Junhua Yan* and Jasleen Kaur

Feature Selection for Website Fingerprinting

Abstract: Website fingerprinting based on TCP/IP headers is of significant relevance to several Internet entities. Prior work has focused only on a limited set of features, and does not help understand the extents of fingerprint-ability. We address this by conducting an exhaustive feature analysis within eight different communication scenarios. Our analysis helps reveal several previously-unknown features in several scenarios, that can be used to fingerprint websites with much higher accuracy than previously demonstrated. This work helps the community better understand the extents of learnability (and vulnerability) from TCP/IP headers.

Keywords: website fingerprinting, feature selection

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

1 Introduction

Why Study Website Fingerprinting? Website fingerprinting refers to the task of identifying the website being visited, based on information available in the TCP/IP headers of the network traffic generated. This task is of significant relevance to at least two types of entities. The first are legitimate entities—including network managers, Internet Service Providers, regulators, and researchers—that are, respectively, interested in protecting enterprise networks, gauging user interests, studying unfair business practices, and studying the performance of Internet applications and services. The second are illegitimate entities with the malicious intent of exploiting vulnerable parts of private user data.[1] With increasing adoption of privacy-enhancing technologies [1–4], access to readable data beyond TCP/IP headers is fairly limited—a key question that needs to be answered by the above entities is: *to what extent can websites be fingerprinted (or not) from the TCP/IP headers of Internet traffic?*

***Corresponding Author: Junhua Yan:** UNC Chapel Hill, E-mail: junhuayan@cs.unc.edu
Jasleen Kaur: UNC Chapel Hill, E-mail: jasleen@cs.unc.edu

1 Closely related are security experts that want to prevent such malicious privacy attacks.

State-of-the-art: Limited Set of Features Several studies have demonstrated the ability to fingerprint websites in different scenarios, including when complete access to TCP/IP header fields is available [5–8], when the IP addresses are anonymized [9], when virtual tunnels are used to hide information about TCP sub-flows [6, 10–15], when anonymization overlays like Tor are used [16–26], and when packet sizes are padded [11, 14, 27–29]. Using machine learning techniques with informative features like packet sizes and sizes of consecutive burst of packets, for instance, these studies have achieved high accuracy in fingerprinting from a closed set of popular websites.

While quite illuminating, these studies answer our key question in only *anecdotal* ways. They focus heavily on only a handful of tried-and-tested informative TCP/IP features—when one study demonstrates that *a* feature like packet sizes is quite informative [11, 12], others come along with techniques to specifically camouflage packet sizes [14, 27–30]. By focusing on only a limited set of features, prior work does not help us understand the "extents" of learn-ability (and vulnerability) from TCP/IP headers—what is the list of *all* TCP/IP features that are informative for website fingerprinting? If some of these features are camouflaged, are there others that can still be informative enough? How effective are they?

Our Approach: Exhaustive Feature Analysis (in Diverse Scenarios) In this paper, we make and test two claims. The first is that there are numerous features that can be derived from the TCP/IP headers of web traffic, and it is important to analyze these exhaustively in order to understand the extents of website fingerprint-ability. The second is that the most informative features-of-interest are likely to change, when some parts of the TCP/IP structure are hidden (e.g., in VPN or Tor tunnels). We evaluate these claims by:

1. Extracting a comprehensive list of TCP/IP features;
2. Considering eight different communication scenarios (which differ in access to TCP/IP information);
3. In each scenario, identifying and analyzing informative features and evaluating their efficacy in website fingerprinting.

Our analysis validates both claims. We discover several previously-unknown features that can be used to fingerprint websites in different communication scenarios, of-

BY-NC-ND

ten with much higher accuracy than previously shown. Our findings emphasize the importance of exhaustive feature analysis in developing a true understanding of the limits of learn-ability from Internet traffic.

In the rest of this paper, Section 2 summarizes prior work on website fingerprinting. Section 3 presents our data collection and analysis methodology. Sections 4-9 analyze the most informative features and evaluate their performance in different communication scenarios. We summarize concluding remarks in Section 10.

2 Problem Formulation

Background: Website Fingerprinting Website fingerprinting refers to the task of *learning* which website is being visited, based on information available from the TCP/IP headers of network traffic. Note that the collective network traffic yields headers from each packet, transmitted within each TCP connection initiated for a webpage download, along with a time at which each packet was observed at the vantage point. The packet headers include the source and destination IP address and port numbers, seq number, ACK number, TCP flags, receiver window size, TCP segment length, IP-ID, Protocol, and fragmentation/reassembly fields.[2] Header-based features derived from traces of known web transfers, can be fed to a supervised machine learning framework to train a classifier to fingerprint websites.

State of the Art Prior work has demonstrated a strong ability to fingerprint websites in several different communication scenarios, including HTTPx transfers[3] [5, 7, 9], encrypted tunnels such as OpenVPN or OpenSSL [6, 8, 10–15], and advanced anonymous networks like Tor [16–26]. This body of work relies on informative TCP/IP features—such as packet sizes, sizes of burst of packets, packet and byte count, inter-arrival times, and count of unique servers. There is also a significant body of work on designing countermeasures to prevent fingerprinting—most of these focus on obfuscating informative features such as packet sizes, burst sizes, and packet timings [14, 27, 28, 30–32]. A brief summary of prior work is included in Table 1 and details can be found throughout the paper.

