Website Fingerprinting using Traffic Analysis of Dynamic Webpages

Yan Shi and Subir Biswas
Electrical and Computer Engineering, Michigan State University, East Lansing, MI

Abstract – This paper presents mechanisms for identification of web traffic masqueraded behind encrypted Virtual Private Network (VPN) tunnels. Website identification using Traffic Analysis (TA) has many administrative applications including preventing access to forbidden websites and site-specific Quality of Service (QoS) provisioning. Previous works in this area mainly looked at the problem of identifying traffic from relatively static websites, thus limiting the applicability of the technique for websites with dynamically changing contents. In this work, we attempt to generalize the mechanism for dynamic sites by the way of introducing a new classification feature *traffic surge period*, and adapting the *first n Components of Haar Wavelet Transformation*, which is commonly used in traditional signal processing applications. Our results from fingerprinting experiments carried out over an SSL VPN shows that the addition of these new features can indeed bridge the fingerprinting performance gap between static and dynamic websites.

Index Terms: **traffic analysis, website fingerprinting, virtual private networks, machine learning, classifiers.**

I. Introduction

Website Finger-Printing (WFP) refers to mechanisms using which an observer can identify the destination of a web transaction via statistical traffic analysis [1]. There are many applications of WFP ranging from enterprise system administrators attempting to prevent access to forbidden websites, to site-specific Quality of Service (QoS) provisioning. WFP becomes particularly challenging when traffic is encrypted, and a client uses some form of tunneling and/or proxy services in order to hide its destination website addresses. The problem of WFP is described using an example shown in Fig. 1.

Consider the arrangement in Fig. 1 in which a user is in the process of browsing a webpage from her home (bottom left) on a destination webserver (middle right). The user in this arrangement browses through an enterprise (e.g., the user's work place) so that the end-to-end connectivity spans through the user's home network, her ISP (Internet Service Provider), the public internet, the enterprise's ISP, the enterprise network, and finally the destination web server through the public internet. As shown in the figure, a tunneled configuration is used under which a Virtual Private Network (VPN) tunnel is set up between the user's machine and an enterprise VPN server. Such a tunneled access is quite common for secure communication in enterprise and electronic commerce applications over the public internet. Various types of VPNs including Secured Socket Layer (SSL) [2] and IPsec [3] tunnels are applicable.

In this arrangement, all packets from the web client up to the enterprise VPN server is end-to-end encrypted, and the IP address of the destination webserver is kept hidden to all the routers on the VPN tunnel. Now consider an application in which the home router wants to detect the webpages visited by the user. With hidden IP layer addresses, this is not feasible. However, if the home router has access to an a priori statistical model of the web traffic to specific webservers, it can probe and analyze the traffic meta-data (i.e., packet size, timing, rate, and their distributions, etc.) in run-time in order to detect the destination web sites.

The goal of such traffic analysis is to correlate a certain hidden (due to encryption) property of the probed stream to a signature in the metadata of that stream. Traffic analysis does not need to analyze the content, and therefore can be an effective tool for fingerprinting webpages based on encrypted traffic [4]. This can be done even when the IP layer addressing and packet contents are hidden. This mechanism is termed as Website Finger-Printing or WFP. Possible motivations for WFP by the home router would to be to either block specific sites or to provide site-specific QoS, when enabled. Similar WFP can be performed by the gateway router of an enterprise for site blocking or site-specific QoS provisioning when its employees may use external proxy services for web access.

As shown in the figure, traffic probing can be done at different points on the tunnel. For applications in which an ISP intends to perform WFP, it can probe tunneled traffic at the ISP's router on the tunnel. Generally speaking, deeper in the network probing is done, more difficult it will be to perform traffic analysis due to more traffic aggregation.

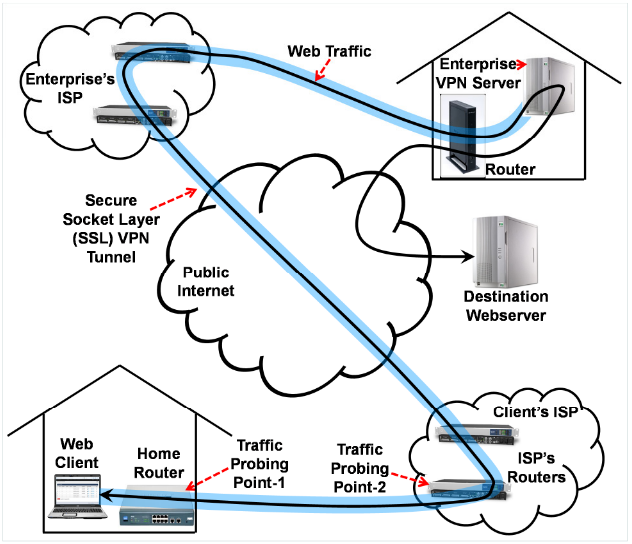


Fig. 1: Traffic and probe configuration for WFP

WFP can be performed on relatively static webpages for sites such as Google.com, Wikipedia and RationalWiki for which the traffic signature remains relatively static for long period of time. Prior work [1] [5] [6]in the literature attempts WFP on such static webpages so that the site classification is done based on very similar meta-data collected a priori. In this paper we set out to develop WFP techniques applicable to more dynamic webpages such as Facebook, LinkedIn, and Amazon for which the statistical differences between collected meta-data and analyzed traffic are more pronounced across different web-access sessions. The focus is in evaluating fingerprinting methods in a realistic context, with variability of web page content and complexities in traffic pattern due to web clients in consideration.

