some of their assumptions have been invalidated with changes in HTTP and web browser behaviors. For example, browsers were assumed not to reuse a TCP connection to download multiple embedded objects [3] and object requests were assumed non-pipelined [4]. Hence their approaches to determine an object size by accumulating the data received either through a TCP connection [3] or between adjacent HTTP requests [4] no longer work. Liberatore *et al.* [2] used the set composed of (direction, packet size) pairs as fingerprint, and used Jaccard's coefficient as the similarity measure ($\frac{|X \cap Y|}{|X \cup Y|}$, where X and Y are the sets representing a website profile and a test fingerprint, and $|A|$ denotes the size of set A). These schemes only exploit side channel features related to sizes but not ordering.

Defences against website fingerprinting have been proposed, such as variants of packet padding. Padding every packet to the size of path Maximum Transmission Unit (MTU) can thwart size related traffic analysis, but the amount of overhead it causes is nearly 150% of the actual data [2]. "Mice-elephant" packet padding incurs relatively less overhead, at nearly 50% growth in the data transmitted. It pads packets to two sizes, either a small size for control packets, or to the path MTU for data packets. Wright *et al.* [1] proposed traffic morphing as a bandwidth efficient alternative to packet padding. It transforms the source packet size distribution to mimic a target website, by splitting or padding the packets. With limited consideration on packet ordering, the morphing technique targets at fingerprinting schemes that only exploit information on packet sizes.

Edit distance [10] is employed in our scheme to measure the similarity of fingerprints. It computes the minimum total cost of insertion, deletion and substitution to make two sequences identical. Edit distance has been applied in a

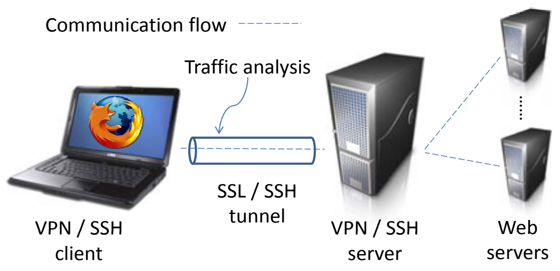


Fig. 1. An illustration of traffic analysis setup

wide range of applications, such as spell checker in Google [11] and in Microsoft Word [12], identifying plagiarism, comparing DNA sequences [13] [14], conducting fuzzy search in EXCEL [15] and evaluating dialect distances [16]. Our work is the first in applying it to match the features of network traffic.

3 Our Website Fingerprinting Scheme

Web contents are retrieved from the server using HTTP and presented by the client browser. SSH, SSL or its successor TLS provides a layer of protection to data secrecy by encrypting the HTTP traffic before transmitting them over TCP/IP. An SSH or SSL/TLS server acts as a proxy, providing a tunnel such that the web server address is encapsulated into the packet payload and encrypted. Website fingerprinting is performed over the encrypted and tunneled HTTP stream.

3.1 Model

We use passive traffic analysis to fingerprint websites. Two important observations enable us to adopt this model: (*i*) webpage download by HTTP is highly structured; and (*ii*) encryption and proxy do not severely alter the packet sizes, nor the packet ordering. The advantage of passive traffic analysis is that the presence of a warden is completely transparent.

The HTTP streams for analysis are captured over the protected tunnel between client and the VPN or SSH server, as shown in Fig. 1. Website fingerprints are extracted from the HTTP streams. Several fingerprint instances form the profile of a website. A testing stream is compared to each profile for identification.

Traffic model. Our fingerprinting model supports most browser configurations and accommodates the following connection patterns (*i*) multiple TCP connections can be opened to download in parallel different embedded objects, (*ii*) each TCP connection can be reused to download multiple objects, (*iii*) HTTP requests can be pipelined in each connection, and (*iv*) all sessions are multiplexed onto a single port.

The connection patterns speed up the webpage download, but they increase the difficulty to fingerprint websites. HTTP pipelining means a client is allowed to make multiple requests without waiting for each response. Because of multiple TCP connections and HTTP pipelining, data packets of different embedded objects can interleave, thus object sizes cannot be determined by accumulating the amount of data received between consecutive HTTP requests. Multiplexing communication sessions hides the number of TCP connections opened and the number of objects in a webpage.

As in previous works [2] [3] [4] [9], we assume browser caching is disabled. If it is enabled, we can determine if a user has accessed some sensitive website using the attack in [17].[1]

Problem scenarios. We tackle two problem scenarios, classification and detection. In both scenarios, a set D of, say 1000, websites are profiled.