Need for an Exhaustive Feature Analysis A key component of machine learning is *feature engineering*, which involves using domain knowledge to manually craft features that are likely to make classifiers perform well—"Applied machine learning is basically feature engineering" [34]. Even with just 40 bytes of information available in TCP/IP headers, there are a surprisingly large number of structural traffic features that can be extracted from a webpage download. These would include the IP footprint (server counts, IP addresses, timing and order of servers contacted), the TCP connection footprint (connection count, sizes, ports used, timing and order of connections), the data transfer footprint within TCP connections (burst counts, sizes, timing), and packet-level footprint (count, sizes, timing)—our efforts to comprehensively list features in Section 3.2 yield more than 109 feature categories, contributing a total of nearly 35,683 features!

The state of the art, in contrast, has considered only a handful of features for website fingerprinting (Table 1). This may seem surprising, but it is not—features such as packet sizes and burst sizes have indeed turned out to be so informative that most studies have been able to achieve very high fingerprinting accuracy by just re-using a few features.

This approach, however, will not work for us—our objective in this paper is not to simply achieve high fingerprinting accuracy in a given scenario, but rather to understand the *extent* to which websites can be fingerprinted using TCP/IP headers (especially when well-known informative features are successfully masked). Such an objective necessitates that we comprehensively analyze the informativeness of *all* TCP/IP features.[4]

The work most closely related to ours is [33], which considers Tor and analyzes the importance of several features proposed in [10, 18, 22], along with some self-defined features—a novel classifier that combines random forest and distance metric is then proposed for website fingerprinting. However, the focus of that work is not an exhaustive feature analysis and a majority of features we define are not considered—besides, only the Tor-based transfers are considered.

2 When privacy-enhancing technologies are used, however, some header information may be unavailable. For example, server IP addresses and port numbers are unavailable in encrypted tunnel and packet sizes are obfuscated when packet-padding is used. We consider several such communication scenarios in this paper.

3 In this paper, we use HTTPx to represent all of HTTPS, HTTP/1.0, HTTP/1.1 and HTTP/2.

4 This need has been partly recognized in [35], which emphasizes the importance of shifting the focus of future website fingerprinting research to identifying optimal feature sets (by deriving security bounds for website fingerprinting defenses based on a given feature set).

Table 1. Summary of prior work evaluated in this paper (including communication scenario, feature set, and classifier). Note that "*" indicates the author did not specify the property.

Author	Scenario	Features	Classifier
Liberatore (*L*) [11]	SSH	packet size count	Naive Bayes
Herrmann (*H*) [12]	SSH, Tor	packet size frequency	Multinomial Bayes
Panchenko (*P*) [18]	SSH, Tor	size markers, HTML markers, # markers, percentage incoming packets, occurring packet sizes, transmitted bytes, # of packets	SVM
Dyer (*Vng++*) [14]	SSH	per-direction bandwidth, total time, burst markers	Naive Bayes
Wang (*FLSVM*) [21]	Tor	Tor cell instances	Distance-based SVM
Feghhi (*DTW*) [15]	SSH	uplink timing information	Dynamic Time Warping
Panchenko (*CUMUL*) [23]	Tor	# of incoming & outgoing packets, sum of incoming & outgoing packet sizes, interpolant of cumulative packet size	SVM
Hayes (*k-FP*) [33]	Tor	# of packets, percentage incoming & outgoing packets packet ordering, concentration of outgoing packets, # of packets per second, inter-arrival time, transmission time	Random Forests
Trevisan (*T*) [7]	HTTP	server IP address count, hostname count	*

Challenges and Goals An exhaustive analysis of all features is an unusual machine learning task. There are several challenges that must be addressed:

- The first step is to exhaustively list all features—we believe this is infeasible, given that the infinite possible feature combinations (and statistical derivatives) can not be manually listed. Instead, we pursue the more modest goal of comprehensively listing semantically-relevant groups of features (Section 3.2).
- The large list of features will necessarily contain several features that are correlated. The existence of interdependent features have three potential side-effects: possibility of over-fitting [36], curse of high-dimensionality [37], and (most relevantly) misleading interpretation of feature importance. Although dimensionality-reducing algorithms such as Principal Component Analysis may help get rid of correlated features, they do not let us understand feature importance, which is the main focus in our work. Our next goal is to derive a comprehensive list of *important* features that are not correlated (Section 3.4).
- There are several communication scenarios used widely in the Internet today that conceal some TCP/IP features (e.g., the use of encrypted tunnels hides all TCP sub-flow information). It follows that features that are informative in one scenario may not be as revealing in others—e.g., packet size distribution has been shown as as a powerful feature in encrypted tunnels; however, this is not the case in Tor since all packets are padded to a fixed length. In order to truly understand the *extents* of website fingerprint-ability, we repeat our comprehensive feature analysis under a diverse set of prominent communication scenarios (Section 3.3).

Innovations This paper presents two key innovations. First, it defines and considers several communication scenarios and performs a comprehensive feature importance analysis in each. Second, it evaluates the efficacy of the newly-discovered informative features in accurately fingerprinting websites (by comparing with prior work). We believe such a comprehensive analysis is an important first step in two seemingly-competing directions—aiding legitimate entities in robustly fingerprinting under different communication scenarios, and aiding security researchers in designing fool-proof countermeasures against website fingerprinting.

3 Methodology

3.1 Data Collection Methodology

Website Downloads The website dataset used in this paper is collected by visiting the landing page of the top 3,000 worldwide websites listed on Alexa [38]. Each website is visited 20 times using *Google Chrome Version 61.0.3163.100* (cache disabled) on a desktop machine—*Selenium webdriver* is used for web browser automation.[5] When visiting a website, we set a 20s *time-out* before closing the Chrome browser—pages that fail to load within this period will be marked as a failure.[6]

A successful visit is defined as an *instance*. In total, we successfully visited 2,712 websites at least once within *20s*. The total number of instances was 44,944.