Specific contribution of the paper includes: 1) design and implementation of a realistic traffic probing arrangement similar to Fig. 1 and Fig. 2) novel feature definitions using probed traffic meta-data, 3) website classification using such features and performance evaluation, and 4) study the influence of dynamic web traffic on stability of the designed features.

II. Related Work

A. Virtual Private Network (VPN)

Virtual Private Networks (VPNs) [7] [8]transfer sensitive information through public network while minimizing the risk of information exposure. The goal is achieved by encrypting traffic payload at the source before sending it through public network while decrypting at the destination. A typical VPN tunnel and its encryption configuration are shown in **Error! Reference source not found.**.

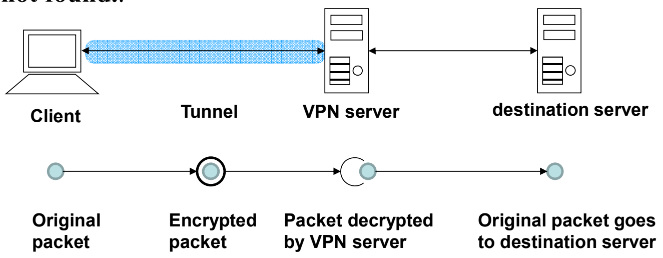
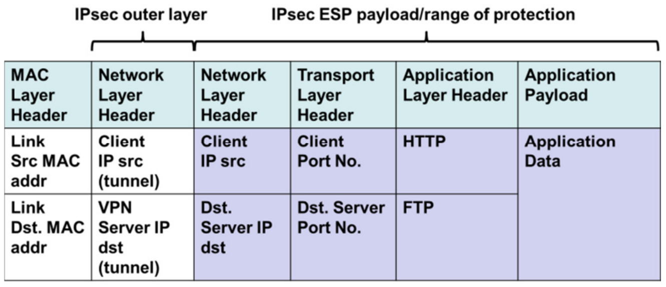

Client
Tunnel
VPN server
destination server
Original packet
Encrypted packet
Packet decrypted by VPN server
Original packet goes to destination server

Fig. 2: Encryption configuration through a VPN tunnel

IPsec is one of the two prevalent VPN tunneling stacks, which is built on Internet Protocol (IP) to provide secured transmission at network layer by authenticating and/or encrypting all IP packets in one IP communication session. IPsec stack contains the following members: Internet Key Exchange (IKE) protocol; two working modes (transport and tunnel mode); two protocols: authentication header (AH) for payload protection and Encapsulated Secured Payload (ESP) for routing protection.

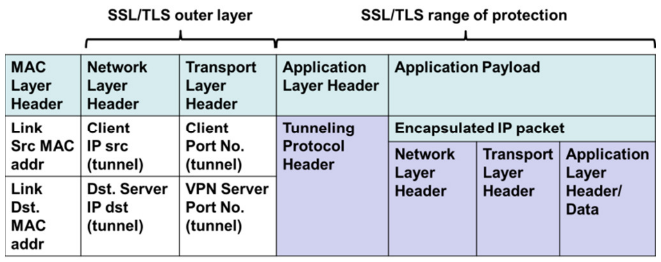

IPsec outer layer
IPsec ESP payload/range of protection

MAC Layer Header	Network Layer Header	Network Layer Header	Transport Layer Header	Application Layer Header	Application Payload
Link Src MAC addr	Client IP src (tunnel)	Client IP src	Client Port No.	HTTP	Application Data
Link Dst. MAC addr	VPN Server IP dst (tunnel)	Dst. Server IP dst	Dst. Server Port No.	FTP	

Fig. 3: Protection details within an IPsec VPN tunnel

When used as a base for VPN networks, tunnel mode with ESP is often used. It allows connection from either a single machine or a network to connect to a private network through an encrypted tunnel. In tunnel mode with ESP, an IP packet is encapsulated as payload into another IP packet. As shown in Fig. 3, the visible routing information, or the outer layer for the IPsec encapsulation, is the IP header that marks the beginning and the end of the tunnel, while the actual source and destination of the packet are encrypted within an inner header. As a result, at any probing point on an IPsec tunnel, no information about the destination web server is visible, which makes traffic analysis essential for website fingerprinting.

SSL/TLS is the other popular VPN tunneling method which is a transport layer cryptographic protocol. It protects application payloads by encrypting with public key encryption methods. It is built on top of TCP, and is used to tunnel other application layer protocols like http or ftp. In SSL/TLS, the client and server establish a secured connection by first negotiating a cipher suite and exchanging their public keys, then exchanging a pre-master secret key using public key encryption. Client and server calculate the master secret and session keys (encryption key and hashing key) from the pre-master secret. The rest of the connection is protected by a symmetrical cipher for encryption, and a hashing algorithm for authentication.