– **Classification.** Given a test stream which is known to be a visit to a website in D, identify the website. This is the same problem addressed in previous work.
– **Detection.** Given a test stream, determine whether it is a visit to a website in D, and identify the website if it is. We examine the false positive rate (FPR) and false negative rate (FNR) in identification. This problem has not been addressed in previous work.

The classification scenario requires the fingerprints of a website be sufficiently similar. While the detection scenario further requires fingerprints of different websites be sufficiently dissimilar.

Noise model. There are a few sources of noise that degrades the consistency of HTTP streams, even if the website contents have not changed. Connection multiplexing and HTTP pipelining are the main sources affecting the ordering of objects. In addition, the order of object requests may be browser specific, and the dynamics in network condition causes some random noise.

3.2 Fingerprint Feature Selection

A straightforward approach to represent fingerprint of an HTTP stream is by using the sequence of all packet sizes and directions, i.e. fingerprint $F = \langle(s_1, d_1), (s_2, d_2), ...\rangle$ where s_i is i-th packet size, and d_i is its direction. Clearly, the approach makes fingerprint comparison inefficient, since each sequence easily contains over a thousand elements. Yet more importantly, the identification accuracy will be affected, because the ordering of some packets often changes due to various noises. Hence, we apply domain knowledge to select features.

Firstly, packets with sizes smaller than a threshold are considered control packets and discarded from fingerprints. Secondly, we keep only the non-MTU

[1] User is issued requests on some of the website's objects. From the response time, we determine if the objects are cached and hence if the user has visited the website.

(Maximum Transmission Unit) downloading packets as features of HTTP responses. The reasons why we exclude the MTU downloading packets are explained below.

Ideally, we would like to monitor object sizes rather than packet sizes, as objects are less variable and more characteristic to a website. However, data of different objects interleave in transmission due to multiple TCP connections and HTTP pipelining. It is difficult to associate the data blocks to their respective objects. Fortunately, HTTP transmission is not random. Majority of web servers transfer data in chunks. In each data chunk, all packets are sized as path MTU, except the last packet transferring the remaining data. Therefore, although we cannot estimate object sizes, we can leverage the last packet size of a data chunk, as it is specific to an object component. As for packet ordering, packets that change their order tend to be the intermediate packets of different objects. By filtering the intermediate packets, we reduce the probability of fingerprint inconsistency.

Thirdly, we design the fingerprint of an HTTP stream to be two sequences, representing the request and response features respectively. Although a request must precede its corresponding response, the order of other requests to this response is not deterministic, but sensitive to the pipelining configuration. Hence the request and response features are not merged into a single sequence.

Therefore, to fingerprint a website over encrypted tunnels, the side channel features we select are (*i*) **sequence of HTTP request sizes** and (*ii*) **sequence of HTTP response sizes** (except MTU packets). Note that *the number of objects* and *the number of components in object responses* are two implicit features represented by the lengths of these two sequences.

The features we select closely adhere to the webpage content and layout. The request sequence reveals the relative locations of embedded objects in a webpage and their URL lengths. The response sequence indicates the download completion order of object components and the sizes of their last packets.

3.3 Fingerprint Similarity Measurement

Edit distance measures the number of edit operations to make two strings identical. *Levenshtein distance* is a form of edit distance that allows insertion, deletion or substitution of a single character. It is commonly computed using Wagner-Fischer's algorithm [18]. We use Levenshtein distance to measure the similarity between two website fingerprints.

We normalize the edit distance by $\max(|x|, |y|)$, where $|x|$ denotes the length of sequence x. We define $similarity = 1 - distance$, to convert the distance measurement into similarity. Given two fingerprints, two similarity values measured on the object request sequences and on the non-MTU response sequences are combined as follows:

$$\alpha \cdot sim_{HTTPget} + (1 - \alpha) \cdot sim_{nonMTUpkts},$$

where α is a tunable parameter, indicating the degree of reliability of the similarity values. We set $\alpha = 0.6$ in our experiments, as the sequence of object requests is more stable. The probability of a test stream coming from a particular website is determined as the maximum similarity between the fingerprint of this stream and the various fingerprints in the website profile.