Website Labels Since the focus of this work is website fingerprinting, we use the domain name that appears in

5 https://github.com/SeleniumHQ/selenium/

6 We use *set_page_load_timeout(time_to_wait)* provided by Selenium; pages that fail to be completely loaded within *time_to_wait* will throw an error. These are either invalid URLs or too slow to load.

Table 2. Examples of features in each level.

Level	*Examples*
Packet	packet size count, packet frequency, initial 30 packets, # of packets per TCP conn., # of incoming packets, ...
Burst	burst size count, # of incoming/outgoing bursts per TCP conn., bytes of incoming/outgoing bursts, burst duration, ...
TCP	# of TCP conn., incoming bytes per TCP conn., ...
Port	port count, transmitted bytes per TCP conn. w.r.t. port 443/80, ...
IP address	server IP address count, hostname count, transmitted bytes per TCP conn. w.r.t. server IP address & hostname, ...

its *URL*, as the ground-truth label for each website. We take *redirection* into account and consider only the final *URL* for a given website. Of the 2,712 websites, 52 are redirected to new *URLs* (e.g., *http://extraimago.com* is redirected to *http://extraimage.net/*), while 125 are redirected to other *URLs* within our list—one of the most common *URLs* for redirection is *http://google.com*. Among them, 1,032 websites have at least 20 instances while 2,032 websites have at least 16 instances. For feature selection and performance evaluation, we randomly picked up 2,000 websites out of 2,032 that have at least 16 instances for training and testing (Section 3.4).

Other Datasets For our evaluations, we also use the *SSH2000* dataset provided by Liberatore et al. [11] and the *Tor* dataset shared by Wang et al. [22]—these evaluations are included in Appendix 12. *SSH2000* [11] is collected by visiting 2,000 popular websites, 51 times each, through a SOCKS proxy. All TCP traffic is encapsulated under one tunnel—thus packet size, direction, and time are available in TCP/IP headers, but all information about individual TCP connections, server IP addresses and port numbers is hidden. The *Tor* dataset [22] consists of instances obtained by visiting 100 websites, 90 times each, with the *Tor* browser. Due to the use of onion routing and packet-padding mechanism in *Tor*, actual packet size, information about individual TCP connections, server IP addresses and port numbers are not available.

3.2 Feature Engineering

In machine learning, powerful features are defined in order to make explicit certain implicit characteristics of data. In this section, we summarize the different types of features we extract from TCP/IP headers. We group our features into five levels—examples of feature categories in each level are presented in Table 2:[7]

1. **Packet-level.** This level includes features directly derived from the count, length, timing, and direction of packets. While packet sizes have been used quite successfully in the state of the art, we include features that span the order, timing, and direction of packets as well. We define 43 semantically-different feature categories in this level.
2. **Burst-level.** A burst is defined as a sequence of packets sent in one direction between two packets from the opposite direction [14]. For example, a sequence of packet sizes (220, 1440, -300, -810, 530) can be described as (1660, -1110, 530) in burst level.[8] In burst-level, we focus on features derived from the count, duration, bytes, packets, as well as inter-arrival times. There are 25 semantically-different feature categories in this level.
3. **TCP-level.** For HTTPx transfers, we are able to infer which TCP connection a packet belongs to. We define features based on the count, duration, number of bytes, packets, and bursts, and inter-arrival times of the TCP sub-flows. There are 14 different semantic feature categories in this level.
4. **Port-level.** Port numbers have been previously used to distinguish different types of traffic [39–41]. In this work we consider features related to the usage of different port numbers as well as the amount of traffic to/from port 443 (HTTPS) and 80 (HTTP) in a traffic trace. We define 7 semantically-different feature categories in this level.
5. **IP address-level.** Server IP addresses and hostnames have been used previously for website fingerprinting [7]. We extract related features by calculating the frequency with which different server IP addresses and hostnames (derived from reverse-DNS lookups)[9] are contacted for a webpage download, as well as the amount of traffic to/from different IP addresses and hostnames. We define 20 semantically-different feature categories in this level.

7 Appendix 11 has a complete list of features extracted in this work along with the number of features in each group.

8 For packet direction, positive sign ('+') is used to represent incoming packets that sent from servers to clients while negative sign ('-') indicates outgoing packets sent by clients. Thus the range of packet size is [-1500, 1500], with a maximum transmission unit (MTU) of 1500.

9 The hostnames were derived after completion of the 20 download instances of the websites, which boils down to a gap of a few days for some websites. Hostnames can be more reliable when the gap is smaller.

Overall, we extract around 35,683 features, that can be grouped into 109 feature categories, from the TCP/IP headers of a web page download traffic.

Feature Reduction/Preprocessing High-dimensional feature spaces may be affected by huge computation cost and the *curse of dimensionality* problem in machine learning [37]. In order to control the number of features for feasible analysis, we performed each of the following preprocessing steps. Most of these are motivated by prior literature—details of experimental evaluations can be found in [42]:

1. Packets without payload (ACK packets) are removed from each instance—this not only aids in extracting burst-level features but also improves the classification accuracy [18, 20].
2. Packet size is rounded off to an increment of 8, while burst size is rounded off to an increment of 600—these values are obtained through experimental evaluations and help manage the tradeoff between controlling the number of features and information loss caused by rounding off.
3. For IP addresses, only the first three bytes of IP addresses are considered, and for hostnames, only the second-level domain names are considered—this helps control the number of features as well as improve robustness when multiple servers are used for the same service.