SSL/TLS outer layer
SSL/TLS range of protection

MAC Layer Header	Network Layer Header	Transport Layer Header	Application Layer Header	Application Payload		
Link Src MAC addr	Client IP src (tunnel)	Client Port No. (tunnel)	Tunneling Protocol Header	Encapsulated IP packet		
				Network Layer Header	Transport Layer Header	Application Layer Header/ Data
Link Dst. MAC addr	Dst. Server IP dst (tunnel)	VPN Server Port No. (tunnel)				

Fig. 4: Protection details for an SSL/TLS VPN tunnel

As shown in Fig. 4, unlike IPsec, SSL/TLS by itself does not provide encapsulation of IP layer routing information, but the IP layer contains information only about the tunnel end-points (i.e., the VPN server in this case). A specially engineered application layer protocol is used in combination with SSL/TLS protocol to encapsulate a packet starting from Network Layer header. In that case SSL/TLS based encapsulation can also be said to have an outer layer that is comprised of the unencrypted network layer which marks the beginning and end of the tunnel and the unencrypted transport layer header that marks the port used on both ends of the tunnel. The main difference from IPsec is that the outer layer port number, which is useful in identifying the protocol, is not protected by SSL/TLS. However, just like the IPsec tunnel, at a probing point on an SSL/TLS tunnel cannot determine the destination website address from the unencrypted information on the IP layer header. Thus, traffic analysis is still needed for website fingerprinting over and SSL/TLS tunnel.

In this work, the SSL based tunnel mode of a Juniper VPN server is used for the presented experiments.

B. Traffic Analysis

Under the protection of VPN, neither the content nor the actual identification of the destination web server is visible to routers on the tunnel. A node interested in WFP must turn to methods that do not have dependence on routing information or message body. Traffic Analysis (TA) is the primary method that can be used to fingerprint encrypted web traffic. Successful example of TA can be found in [9]. Those studies target a widely used encrypted Voice-over-IP (VoIP) protocol Speex. The classifier used consecutive packet size pairs as the features. Histograms of inter-packet size pairs are drawn from traffic streams, and then a nearest neighbor classifier is built based on the histogram feature. This scheme can achieve over 90% accuracy at recognizing the language a speaker is speaking out of 2 languages [9]. Similar results are also published by the same group for recognition of key phrases [10].

C. Website Fingerprinting

In [4] the authors lay a theoretical groundwork for analyzing network traffic using TA. Later Hermann et. al applies TA in [1] to tackle a practical WFP problem. The feature used in [1] is the frequency distribution of packet sizes. Traffic traces belonging to 775 popular domains are collected over a period of 2-17 days for several Privacy Enhancing Technologies (PETs), including

Open VPN [7], CiscoVPN [8], and The Onion Router (Tor) [11]. The study yielded good accuracy on both CiscoVPN (96.17%) and OpenVPN (94.94%), but poor accuracy on Tor (2.96%).

Tor clients communicate with web servers using randomly chosen multi-hop circuits through a network of Tor routers, making end-to-end correlation of Tor traffic difficult [11]. First major breakthrough of WFP attack on Tor is reported in [5], in which Tor traffic is intercepted on the route between client and Tor entry node. Features are extracted from traffic stream and then classified using a Support Vector Machine (SVM) classifier. They reported 55% accuracy on a closed-world scenario with a world size of 775 classes. The success of [5] resulted in several follow-up publications, including [6], which provides an evaluation of website fingerprinting techniques versus different feature sets and 9 countermeasures. Evaluated features include the feature set used by [5] , response time, packet lengths, total up/downstream byte counts and Variable n-gram (VNG), which is byte count of each burst. Finally a hybrid feature set called VNG++, constructed of VNG, trace time and total up/downstream byte counts, is evaluated and compared to other feature sets. All classifiers from previous works are evaluated with their respective feature sets accompanying them in the original work, while a naïve Bayes classifier is used to evaluate new feature sets like VNG++. The results showed that VNG++ with a naïve Bayes classifier is able to achieve similar performance to the classifier used in [5].

Website fingerprinting attacks on Tor as a real threat to Tor's anonymity are dismissed [12] by the Tor project. It is argued in [12] that all the works except [5] have insufficient world size, and that the works fail to address the problem of traffic features with dynamically generated web pages. They in turn suggested that future studies of traffic analysis attacks should focus on recognizing dynamically generated web pages. This claim is grounded on the fact that previous works fail to highlight the influence of dynamic contents on their features. A detailed study is therefore necessary to understand the extent of Tor's concern.

This paper attempts to address this by demonstrating how the previously used [1] [5] [6]and new WFP features perform in the presence of dynamic web traffic in which the traffic signature to the same site varies significantly across multiple access sessions.

III. Experimental Setup

In the experimental set up in Fig. 5, a web client is used from home to browse different servers through an SSL/TLS VPN service provided by one of Michigan State University's Juniper VPN servers. All packets are application layer encrypted (see Fig. 4) through the VPN tunnel which terminates at MSU's VPN server. The server interprets the packets sent by client through an SSL session within the tunnel, and forwards them to the intended target web server. When the target server responds, the VPN server uses the same SSL session for sending traffic back to the client. From the client application's perspective, the content of the web page comes from the target server.