Edit distance is chosen as our similarity measure because (*i*) fingerprint instances vary as a result of network dynamics and webpage updates, whose effects are shown as insertion, deletion or substitution in the sequences, (*ii*) it considers the order information, which differs from Jaccard's coefficient that treats the feature values as a set, and (*iii*) the length information of fingerprint sequences are encapsulated for evaluation.

Example. Given two fingerprints $F_A = (A_{req}, A_{res})$ and $F_B = (B_{req}, B_{res})$, where $A_{req} = \langle 436, 516, 500, 532\rangle$, $A_{res} = \langle 548, 628, 1348, 1188, 596, 372, 980\rangle$, $B_{req} = \langle 436, 516, 500, 253, 500\rangle$ and $B_{res} = \langle 548, 628, 1348, 1188, 436, 412, 1268\rangle$. The normalized edit distance between the request sequences A_{req} and B_{req} is $2/5$. The distance between the responses A_{res} and B_{res} is $3/7$. Hence the similarity between F_A and F_B is 0.59 when $\alpha = 0.6$.

4 Website Fingerprinting under Traffic Morphing

Traffic morphing [1] aims to provide a bandwidth efficient countermeasure to website fingerprinting. Unigram morphing changes the distribution of single packet size, while bigram morphing changes the distribution of two consecutive packet sizes, to make the traffic appear as if from some other website. The problem of finding the morphing matrix to transform packet size distributions is formulated as an optimization problem, in which the goal is to minimize the bandwidth overhead, and the constraints are the source and target packet size distributions.

An important objective of traffic morphing is bandwidth efficiency. However, if traffic morphing is extended to n-gram ($n \geq 2$) to consider more packet ordering information, it becomes increasingly bandwidth inefficient. The reason is that higher-gram morphing has to concurrently satisfy all the lower-gram distributions, so there are very limited choices for the size of the n-th packet once the previous $n-1$ packet sizes are fixed. Source packet sizes have to map to the target distribution even though the size differences are large. For some value of n, its bandwidth efficiency drops below packet padding schemes.

4.1 Fingerprint Differentiation

Our scheme can differentiate website fingerprints under traffic morphing [1]. For the ease of discussion, let us call the source website W_S, and its morphed traffic S'. Website W_S mimics a target website W_T. The traffic of W_T is T. From S', we want to identify W_S.

We differentiate W_S and W_T fingerprints by packet ordering. Assume that we know website W_S morphs its traffic, while website W_T does not. A large edit distance between T and test stream S' indicates that S' is not from W_T, because the fingerprints of W_T should be fairly consistent without morphing. Hence, test stream S' is from website W_S.

If there are k websites imitating the same target W_T, source W_S is likely uniquely identifiable. The reason roots at the morphing constraint to minimize bandwidth overhead. A morphing matrix maps each source packet to some minimally different sizes in the target distribution. Because of the correlation in packet sizes before and after morphing, and the consistency in the ordering of unmorphed packets, morphed traffic also have reasonably consistent order. It leaves us a loophole to differentiate among the k websites.

In evaluations, we examine the identifiability of k morphed websites amid other websites not related by morphing. We preprocess to narrow down the candidate websites by packet size distributions, e.g. using $L1$ distance measurement.[2] On one hand, the preprocessing safeguards that the morphing websites are not filtered prematurely. Morphed streams tend to have larger variations in packet ordering, because a source packet can map to a few target sizes and the choice is probabilistic for each morphing instance. On the other hand, preprocessing eliminates noise websites that the test stream may incidentally share similar size sequence with, but are dissimilar in size distributions.

Since we evaluate the similarity of fingerprints based on the whole sequence of feature values, our scheme can identify a morphed traffic, as long as morphing does not handle the distribution of n-gram for n close to the sequence length.

5 Evaluation

We present the performance analysis of our scheme in this section. Experiment settings and data collection are described in Sect. 5.1. We evaluate the fingerprinting scheme in three aspects: identification accuracies (Sect. 5.2), robustness to traffic morphing (Sect. 5.3) and consistency of fingerprints (Sect. 5.4).

5.1 Experiment Setup and Data Collection

We prepared three datasets on OpenVPN for evaluation, namely, *static50, dynamic150*, and *OpenVPN1000.* The dataset *static50* contains trace data of 50 websites that are rarely updated (less than once in a few months); the dataset *dynamic150* contains traces of 150 websites that are frequently update (daily); and the dataset *OpenVPN1000* contains traces of 1000 popular websites.