3.3 Communication Scenarios Considered

It is important to note that not all of the features identified in Section 3.2 may be available for a given webpage download. Indeed, due to increasing reliance on privacy-enhancing mechanisms, several features in all five levels (packet-level to IP address level) may be unavailable in reality. In this section, we identify eight different scenarios (summarized in Table 3) that differ in the type of information available from TCP/IP headers:

1. *S0: All TCP/IP features available.* The first scenario we consider is the baseline case, in which no privacy-enhancing mechanisms that influence TCP/IP headers have been applied when either visiting a website or collecting a trace [5–9]. This would include a majority of Internet traffic—that is transmitted using direct HTTPx connections between clients and web servers. Packet headers from all TCP transfers initiated are available, with the IP addresses and port numbers intact—features from all levels identified in Section 3.2 are available.
2. *S1: Anonymized IP address.* For some entities, access to packet traces is available only after IP addresses have been anonymized [4]. Therefore, for a given website, an IP address may be dynamically mapped to different random values in training data versus test data. Even more fundamentally, features derived from specific server IP addresses as well as reversed hostnames are no longer available (e.g., *unique hostname*, *server address count* and *transmitted bytes w.r.t. a specific IP address*).[10]
3. *S2: Encrypted tunnel (TCP sub-flows hidden).* Many users rely on services that allow them to connect to servers using an encrypted connection (e.g., SSH), or through the use of a proxy (e.g., VPN, Tor, etc.). IP addresses and contents are hidden from attackers. More importantly, information about TCP sub-flows is hidden; packets and bursts across all TCP transfers are merged and interleaved into one tunneled connection. Thus in *S2*, we are only able to observe a single TCP connection and features from *TCP-level*, *port-level*, and *IP address-level* are no longer available [5, 6, 10–15, 18, 33].
4. *S3-S4: Packet padding (packet sizes camouflaged).* The distribution and sequence of packet sizes has been shown to be one of the most informative features for website fingerprinting [11–13, 18]. Padding-based privacy-enhancing techniques aim at hiding actual packet size—for instance, by padding all transmitted packets to MTU size (PadToMTU). We next define two scenarios—*S3*, in which PadToMTU is employed within HTTPx transfers (*S3*: *S0*+PadToMTU) and *S4*, in which PadToMTU is employed within encrypted tunnels (*S4*: *S2*+PadToMTU) [11, 12, 14, 18].[11].
5. *S5: Packet timing camouflaged.* In order to consider techniques that camouflage inter-packet arrival times within anonymization networks such as Tor, we define *S5*, in which the inter-arrival times between consecutive padded packets is set to a fixed value (*S5*: *S4*+Fixed IAT) [14, 27–29, 43].
6. *S6-S7: Unidirectional traffic.* In some cases, incoming and outgoing traffic may be routed through different links between clients and servers (due to asymmetric routing or privacy concerns), which could imply that packet headers sent in only one

10 Note, however, that distribution-based features, such as *number of different IP addresses* and distribution of *transmitted bytes per TCP across different servers* can still be derived.

11 In Section 9, we also evaluate several recently-proposed countermeasures that have been shown to be more effective than PadToMTU [14]

direction may be available for website fingerprinting. We incorporate this situation by defining two scenarios (derived from *S0*)—*S6*, when only incoming traffic headers are available, and *S7*, when only outgoing traffic headers are available.

Table 3. Information available in different communication scenarios (P_d: packet direction, P_l: packet length, P_t: packet time, *IP/H*: server IP and/or hostname, *PN*: port number).

	P_d	P_l	P_t	IP/H	PN	TCP
S0:Baseline	✓	✓	✓	✓	✓	✓
S1:S0+Anonymized IP	✓	✓	✓		✓	✓
S2:Encrypted Tunnel	✓	✓	✓			
S3:S0+PadToMTU	✓		✓	✓	✓	✓
S4:S2+PadToMTU	✓		✓			
S5:S4+Fixed IAT	✓					
S6:S0+Incoming Only	✓	✓	✓	✓	✓	✓
S7:S0+Outgoing Only	✓	✓	✓	✓	✓	✓

3.4 Identifying Informative Features

The main goal of this paper is to identify *all* types of features that are notably informative for website fingerprinting in a given communication scenario. Given the large number of features we are considering (35,683), we believe this step needs to be guided by prudence. Our list of features is certain to contain several features that are highly correlated to each other. To realize our goal, it is important to select features that are not only highly informative, but are also uncorrelated to each other. For each communication scenario, we achieve the above using the following three steps:

1. *Filtering out less important features:* We first calculate the *importance* of each feature for the task of website fingerprinting. For this, we use Extra-Trees [44] from *scikit-learn* [45], with $max\_features=\sqrt{\#\,of\ features}$ and *entropy* [46] as the "impurity" criterion, to measure the Mean Decrease Impurity (MDI) importance [47].[12] We use cross-validation to determine number of trees in the forest (*n_estimators*). The larger the MDI importance, the more informative the feature is.[13] Our goal is to use the importance score to filter out features that are not important.