The SSL/TLS tunneled traffic is probed at an egress router by the traffic capture software WireShark. At a probing point, packets do not have any destination specific identifiers/addresses since they are embedded into the application layer header as shown in Fig. 4. To separate the traffic stream it is assumed that the client IP address as well as the VPN server IP address is known to the probing entity, which is consistent with the QoS provisioning or site blocking applications as mentioned before. The clients run Internet Explorer *V.11* in a windows 7 machine to access the target webpages. Note that this arrangement also emulates scenarios in which probing is done by an ISP when it intends to perform website blocking or site-specific QoS allocation based on traffic analysis based website fingerprinting.

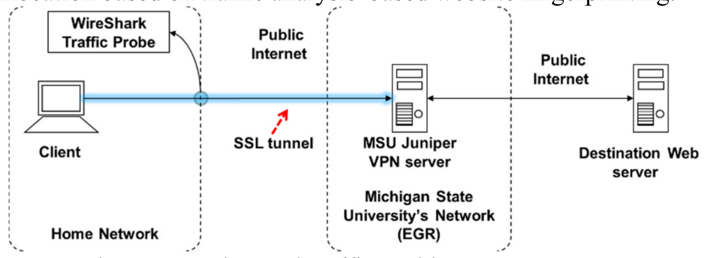


Fig. 5: Experimental traffic probing arrangement

Traffic is collected for 11 popular website front pages that represent dynamic content. Those are listed in Table 1. This group of websites is referred to as the experimental group. Content dynamism is ensured by accessing personal home pages with sufficient interval so that the home page changes significantly in terms of text and image as parts of newsfeeds, status updates etc. Quantification of such intra-site dynamism is presented in Section *VI*. Data collection is done through a time span of 10 days, with an average of 30 samples of each page being taken a day. Consecutive data collections are separated by at least 20 minutes to ensure that the web pages are sufficiently different. During the data acquisition phase, we collected 3250 valid samples. The 11 sites chosen comprises of major classes of web sites with dynamic contents.

No.	**Web page**	**Class**	**No. samples**
1	www.amazon.com	E-commerce	315
2	www.ebay.com	E-commerce	320
3	www.facebook.com	Social news	311
4	news.google.com	News	312
5	www.linkedin.com	Social news	147
6	www.reddit.com	Social news	322
7	www.slashdot.org	Social news	287
8	www.stackoverflow.com	Social Q&A	308
9	en.wikipedia.org/wiki/Main_Page	News	310
10	www.washingtonpost.com	News	305
11	www.yahoo.com	News	313

Table 1: Target dynamic websites as the experimental group

As a control group, we also collect data for several static web pages as shown in Table 2. The web pages in the control group don't change much over time. Traces from the control and the experimental groups are collected under identical conditions. In Section *VI*, we will assess the impacts of dynamic pages on WFP performance with respect to the static control group.

No	**Web page**	**No. samples**
1	http://www.audioheritage.org/html/history/altec/altec-1963.htm	186
2	http://rationalwiki.org/wiki/Global_warming_denialis	193

	m	
3	http://grandcanyonhistory.clas.asu.edu/sites_adjacentlands.html	188
4	http://docs.oracle.com/javase/7/docs/api/	190
5	http://rationalwiki.org/wiki/Moon_landing_hoax	188
6	http://en.wikipedia.org/wiki/Panellus_stipticus	187
7	http://en.wikipedia.org/wiki/Plato	180
8	http://tools.ietf.org/html/rfc5246	195
9	http://www.speex.org	194
10	http://www.voynich.nu/descr.html	182
11	http://www.fcps.edu/islandcreekes/ecology/water_bear.htm	190

Table 2: Target static websites as the control group

IV. Traffic Pattern for Web Transactions

A typical web transaction is initiated by a request from a web browser to a web server, which replies to the request with objects that are parts of a web page via HTTP protocol. In the beginning, the client sends a request including the URL of web page and any parameter needed to server. The server responds with the main textual content of the web page in Hypertext Markup Language (HTML) form to the client. The client/browser then parses the main content, discovers reference to any additional resource needed to show the page, and sends one request for each resource. The server responds in the same order as it receives the requests. This process continues recursively until all references are responded to, and the transaction concludes.

There are several additional complexities that are added due to the following optimizations in the transaction process. First, at the client side the download and parsing often go in parallel for improved viewing responsiveness. This often means that requests are usually sent out as soon as possible when a reference is revealed by the parsing process. This behavior also has the implication that the timing of request should indicate its position in the HTML text stream.

Second, for higher performance, a client can choose to establish several HTTP connections to download resources in parallel. Also, if the server supports HTTP pipelining, then the client can send several requests in one TCP datagram. The server can also respond to such requests in one batch. These optimizations can interfere with the order of request/response pairs. The timing of requests can be mingled with response data from other connections. Third, the underlying network conditions can change the shape of traffic, especially in terms of the inter-packet delay jitter. This means that the timing of each burst can change because of different download times.