To demonstrate that our scheme also works on SSH, we use the publicly available dataset *OpenSSH2000* [2], which contains traces of 2000 most visited websites accessed through OpenSSH, captured once every 6 hours for 2 months. The trace captures only packet headers in the first 6 seconds of a webpage download.

[2] $L1$ distance between two website fingerprints is computed as the sum of absolute differences in their packet size distributions.

Table 1. Fingerprint identification accuracy for various datasets

Dataset	Static50	Dynamic150	OpenVPN1000
Accuracy	99%	97%	97%

We prepare the OpenVPN datasets by visiting the websites[3] and capturing the packet traces daily for 3 months. The traces are captured using TCPDump [19] between the client and the VPN server. We use Firefox as the browser, HTTP pipelining is enabled and its default value is 4 (at most 4 HTTP requests can be served concurrently). Cache is cleared after each access. Flash player is installed. Each web access is monitored for 30 seconds, to allow operations of object requests and responses to fully complete.

The topology, size and content of high level cache is not within our control for evaluation. In our experiments, the profiled websites are accessed regularly, so we expect most objects are consistently cached close to the VPN or SSH server.

We use Liberatore and Levine's scheme [2] for performance comparison, and refer to it as the *reference scheme*. Our scheme that performs preprocessing is referred to as the *improved scheme*, otherwise it is called the *basic scheme* or simply our scheme. For each website, 15 traces are used to generate the fingerprint profile and 10 other traces are used for testing. Analyzing from the trace data and protocol specifications, we use 300 and 1450 as thresholds on packet sizes to upper bound control packets and to lower bound MTU packets respectively. Thus the fingerprint sequence of object requests contains packet sizes that are larger than 300 bytes in the uploading direction, and the fingerprint sequence of non-MTU responses consists of packet sizes between 300 to 1450 bytes in the downloading direction. These settings are used in our experiments, unless otherwise specified.

5.2 Fingerprint Identification Accuracy

Accuracy of classification scenario. The fingerprint identification accuracies on the OpenVPN datasets *static50*, *dynamic150* and *OpenVPN1000* are shown in Table 1. Note that for all three datasets, the accuracy is close to 100%.

To demonstrate that our scheme works well for identifying websites accessed in other tunnel, we tested it on the dataset *OpenSSH2000*, with parameters fully complying with the settings in [2]. The accuracy of our scheme is 81%, outperforming the reference scheme by 11%. The improvement is attained as our scheme differentiates websites that have similar sets of packet sizes but different packet ordering.

Accuracy of detection scenario. It is unknown in this scenario whether a test stream is from any profiled websites. Thus a test stream not from the profiled websites may be identified wrongly as from one of the websites in the profile

[3] We are fingerprinting a website by its homepage. If the internal pages of a website are also profiled, we can use the fingerprints of individual pages and their linkage information to identify more accurately and confidently the website being surfed.

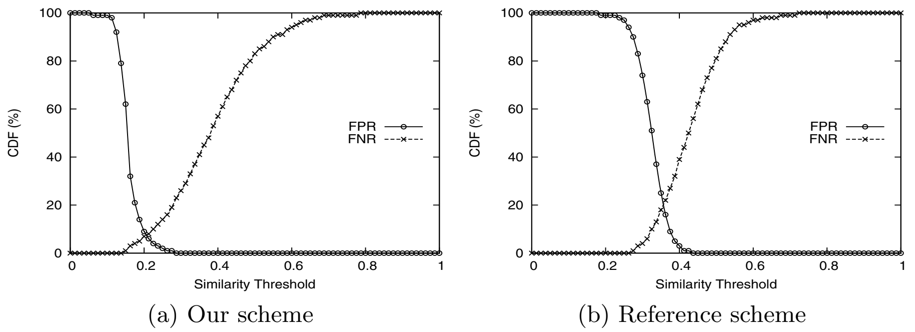


Fig. 2. False positive and false negative rates with respect to similarity thresholds

(false positive); and a test from the profiled websites may be identified wrongly as not from the profiled websites (false negative). We evaluate the fingerprint identification accuracy in terms of false positive rate (FPR) and false negative rate (FNR) for the detection scenario.