   Theoretically, MDI importance of a feature is equal to 0 if and only if the feature is irrelevant with totally randomized trees [47]. However, MDI-based importance scores may be biased in the presence of correlated features [52]: as the number of correlated features increases, the MDI for each individual inter-dependent feature decreases—this may mislead our importance analysis. Identifying and removing correlated features is not feasible for such a large feature space. Instead, we rank features according to their importance score and select the first n features that contribute to 99% of the total MDI of the feature list.[14] This filtering step leads to a huge reduction in the number of features—e.g., 5,852 important features in *S0* (see Table 4). In the second step below, we remove correlated features.
2. *Removing correlated features:* We next cluster together correlated features and choose one representative feature from each cluster. For clustering, we first normalize features to zero mean and one standard deviation and compute the Euclidean distance between each feature pair. We then perform average-linkage hierarchical agglomerative clustering [53] based on Euclidean distance by using *AgglomerativeClustering* in *scikit-learn*. To select the optimal number of clusters, we consider both supervised and unsupervised approaches—both yield consistent results, so we focus on the latter since it has a much lower computation cost. We evaluate the "goodness" of each clustering scheme based on average silhouette scores [54, 55]: the higher the value, the better the clustering scheme.

   After finding the optimal clusters for grouping correlated features, we select the feature with the maximum MDI importance from each cluster. With this, our final feature list is reduced to only relevant and uncorrelated features (Table 4). We feed this list into Extra-Trees, to re-compute the MDI importance for each feature without correlation bias.
3. *Grouping semantically-similar features:* In contrast to [33], we do not focus on analyzing the importance of each fine-grained feature (such as the *maximum packet size* and *median packet size*) in a traffic trace. Instead, we focus on ranking the importance of features with different semantics—defined according to the 109 semantically-similar feature categories in

12 We use Extra-Trees instead of Random Forest[48] to calculate MDI, since the former has been shown to be more computationally efficient and requires less memory—with comparable classification performance [44].

13 MDI importance has been used widely in gene selection [49–51] as a screening procedure for identifying important features.

14 We do not use a predefined threshold for MDI, since the presence of inter-dependent features can artificially deflate the MDI values and make us filter out even important features.

Table 4. Feature Selection Statistics (n_original: total number of original features; n_MDI: number of features after removing unimportant features; n_final: number of features after removing correlated features; n_catrgories: number of categories.)

	n_original	n_MDI	n_final	n_cat.
S0: Baseline	35,711	5,852	2,512	106
S1: S0+Anynomized IP	15,598	4,224	2,068	91
S2: Encrypted Tunnel	12,198	2,472	1,099	56
S3: S0+PadToMTU	31,612	5,401	1,944	107
S4: S2+PadToMTU	7,650	1,889	807	58
S5: S4+Fixed IAT	7,647	1,853	882	59
S6: Incoming Only	32,400	3,058	1,008	37
S7: Outgoing Only	22,315	3,253	948	41

Section 3.2. We believe that this is prudent given the goal of this paper—it is more informative to understand which feature category (such as *number of packets per TCP*, *packet size count*, or *initial 30 packets*) is important for website fingerprinting, rather than which statistical derivatives of a feature category is more important. Indeed, when a camouflaging technique is adopted, it is likely to hide all features within a feature category (such as padding each packet to a fix value in the traffic, rather than merely trying to cover the maximum packet size). Thus given the final feature list, we group features according to their semantically-defined feature category, and use summation of the weight of features within a group as the metric to measure importance of each category of features.[15]

Arguably, the order of the first two steps should be changed—first remove redundancy and then filter out less important features. However, the cost for computing distance between all feature pairs is $O(mn^2)$, while the cost for building a decision tree is $O(nmlgm)$, where n is the number of features and m refers to size of training samples. For computation efficiency, we prefer to calculate the MDI importance first.

Stability of Feature Selection To verify the stability of our feature selection methodology, we repeat the above procedure 30 times for each communication scenario—in each iteration, we randomly choose 2,000 websites with 16 instances from our dataset. We compute the standard deviation (across all iterations) in the importance of each feature category in the final list. The overall standard deviation is around 0.1%, which demonstrates the consistency of the selected features.

In what follows, we use the above methodology in each communication scenario. Due to space constraints and the prevalence of scenarios in the literature, we only present results in *S0*, *S2*, *S4* and *S6* in this paper—details for other scenarios are included in [42].

4 Baseline: HTTP*x* Transfers (Full TCP/IP Headers)

To begin with, we consider the most common scenario of visiting webpages using HTTPx transfers, including HTTP/1.x, HTTP/2.0, and HTTPS, without using enhanced privacy technologies. Although advanced technologies are available, a majority of Internet users do not use them due to either lack of privacy concerns or accessibility issues (several countries block VPN and advanced anonymization networks such as Tor due to legislation issues).[16] Thus, complete access to TCP/IP headers is available for website fingerprinting.

Related Work Website fingerprinting using HTTPx traces has been considered in previous studies [5–9], using different types of features. In the earliest work, Sun et al. used *HTTP object counts and sizes* to identify a website using statistical techniques based on the Jaccard coefficient [5]. Macia et al. extracted and used the *size and position of the root file and objects* corresponding to a given web page [9]. Gong et al. analyzed *round-trip times* using k-NN with the Dynamic Time Warping (DTW) metric to remotely fingerprint a website [6]. Miller et al. extracted *burst pairs from each TCP connection* and used a Bag of Gaussian classification scheme along with a Hidden Markov model [8]. Trevisan et al. analyzed two datasets with more than 790 million records in DNS requests/response and relied on *server IP addresses and lists of hostnames* to classify traffic [7]. The maximum number of wegpages considered across these prior work is 2,000, and the reported classification accuracy ranges from 50%-90%.

Although prior work has achieved high fingerprinting accuracy with HTTPx transfers, performance is expected to degrade as more websites are considered for classification [14]. An exhaustive feature analysis helps understand whether there are additional features that are significantly informative. We do this analysis next.