Fourth, a TCP connection can be reused for multiple requests in order to avoid the delay involved in successive connection setup and termination. This means that it is not safe to assume that one connection corresponds to one resource. If the protocol is plain HTTPS, then a connection can still tell actual object sizes transferred, while if the protocol is modified by the VPN client, for example distributing one resource among multiple connections, even connection-wise burst sizes can be unreliable. Assuming that encryption does not change traffic signature drastically, all the above optimization related traffic pattern changes apply to encrypted web traffic within a VPN tunnel.

Fig. 6 shows an example 4 sec. long traffic trace for a transaction from LinkedIn web server, collected using WireShark as shown in Fig. 5. Decomposition of the trace was performed as follows. In the figure, the stripe to the left marks timing of different HTTP requests sent by the client to the server. The middle plot area is divided into a number of lanes, and each TCP connection used in this transaction is plotted in one lane. One lane is further divided into 2 halves vertically, with the left half used for upstream traffic and the right half for downstream traffic. Dark stripes mark the point in time where the connection is downloading. It can be seen as to how requests are processed by different TCP connections (numbered #1 to #8) and how the traffic of different connections interleaves. To the right, the traffic of all TCP connections combined together is shown.

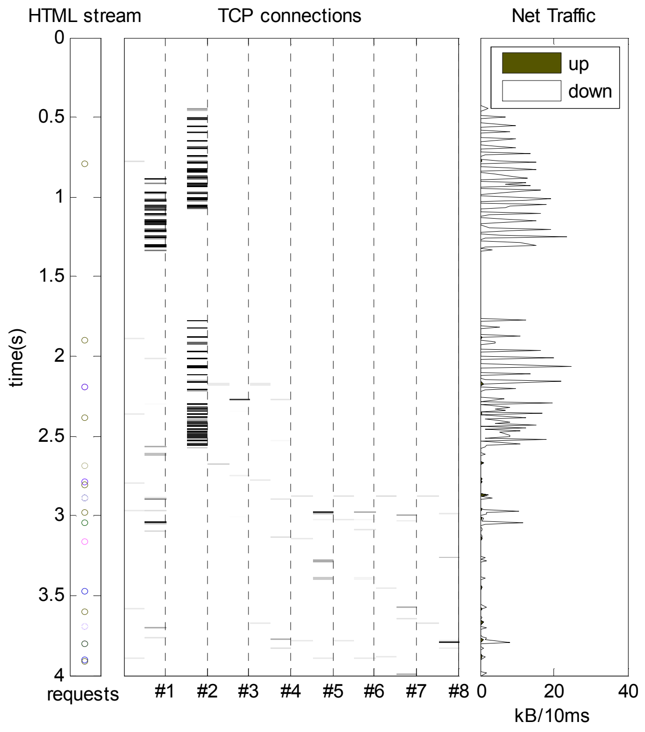


Fig. 6: Decomposition of trace from an example transaction

This example trace shows the complexity of underlying process and the challenges of finding traffic signature patterns in the presence of deviations from the baseline transaction process.

V. Feature Definition And Extraction

Pre-processing of the probed traffic is performed following the approach in [5]. Pre-processing removes the non-TCP and pure ACK packets that might affect the results.

A. Existing Features

The term *trace* is used to describe a time-stamped sequence in which every packet is associated with the time it appears on a traffic probing point. It can be represented as a series of timestamps as $\{t_n\}$. We assume that the content of traffic is generally encrypted and that only the direction and packet size are of importance. Following the convention in [1], we combine the packet size and direction into one scalar. The absolute value of the scalar equals the size of the packet. All upstream packets have negative signs and downstream packets have positive signs. This series of packet sizes is defined as $\{b_n\}$. The following features have been used in the literature.

Burst Size: In [6], a burst is defined as the interval during which there are only packets in one direction, but preceded and followed immediately by packets in the opposite direction. There re upstream and downstream bursts. In Fig. 6 upstream bursts are shown on the negative half of the graph while downstream packets on the positive half. Each time the stream switches direction from the negative half to positive half or vise-versa, a burst boundary is recorded. Each burst is marked by 2 boundaries and the direction of packets inside that burst becomes the direction of the burst itself. Burst is also used in [5], albeit named differently. It is used as part of the feature set in two different ways. 1) Number of bytes transmitted in each burst is summed up into a histogram and contained in the feature. 2) Number of packets transmitted in each burst is summed up into a histogram and contained in the feature.

First *n* Components of Haar Wavelet Transformation: Haar wavelet transform is often used in signal analysis to analyze local information of a signal. We use this as a feature to represent coarse-scale variation information of a trace. Each trace is first converted to a bit-rate series using a 100ms time window, then interpolated to 4096 data points before transforming. We take first 15 components of the transformed vector as the feature.

B. Surge Period

We propose a new feature, termed as *Surge Period*, which unlike the packet direction based bursts, is defined based on the timing of surges in traffic density, or high bandwidth utilization. A *Surge Period* marks the parts of traffic trace where the channel is busy transmitting packets upstream or downstream with back to back packets. Any packet except for the first one and the last one within the surge period should be separated from its predecessor and subsequent packets by a time period no larger than a pre-defined time window size. The window size depends on the network condition and typically ranges from 10 milliseconds to several hundred milliseconds.