From the dataset *OpenVPN1000*, we randomly pick 200 websites to profile. Traces from both these 200 websites and the remaining 800 are used to test the fingerprint identification. The accuracy of our scheme with respect to different similarity thresholds is shown in Fig. 2a, and the performance of the reference scheme is shown in Fig. 2b. The equal error rate (EER) of our scheme occurs when the similarity threshold is 0.21, at which both FPR and FNR are 7%; while for the reference scheme, at the similarity threshold of 0.36, EER occurs to be 20%, almost three times of ours. If we minimize the total error rate, i.e. sum of FPR and FNR, the optimal similarity threshold for our method is 0.22, with only 6% FPR and 8% FNR; whereas for the reference scheme, the optimal similarity threshold is 0.37, at which its FPR and FNR are 9% and 27% respectively. The results show that our scheme is much stronger at differentiating websites.

5.3 Accuracy with Traffic Morphing

In this subsection, we empirically show that our scheme can differentiate fingerprints of websites morphed by packet sizes, and we verify that there is significant bandwidth overhead for n-gram ($n \geq 2$) morphing.

5.3.1 Fingerprint Differentiation under Traffic Morphing

Differentiating morphed traffic. We take 2000 websites as the mimicked targets, and generate the morphed traffic such that the packet size distributions are within $L1$ distance of 0.3 between a morphed traffic and its target. The same $L1$ distance threshold is used in the traffic morphing scheme [1]. For each target website, we generate 4 variants of the morphed distributions, and draw 5 instances from each variant. The distance in our scheme is measured using edit distance; while that in the reference scheme [2] is $1 - S$, where S is Jaccard's

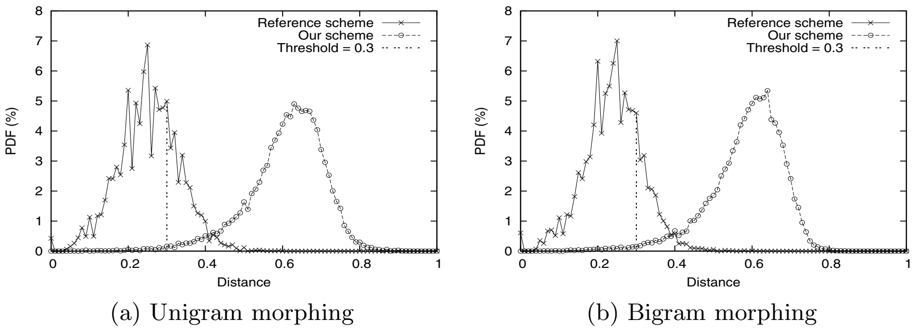


Fig. 3. Distance distribution of the morphed traffic and the mimicked target

coefficient. The distribution of distances between morphed traffic and their targets is shown in Fig. 3.

For unigram morphing as shown in Fig. 3a, compared to the reference scheme, our scheme shifts rightwards the range of distances from 95% in the interval [0.1, 0.4] to 95% in [0.4, 0.8]. The larger distance indicates that our scheme differentiates more clearly the morphed traffic and the target. The vertical bar at 0.3 indicates the distance threshold, and a test returning a value below the threshold means that the distance evaluation cannot separate the morphed instance and the mimicked target. For the reference scheme, 75% of the tests fail; while for our method, there is only 1.0% error cases, i.e., our scheme differentiates 99% of the morphing traffic. As shown in Fig. 3b, the experiment on bigram morphing yields similar results. For the reference scheme, 78% of the tests fail; while for our scheme, there is only 1.3% error cases. The experiment result shows that although traffic morphing is effective against the reference scheme, our scheme is highly resistant to it.

Identifying multiple websites morphed to the same target. We extend the evaluation to multiple websites morphing to the same target, and compare the distinguishability using our scheme, the reference scheme and random guess. We take 20 disjoint sets of k websites, where the first $k-1$ websites bigram morph towards the k-th website. Each morphing website generates 6 instances, 3 as learning samples and the rest 3 as test cases. We vary k from 2 to 6, and measure the identification accuracy of morphed traces in each set. The average identification accuracy of all sets at each k is presented in Fig. 4.