15 The semantically-similar category here is different from the correlated groups identified in step 2. There may be correlated features across semantically-defined feature categories, and there may be uncorrelated features within a given category.

16 https://www.theregister.co.uk/2017/07/11/russia_china_vpns_tor_browser/

New Informative Features Table 5[17] lists the feature categories (ranked by importance) identified for this scenario using the methodology of Section 3.4—note that these are *categories*, and include statistical derivatives and finer-granularity features. Feature categories that have not been discovered by prior work are:[18]

1. ***Transmitted bytes per TCP conn., w.r.t. top 20 most common hostnames and server IP addresses.*** Due to the influence of big players [56] in the Internet, more content is now being served using shared infrastructure such as Content Delivery Networks (CDNs) and cloud computing platforms. These two feature categories are defined with respect to the top 20 most commonly visited server hostnames and IP addresses in our dataset,[19] and indicate: (i) whether a website uses services provided by prominent providers, such as *Akamai*, *Google* and *Facebook*, to serve their content, and (ii) how much data is served from each of these (which may differ across websites). We find that these categories are fairly informative. We also find that hostnames identify a service more reliably than IP addresses—indeed, the latter may vary across visits and across client locations.
2. ***Cumulative packet sizes with/without direction.*** These feature categories are based on the sum of the first n packet sizes, with n ranging from 1 to 100. For example, given a packet sequence of [-100, 100, -70], the first 3 cumulative packet sizes are [100, 200, 270] and the first 3 cumulative packet sizes with direction are [-100, 0, -70]. These two feature categories capture a wealth of information on packet sizes, cumulative burst sizes, request/response ordering/direction. While these categories are less intuitive than simply examining packet sizes and burst sizes, our analysis reveals that they can be quite informative for website fingerprinting.
3. ***Number of bursts per TCP connection.*** These feature categories are based on number of bursts observed within each TCP connection, including incoming and outgoing bursts. Among them, the most important one is ratio of incoming bursts per TCP connection. Normally, the ratio should be 0.5, given the expected *request* and *response* communication pattern between clients and servers. However, variations may arise from: (i) the tail of a traffic trace getting truncated during data collection (which affects TCP connections that require more time to finish); or (ii) the server failing to receive the last *request* or the client failing to receive the last *response* due to congestion; In either case, the ratio of incoming bursts may change from 0.5 to $\frac{n}{2n-1}$, where $2n-1$ is total number of bursts observed.
4. ***Ratio of incoming bytes per TCP connection.*** This feature category is based on the ratio of incoming bytes to the total bytes sent in each TCP connection. This is likely to be influenced by both content (length) on a website (which determines the *response* sizes) and the transfer protocol configuration for application data (which determines the *request* sizes). Other categories related to transmitted bytes per TCP connection—including total/incoming/outgoing bytes per TCP connection—are also informative (cumulative importance of 2.23).
5. ***Initial packets in first TCP connection.*** This feature category includes the size and direction for the first 30 packets transmitted within the *first* TCP connection. While a similar feature has been used for Tor traffic analysis, it has not been used before for HTTPx analysis [22]. When visiting a website using HTTP/1.x, the first TCP connection carries the base page (index.html), whereas subsequent TCP connections fetch embedded objects. The base page typically describes the template of the website and the placement of different objects—which are more stable than the content (or the embedded objects). Hence, packet features based only on the first TCP connection are likely to be informative.
6. ***Initial bursts in first TCP connection.*** Similar to initial packets, initial bursts indicate the size and direction of the first 30 bursts in first TCP connection and capture uniqueness of the *request*/*response* pattern for each website.
7. ***Transmitted bytes per TCP connection w.r.t. port 443 and 80.*** These categories are based on the outgoing bytes in each TCP connection that are sent over HTTPS (443) and HTTP (80), respectively. Although some websites encrypt all of their traffic with HTTPS, others use a mix of HTTP and HTTPS. These features help fingerprint websites in terms of their HTTPS adoption and object sizes.
8. ***20 largest bytes per TCP connection.*** These feature categories are based on the 20 largest values

17 Tables of important features are truncated in this paper due to space limitation. Complete tables are included in [42].

18 Features that have been identified in prior work will not be discussed here due to space limitation.

19 A list of 20 most common server IP addresses and hostname in our dataset can be found at the end of Appendix 11.

of bytes transmitted per TCP connection, including total bytes and incoming/outgoing bytes. These are influenced by the number of *requests* that are sent in each TCP connection and their total size. In HTTP/1.0, each *request* requires its own TCP connection. In HTTP/1.1, reuse of TCP connections becomes possible—thus, more than one *request* may be sent in a persistent HTTP transfer (mostly in a non-pipelined synchronous manner, though) HTTP/2 allows more *requests* to be transferred over a single TCP connection in a pipelined fashion, thereby enabling asynchronous requests. Thus, this feature category captures not only information about website contents but also about the transfer protocol used by its server.