During web transaction, the timing of a *Surge Period* corresponds to one object, or a group of objects being downloaded together. We have speculated that, on a coarse scale, the timing of a *Surge Period* corresponds to the location of those references that resulted in it. This is based on the fact that modern browsers want to render the contents with the highest speed possible, and that they should send requests for resources quickly after the references are discovered by the parser.

As demonstrated in Fig. 4, while the probed data can be decomposed into individual TCP connections, the *Surge Period* feature is extracted from the combined traffic pattern (the far right column in Fig. 4). This is done so that the feature should be prepared for situations, such as Tor [11], in which the traffic cannot always be separated into individual connections.

As done in [1] [5], we assume that all traffic from or to a browser client are from the same transaction. Meaning, we ignore interleaving of multiple web transactions.

The *Surge Period* feature is extracted as follows. First, a probed trace is converted into a time-stamped series of (time, packet-size) pairs. Second, the time-stamped series is converted into a bit rate time series computed over 100ms non-overlapping windows. Third, the most significant *Surge Periods* are then extracted from the bitrate time series. To do this, we apply the following adaptive method. A continuous period of bitrate higher than a certain threshold I_{th} is counted as a *Surge Period*. Multiple thresholds are experiment with, starting from the highest value possible, and gradually descending towards until a threshold where more than 80% of all the bytes transferred are covered by in burst periods. Finally, the number of bytes transmitted in a *Surge Period* is summed up as the 'size' of that period, and all *Surge Periods* are lined up according to their timing order. Top *N* periods in size are then chosen, where *N* is an arbitrary number. The resulting vector of period sizes are used as the feature that represents the sample. If there is not enough number of *Surge Periods* even after $P_{th} = 0$ is tried, the vector is padded with zeroes at the end.

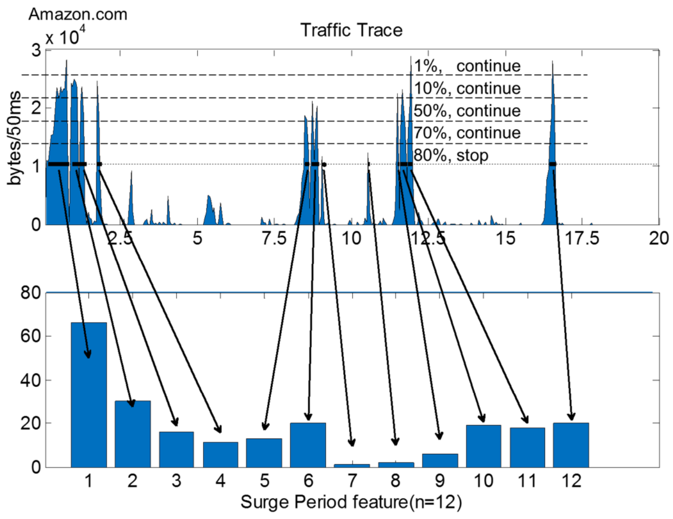


Fig. 7: Example of *Surge Period* for amazon.com traffic

Fig. 7 shows an example computation of the *Surge Period* feature on an actual trace captured from amazon.com traffic. The bitrate vs. time graph is shown in the upper half of Fig. 7 with a descending threshold. When the threshold is lowered to a level where 80% of the traffic is included in various surges, the threshold is made fixed. With this threshold level, the surges are extracted and then rounded up to form the vector shown in the lower part of Fig. 7. The arrows indicate the correspondence between surge periods and feature.

VI. Performance

A. Feature Analysis

In this section, we present performance comparisons of classifying websites using different features. First, we demonstrate the dynamism of the dataset we collected by presenting sample probed traffic trace for web transactions corresponding to different destination sites. Fig. 8 shows data rate over time during the progression of a transaction. Traffic in both directions is considered in the graphs.

It can be observed that the data rate signatures for different destination sites vary significantly in terms their burstiness, number of bursts, and the aggregated average rate. These variations in the traffic patterns translate into distinguishable features across various target websites, thus yielding good classification performance as reported later in this section. To depict dynamism of the pages to the same website, in what follows, we present intra-site feature variation by showing samples of burst sizes, Haar wavelet transform components, and surge period features from various target sites.