When our scheme differentiates between $k = 2$ websites, i.e. the source and target of morphing, it yields a high identification accuracy of 85%. As k increases, more websites share the same morphing target, the accuracy gradually decreases. When $k = 6$ candidates, which means five source websites morph to one target, our scheme has an accuracy of 50%. In contrast, accuracy of the reference scheme is 29% when $k = 6$, while random guess gives a probability of 17%. The reference scheme performs not much better than random guess. It

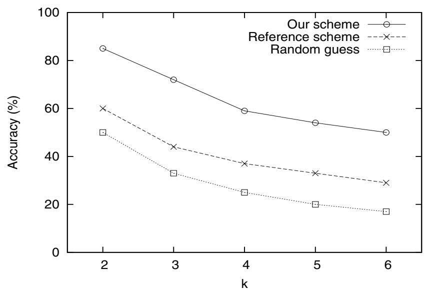


Fig. 4. Identification accuracy of k websites that morph to the same target

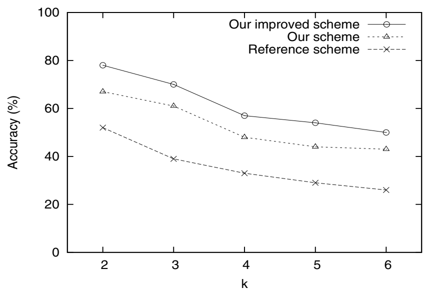


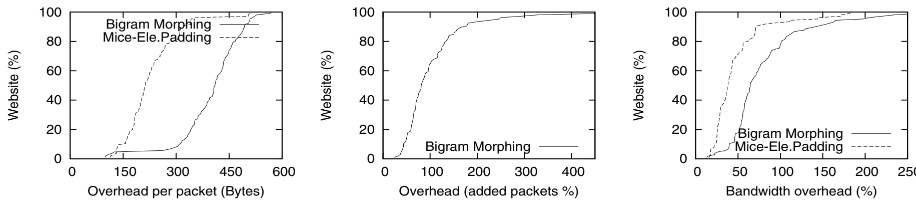
Fig. 5. Identification accuracy of k morphed websites among 2000 other websites

shows that the reference scheme cannot reliably distinguish websites that share similar packet size distributions. Our scheme is much stronger at differentiating among multiple morphing websites and their mimicked target.

Identifying morphed traffic mixed with other website profiles. Previous experiments examine the distinguishability within morphing sources and their target. Next we evaluate the identifiability of 20 sets of k morphing websites when they are mixed with 2000 other websites. Morphing websites are profiled based on their morphed traces. We set the $L1$ distance threshold to be 0.4 in screening websites by packet size distributions. The threshold is slightly larger than in morphing matrix computation to accommodate some incompliance between the computed and generated packet size distributions. The incompliance is caused by packet splitting which generates new packets that are not accounted for in the computation of morphing matrix. The identification accuracies are shown in Fig. 5.

Comparing Fig. 5 and Fig. 4, our improved scheme performs equally well when it identifies k morphing websites among a large set of unrelated websites and when it differentiates within the morphing websites. When k varies from 2 to 6, our improved scheme has an identification accuracy ranges from 78% to 50%, while the reference scheme has a corresponding range of 52% to 26%. The performance of our basic scheme in Fig. 5 degrades compared to in Fig. 4, because some morphed traces incidentally map to websites unrelated by morphing. However, it still outperforms the reference scheme. Our basic scheme correctly identifies 61% fingerprints of the morphing websites at $k = 3$, while the reference scheme identifies 39%.

We run our improved scheme on 2000 non-morphing websites. It gives the same identification accuracy of 81% as the basic scheme in this case. It is compatible to website fingerprinting at the absence of morphing, since non-morphing websites have HTTP streams that are consistent in both packet size distribution and ordering.



(a) Extra bytes per packet (b) Added packets (c) Bandwidth overhead

Fig. 6. Bandwidth overhead of bigram morphing and mice-elephant packet padding

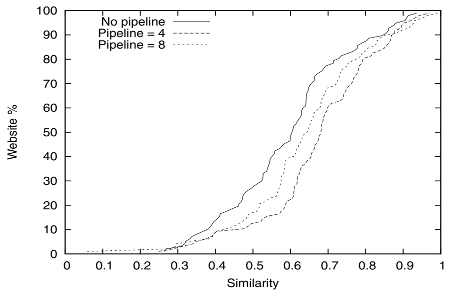
5.3.2 Bandwidth Overhead of n-Gram ($n \geq 2$) Morphing

N-gram ($n = 1, 2, 3$) morphing has been evaluated on several applications in [1]. Yet for website fingerprinting, its bandwidth overhead was presented only for unigram morphing ($n = 1$). The bandwidth overhead was compared to padding packets to the path MTU. In this paper, we evaluate the bandwidth overhead of bigram ($n = 2$) morphing. The overhead is compared to mice-elephant packet padding, which is another simple and effective padding scheme against website fingerprinting, but less constraining on the network bandwidth. We use traces of 100 websites from the *SSH2000* dataset. The i-th website is morphed to the $(i + 1)$-th website, where each HTTP stream is morphed 5 times. The average overhead is presented in Fig. 6.