Classification Accuracy Gains Due to New Features We next evaluate how much improvement in website classification accuracy do the newly discovered features offer. For this, we classify websites using all features listed in Table 5. For comparison, we also classify websites using feature sets used in prior work (Table 1)—for completeness, we also include feature sets that have been used in (only) other communication scenarios. In order to control for performance variations due to the use of different machine learning algorithms by different prior work, we focus only on the feature sets they use and apply the *same* classifier (Extra-Trees) on top of each feature set. As mentioned in Section 3.1, the evaluations are conducted using data from 2,000 websites, each with 16 instances. We perform 10-fold cross-validation to obtain average accuracy. The results are summarized in Table 6 (our feature set is denoted as *Wfin*). We find that:

– As can be seen, classifiers with *packet size count* as features (*H*, *L* and *P*) outperform classifiers designed for Tor, including *CUMUL* and *k-FP*, since packet sizes are fixed in Tor and not considered as informative for classification. *FLSVM* that treats packets as a sequence and performs classification based on the edit distance between each pair of sequences is also able to achieve a high accuracy since it utilizes both packet size and ordering—however, it imposes a huge computation cost for calculating the distance between a test sequence and each training sequence. A high accuracy achieved with *T* further confirms that server IP address and hostnames can be quite informative for website fingerprinting [7].
– Our feature set achieves the highest classification accuracy. However, since the distribution of packet size alone is quite informative for identifying a website in *S0*, our feature set performs only somewhat better (1.8%) compared to the best-performing feature set used in prior work (*L*) [11]. Nonetheless, our analysis helps discover several other informative features that are quite powerful.

Table 5. Informative feature categories in *S0: HTTPx*. '**' indicates features that have not been discovered before.

1	unique packet size	25.686
2	preposition of first 300 incoming packets	8.328
3	packet size count	6.413
4	unique burst size	5.551
5	** ratio of incoming bytes per TCP conn. w.r.t. hostname	5.127
6	** initial 30 incoming in first TCP conn.	3.875
7	initial 30 packets	2.349
8	** initial 30 packets in first TCP conn.	2.333
9	initial 30 outgoing packets	2.201
10	unique server IP address	2.007
11	initial 30 incoming packets	2.006
12	** initial 30 outgoing in first TCP conn.	1.867
13	** ratio of incoming bytes per TCP conn. w.r.t. server IP address	1.746
14	burst size count	1.667
15	** outgoing bytes per TCP conn. w.r.t. hostname	1.517
16	** initial 30 outgoing bursts	1.237
17	** ratio of incoming bursts # per TCP conn.	1.107
18	** ratio of incoming bytes per TCP conn.	1.094
19	position of first 300 outgoing packets	1.006
20	** outgoing bytes per TCP conn. w.r.t. Port 443/80	0.976
21	position of first 300 incoming packets	0.851
22	** transmitted bytes per TCP conn. w.r.t. hostname	0.834
23	** outgoing bytes per TCP conn. w.r.t. server IP address	0.833
24	** cumulative size of first 100 packets	0.812
25	preposition of first 300 outgoing packets	0.809
26	** outgoing bytes per TCP conn.	0.775
27	size of outgoing bursts	0.766
28	** incoming bytes per TCP conn. w.r.t. hostname	0.735
29	** # of bursts per TCP conn.	0.699
30	** # of outgoing bursts per TCP conn.	0.671
31	** # of incoming bursts per TCP conn.	0.656
32	concentration of outgoing packets in first 2,000 packets	0.652
33	** cumulative size with direction of first 100 packets	0.612
34	** ratio of incoming bytes w.r.t. Port 443/80	0.552
35	size of first incoming burst in first TCP conn.	0.522
36	ratio of incoming packets # per TCP conn.	0.489
37	hostname count	0.467
38	** initial 30 bursts	0.44
39	ratio of incoming bursts size per TCP conn.	0.439
40	alternative concentration of outgoing packets	0.432

Why Extra-Trees? For deciding which machine learning classifier to use in the evaluations above, we considered SVM [57], *k*-NN and Extra-Trees. The classification accuracy achieved with Extra-trees was, on average, about 15% higher than that achieved with *k*-NN using our dataset. Compared to SVM, Extra-Trees was also more computationally efficient and consumed less memory with the large number of features we consider. Appendix 12 also includes evaluations of the state of the art using classifiers used in the original work.

Table 6. Classification accuracy achieved with different feature sets proposed in the state of the art (*indicates the target scenario in the original work).

	H	L	P	Vng++	T	DTW	CUMUL	FLSVM	k-FP	Wfin
S0: Baseline	*95.37	*96.16	95.67	79.09	*92.61	*12.70	76.83	90.13	*87.27	97.96
S1: Anonymized IP					N/A					97.73
S2: Encrypted Tunnel			*95.68	*15.61						97.41
S3: S0+PadToMTU	8.29	35.00	62.07	58.52			*76.78	65.10		97.54
S4: S2+PadToMTU			11.36	9.98						96.83
S5: S4+Fixed IAT			11.41	9.77		11.20			85.45	95.44
S6: Incoming Only	93.74	93.99	91.09	13.44	92.61	N/A	22.61	81.82	70.49	96.70
S7: Outgoing Only	94.08	94.64	94.95	24.33		12.70	27.52	90.13	58.59	96.76

5 Use of Encrypted Tunnels & Proxies (TCP Flows Hidden)

In order to protect users' browsing activities from eavesdropping, a number of privacy enhancing technologies have been devised, such as virtual private networks (VPN), simple SSL proxies and OpenSSH tunnels [2]. As users become more security-savvy, the growing popularity of these privacy mechanisms has greatly enhanced the user security experience on the Internet [58]. For example, OpenSSH has been integrated into many operation systems and products, such as Linux, Mac OS X Version 10.1 and later, Cygwin and Nokia IPSO [59]. When a user transfers data over an encrypted tunnel to servers (e.g., OpenSSH) or connects through a proxy (e.g., OpenVPN [1]), packets from different TCP connections are merged into one tunneled connection (server IPs and TCP information is hidden). We next study this scenario (*S2*).