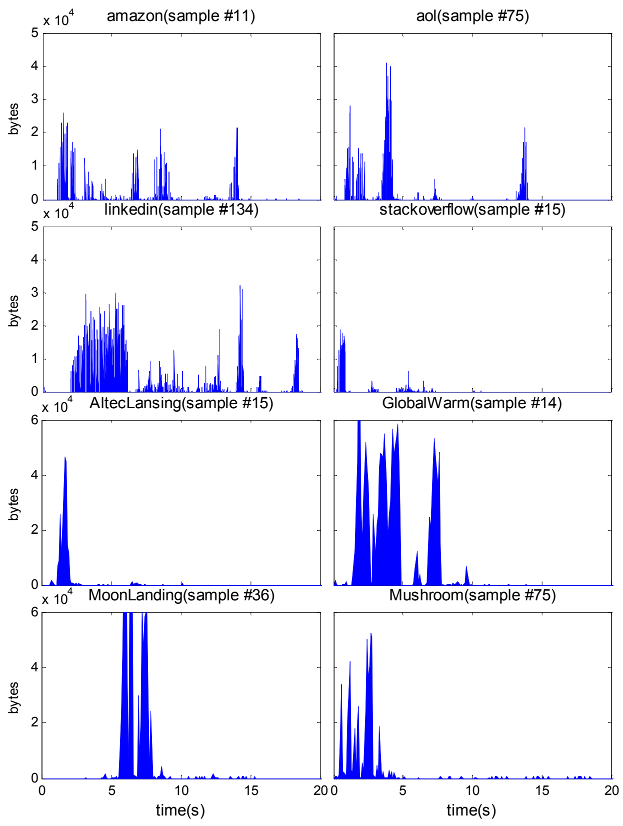


Fig. 8: Traffic rate traces for web transaction to different sites

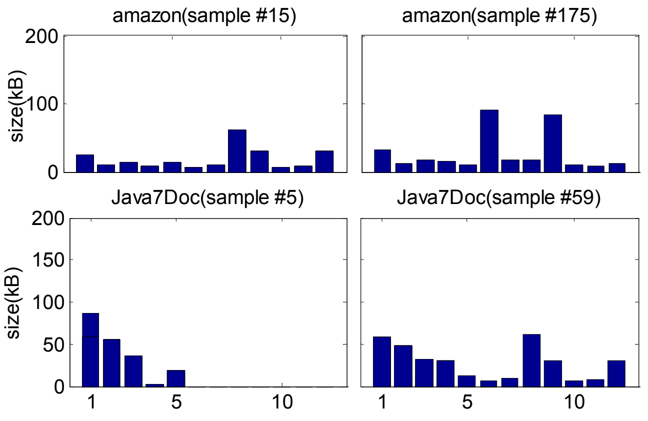


Fig. 9: Surge Period for dynamic and static sites

Surge Period of 2 sites are presented in Fig. 9. One row in the graph corresponds to one site. The site above comes from the experimental dynamic group while the bottom one comes from the static control group. It can be seen that the feature is able to capture the intra-site variation, as well as the obvious inter-site variation between the 2 sites. Samples of Haar wavelet transformation components are presented in Fig. 10. The distinction between 2 sites and their two access sessions are obvious for this feature.

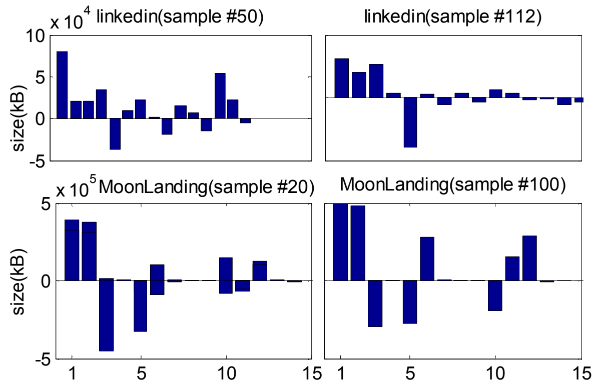


Fig. 10: First 15 Haar components for dynamic and static site

Due to space constraint, all intra- and inter-site variations in Figs. 9 to 10 are shown only for a limited number of sites and samples. Similar variations were observed across all experimented sites and across all sessions for the same sites.

A. Website Fingerprinting

We use the Bayesian network classifier (BayesNet) from the Weka [13] machine learning library for executing WFP. As shown in Fig. 11, the classifier is trained offline with individual features from Table 3 or combinations of them.

	Feature Set
1	Histogram of burst sizes
2	First *n* components of Haar wavelet transform (n=15)
3	*Surge Period (N = 10)*

Table 2: Feature sets used in experimentation

The training is supervised and features extracted from each trace are marked with a class label that represents the URL where the trace is from. There are 11 classes (i.e., URLs) for the experimental set and 11 classes in the control set. A full list of feature sets experimented with is provided in Table 3.

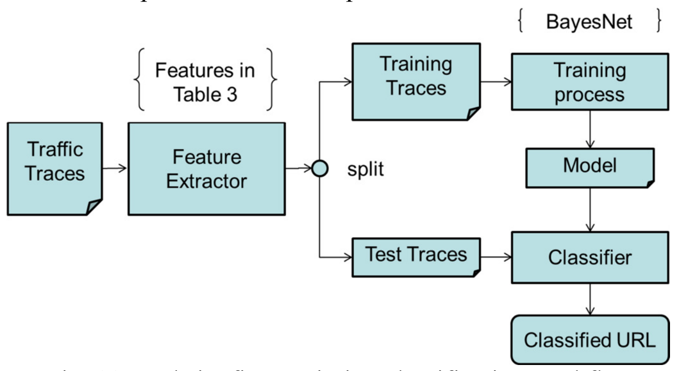


Fig. 11: Website fingerprinting classification workflow

To test the performance after training phase, we do a pseudo-random 2:1 training/set partitioning before the classifier is trained, then train the classifier with the training set and test it with the test set. The whole process is shown in Fig. 11.