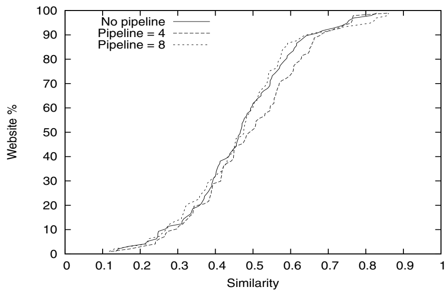
Fig. 6a shows the average added bytes per packet by morphing and mice-elephant. Using mice-elephant, 80% websites have 150 to 300 bytes of overhead per packet; but using traffic morphing, about 90% websites need to send 300 to 500 extra bytes for each packet, which is much higher than mice-elephant. Fig. 6b shows the percentage of added packets caused by packet splitting during morphing. Transmission of extra packets incurs overhead in bandwidth and delay. For some website, more than 4 times of packets are added. Fig. 6c shows the overall bandwidth overhead. Using bigram morphing, 20% websites have an overhead that exceeds 100% and can be 300% of the total source packet sizes. While using mice-elephant packet padding, 75% websites have the bandwidth overhead less than 50% of the total source packet sizes, and only 5% websites have bandwidth overhead more than the total source packet sizes. As n increases, the bandwidth overhead of n-gram morphing also increases. In summary, our experiment shows that n-gram morphing is less bandwidth efficient than mice-elephant packet padding even when $n = 2$.

5.4 Consistency of Fingerprints

Fingerprint consistency can be influenced by browser pipeline configurations, web content updates and network dynamics which cause packet loss, reordering and retransmission. Note that our fingerprints are not affected by the randomness in the amount of control packets, as we have excluded control packets from website fingerprints. In practice, packet loss, reordering and retransmission are



(a) Sequence of HTTP requests



(b) Sequence of non-MTU responses

Fig. 7. Effect of pipelining on the fingerprint sequences

not severe. In this subsection, we empirically show that the effect of pipelining is not serious on our fingerprints, and our fingerprints need not be updated frequently for most websites.

Effect of HTTP pipelining. To illustrate the effect of HTTP pipelining on the fingerprint sequences, we vary the pipeline degree from 1 (non-pipelined) to 8 (the maximum supported by Firefox). We capture OpenVPN traces of 100 websites. For each website, fingerprints from 15 traces with varying pipeline degrees are kept as profile, and 15 test streams of pipeline degrees 1, 4 and 8 are compared with their own website profile.

Fig. 7 shows the fingerprint sequences of a website have high similarities. 70% websites have similarity in the object request sequences higher than 0.5, and 50% of the non-MTU response sequences have similarities larger than 0.5. The figure also shows that the distributions of a website's fingerprint similarities at different pipeline degrees are similar. HTTP pipelining does not drastically affect our fingerprint sequences.

Effect of website update. To evaluate how well the profiles represent websites over time, we measure the fingerprint identification accuracy weeks after constructing the profiles. The evaluation helps to decide the update frequency of website profiles. We perform an experiment on the datasets *OpenVPN1000* and *dynamic150* to study the impact of (frequent) content updates on the fingerprint identification. For each website, we randomly pick 5 traces captured within week 0 to generate the website profile, then the traces of 4 randomly chosen days in each of week 1, 2, 3, 4 and 8, are tested against the website profiles.

The results are shown in Fig. 8. For dataset *OpenVPN1000*, the identification accuracy gradually decreases over time, which remains high at 94% a month later and becomes 88% after two months. Profiles of most websites need not be regenerated after 2 months. Only 6% most frequently updated websites need to update their fingerprint profiles in the first month, and another 6% in the second month. For the dataset *dynamic150*, the decrease in identification accuracy

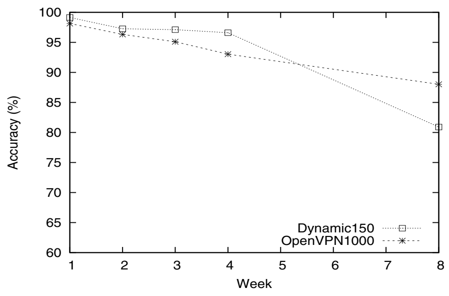


Fig. 8. Effect of time on accuracy

from week 4 to week 8 is obvious, because their website contents are frequently updated, and the changes accumulate over time. Nevertheless, its fingerprint identification accuracy is 96% one month later, and 81% two months later.