Given their lower intra-site variations in the traffic signatures, it is expected that the classification results for the static sites would be better than the dynamic sites. We first experiment with the static sites with three feature scenarios, namely, *burst size,* *burst size plus surge period, and burst size plus Haar*. Class-wise results for the control group are shown in Table 6.

Site No.	Burst size	Burst size + surge period	Burst size + Haar
1	0.938	0.969	0.984
2	0.982	0.964	0.982
3	0.849	0.904	0.959
4	0.896	0.955	1
5	0.953	0.953	0.953
6	0.878	0.959	0.959
7	0.877	0.985	0.892
8	0.958	0.958	0.958
9	1	1	1
10	1	0.981	0.963
11	0.984	1	0.984

Table 4: Class-wise true positive rate for the control group

Observe that all three features scenarios generally work very well for the static website fingerprinting. This is consistent with previous results published in [1] [5] [6] . The next step is to apply the same feature scenarios for the experimental dynamic sites. Class-wise results for dynamic sites are shown in Table 5.

Site No.	Burst size	Burst size + surge period	Burst size + Haar
1	0.963	0.984	0.982
2	0.771	0.982	0.925
3	0.981	0.959	0.854
4	0.976	1	0.973
5	0.979	0.953	0.992
6	0.935	0.959	1
7	0.726	0.892	0.925
8	0.818	0.958	0.878
9	0.964	1	0.972
10	1	0.982	0.99
11	0.972	0.963	0.981

Table 5: Class-wise true positive rate for the experimental group

It should be observed that when the traditional feature *burst size* is used alone, the fingerprinting performance degrades compared to the static sites as shown in Table 4. For certain sites (e.g., sites 2 and 7), the loss of performance was substantial. The reasons for such loss are due to high intra-session diversity for the same target site as depicted in Fig. 8 through 9.

Table 6 also shows results when the feature *burst size* is aided with our proposed feature surge size, and the components of Haar transform. It is evident that these new features improve the overall fingerprinting performance by being able to characterize and identify the inter-site variation in probed traffic data.

Site Group	Burst size	Burst size + surge period	Burst size + Haar
Experimental	0.873	0.939	0.967
Control	0.938	0.970	0.978

Table 6: Average true positive rates for both the site groups

Table 4 reports the overall true positive rates averaged over all the sites within both the groups. These figures confirm the general observations made for individual site-specific performance as reported in Tables 4 and 5. Following two summary observations can be made. First, fingerprinting performance on static (i.e., control) sites are generally better than those for the dynamic experimental sites. Second, adding the new features, namely, surge size and the components of Haar transform, improves results for both static and dynamic sites, although more notably for the dynamic group.

VII. Summary and Conclusions

We have presented characterization and performance enhancement of a website fingerprinting mechanism. Using experiments carried out over an SSL VPN, it was first shown that a widely used classification feature, namely, traffic burst size, is not able to perform well when the target website contents are sufficiently dynamic. This effect was mitigated by using a new classification feature *traffic surge period*, and adapting the *first n Components of Haar Wavelet Transformation*, which is commonly used in traditional signal processing applications. Presented results show that the addition of these features can bridge fingerprinting performance gap between static and dynamic websites. Overall, the true positive rates for dynamic website fingerprinting were enhanced from about 87% with traditional features, to up to 97% by leveraging those new features. Future work includes generalizing the mechanism for multi-site probing and over heterogeneous VPN tunnels.

references

[1] D. Herrmann, R. Wendolsky and H. Federrath, "Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naive-bayes Classifier," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, New York, NY, USA, 2009.

[2] T. Dierks and E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2,* IETF, 2008.

[3] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," 2005.

[4] J.-F. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems," in *PROCEEDINGS OF INTERNATIONAL WORKSHOP ON DESIGN ISSUES IN ANONYMITY AND UNOBSERVABILITY*, 2001.

[5] A. Panchenko, L. Niessen, A. Zinnen and T. Engel, "Website Fingerprinting in Onion Routing Based Anonymization Networks," in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, New York, NY, USA, 2011.

[6] K. P. Dyer, S. E. 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*, Washington, DC, USA, 2012.

[7] M. Feilner, OpenVPN: Building and Integrating Virtual Private Networks, Packt Publishing, 2006.

[8] A. G. Mason, Ed., Cisco Secure Virtual Private Networks, Cisco Press, 2001.

[9] C. V. Wright, L. Ballard, F. Monrose and G. M. Masson, "Language Identification of Encrypted VoIP Traffic: Alejandra Y Roberto or Alice and Bob?," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, Berkeley, CA, USA, 2007.

[10] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose and G. M. Masson, "Spot Me if You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2008.

[11] R. Dingledine, N. Mathewson and P. Syverson, "Tor: The Second-generation Onion Router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, Berkeley, CA, USA, 2004.

[12] M. Perry, *A Critique of Website Traffic Fingerprinting Attacks,* 2013.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explor. Newsl.,* vol. 11, no. 1, pp. 10-18, #nov# 2009.