6 Countermeasures

We propose some countermeasures that randomizes the size and timing channels. These countermeasures incurs almost zero bandwidth overhead. (*i*) *Randomizing the object order.* We can use a browser plug-in to muddle the object request order. With the plug-in, browser buffers x HTTP requests, and sends them in a random order, where x can be a random number dynamically generated at each web access. (*ii*) *Randomizing the packet sizes.* To make the size and order of packets untraceable, browser and server can generate a sequence of random numbers for each access, and buffers the data to send in packets of the random sizes. Random delay can be added to conceal which packet sizes have been touched up in the HTTP stream. Although data buffering causes delay, the countermeasures significantly increase the difficulty of website fingerprinting.

7 Conclusion

In this paper, we developed a robust website fingerprinting scheme over low latency encrypted tunnels. We make use of the seemingly noisy packet ordering information rather than just the distribution of packet sizes as in previous work. The ordering of a selection of packets is consistent due to the behaviorial characteristics of protocols and browsers, and such information is preserved and exposed regardless of proxy and encryption. Our scheme identifies websites with high accuracy on both SSH and SSL tunnels, even for websites with dynamic contents. Furthermore, our scheme is designed to withstand traffic morphing [1]. Traffic morphing cannot handle packet ordering while satisfying low bandwidth overhead. Its bandwidth efficiency is worse than mice-elephant packet padding, even in bigram morphing. Our scheme is able to identify websites from morphed traffic with significantly higher accuracy than the previous scheme. In future work, we would analyze performances of the countermeasures in both their effectiveness and overheads.

References

1. Wright, C.V., Coull, S.E., Monrose, F.: Traffic morphing: An Efficient Defense against Statistical Traffic Analysis. In: 16th NDSS (2009)
2. Liberatore, M., Levine, B.N.: Inferring the Source of Encrypted HTTP connections. In: 13th ACM CCS, pp. 255–263 (2006)
3. Hintz, A.: Fingerprinting Websites using Traffic Analysis. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 229–233. Springer, Heidelberg (2003)
4. Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, V.N., Qiu, L.: Statistical Identification of Encrypted Web Browsing Traffic. In: IEEE S&P 2002, pp. 19–30 (2002)
5. Song, D.X., Wagner, D., Tian, X.: Timing Analysis of Keystrokes and Timing Attacks on SSH. In: 10th USENIX Security Symposium (2001)
6. Saponas, T.S., Lester, J., Hartung, C., Agarwal, S.: Devices that Tell on You: Privacy Trends in Consumer Ubiquitous Computing. In: 16th USENIX Security Symposium, pp. 55–70 (2007)
7. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P.: Revealing Skype Traffic: When Randomness Plays with You. ACM SIGCOMM Computer Communication Review 37(4), 37–48 (2007)
8. Wright, C.V., Monrose, F., Masson, G.M.: On Inferring Application Protocol Behaviors in Encrypted Network Traffic. Journal of Machine Learning Research 7, 2745–2769 (2006)
9. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy Vulnerabilities in Encrypted HTTP Streams. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 1–11. Springer, Heidelberg (2006)
10. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Journal of Soviet Physics Doklady 10(8), 707–710 (1966)
11. Norvig, P.: How to Write a Spelling Corrector, `http://www.norvig.com`
12. Wilson, C.: Who checks the spell-checkers, `http://www.slate.com/id/2206973/pagenum/all/`
13. Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge (1997)
14. Needleman, S.B., Wunsch, C.D.: A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. Journal of Molecular Biology 48, 443–453 (1970)
15. Navarro, G.: A Guided Tour to Approximate String Matching. Journal of ACM Computing Surveys 33(1), 31–88 (2001)
16. Gooskens, C., Heeringa, W.: Perceptive Evaluation of Levenshtein Dialect Distance Measurements using Norwegian Dialect Data. Journal of Language Variation and Change 16(3), 189–207 (2004)
17. Felten, E.W., Schneider, M.A.: Timing Attacks on Web Privacy. In: 7th ACM CCS (2000)
18. Wagner, R.A., Fischer, M.J.: The String-to-String Correction Problem. J. ACM 21(1), 168–173 (1974)
19. TCPDump, `http://www.tcpdump.org`