Related Work Website fingerprinting under encrypted tunnels has received a lot of attention and several informative features have been identified [5, 10–15, 18, 33]. To our knowledge, the earliest work was by Bissias et al., who proposed two features: *inter-arrival time* and *size of each packet*, and used cross-correlation to measure similarity of traffic from different websites [10]. Later, Liberatore et al. [11] and Herrmann et al. [12] validated the importance of *packet size count*. Lu et al. considered *sequence of packets* as strings to measure their edit distance [13]. Dyer et al. utilized three coarse-grained features: *total transmission time*, *per-direction bandwidth*, and *burst size* information with the Naive Bayes Classifier [14]. Feghhi et al. used only *packet timing* information in the uplink direction with a variant of the Dynamic Time Warping distance metric [15].

New Informative Features Table 7 lists the most important feature categories in scenario *S2*. Most of these—such as *unique packet size* [11, 12], *initial packets* and *packet position* [22]—have already been used in prior work. With TCP sub-flows being hidden, importance of feature categories related to packet size is increased (compared to *S0*). Due to the interleaving of packets from multiple TCP connections in an encrypted tunnel, burst size can no longer be used to reliably infer *request/response* patterns when loading a web page—thus the overall importance of burst-level features is greatly reduced. For example, importance of *unique burst size* is decreased from 5.55 to 0.64, while *burst size count* is no longer in the list.

Table 7. Most informative features in *S2: Encrypted Tunnel*. '**' indicates features not been discovered before.

1	unique packet size	52.264
2	packet size count	13.528
3	preposition of first 300 incoming packets	12.714
4	initial 30 packets	2.722
5	initial 30 outgoing packets	1.952
6	position of first 300 outgoing packets	1.611
7	initial 30 incoming packets	1.48
8	concentration of outgoing packets in first 2,000 packets	1.351
9	position of first 300 incoming packets	1.189
10	preposition of first 300 outgoing packets	1.141
11	alternative concentration of outgoing packets	1.099
12	** cumulative size of first 100 packets	0.901
13	** average inter-arrival time of first 20 packets	0.736
14	** cumulative size with direction of first 100 packets	0.73
15	unique burst size	0.647
16	size of outgoing bursts	0.522
17	** average inter-arrival time of first 20 outgoing packets	0.496
18	** average inter-arrival time of first 20 incoming packets	0.491
19	concentration of first 30 outgoing packets	0.373
20	ratio of incoming packets # per TCP conn.	0.363
21	# of packets per TCP conn.	0.306
22	** # of packets in a burst count	0.301
23	** initial 30 incoming bursts	0.286
24	** initial 30 outgoing bursts	0.272
25	# of outgoing packets per TCP conn.	0.266
26	** initial 30 bursts	0.253
27	** # of packets in incoming burst count	0.231
28	interpolant of cumulative packet size	0.227
29	size of incoming bursts	0.221
30	alternative outgoing packets per second	0.215

Compared to *S0*, we also note the increased importance of features related to the inter-arrival time between subsequent packets (***inter-arrival times of first 20 incoming/outgoing packets***). These features are affected by object sizes and the number of parallel TCP connections initiated within encrypted tunnels.

Classification Accuracy Gains We next evaluate the classification accuracy yielded by the newly discovered features, and compare it to features used in prior work (Table 6). Since encrypted tunnels do not hide actual packet size, performance of classifiers that use *packet size count* as features, including H, L and P, remains the same. With TCP sub-flows being hidden, accuracy achieved with *Vng++* [14] reduces significantly from 79.90% to 15.61%— such a significant drop is mainly due to the decrease of importance of *burst size count* in encrypted tunnels. Overall, accuracy achieved with the best-performing features from the state-of-the-art (L), which uses raw packet size count as features, is only somewhat lower compared with *Wfin* (around 1.3%), which further validates the informativeness of packet size for website fingerprinting. Evaluation results with other datasets are included in Section 12.1.

6 Padding-based Camouflaging

The distribution and sequence of packet sizes has been demonstrated as one of the most informative features for website fingerprinting in both prior work [11, 12, 14, 18] and our work. In this section, we consider padding-based privacy-enhancing techniques that hide actual packet sizes. We evaluate these both within HTTPx transfers (*S3*) as well as encrypted tunnels (*S4*). In addition, we examine the influence of techniques that camouflage inter-packet arrival times between consecutive padded packets (*S5*). Due to space constraints, evaluations with *S3* and *S5* are included only in [42].

6.1 Encrypted Tunnel + PadToMTU

The next scenario we consider (*S4:S2+PadToMTU*) resembles *Tor*, since: (i) all packets are padded to a fix size, and (ii) IP addresses, port numbers, and TCP sub-flows are all hidden in TCP/IP headers, which is characteristic of Tor due to onion routing. Tor [16] aims at protecting users' surfing activities from eavesdropping by routing data through several relay nodes, as well as using layered encryption of the content. It has been regarded as one of the most secure networking technology with more than 2 million current daily users [3]— consequently, it has received a lot of attention in traffic analysis literature in the past few years.

In *S4*, only limited information from packet-level and burst-level is available and we extract around 7,600 initial features (Table 4) for importance analysis to understand whether there are other features that can be used for website fingerprinting when actual packet size is camouflaged within encrypted tunnels.

Related Work Tor traffic analysis has received a lot of attention and several informative features have been identified in prior research. Murdoch et al. showed that an attacker with control over both-ends of a Tor connection can compromise the client's privacy by analyzing *timing characteristics* [17]. Panchenko et al. used *packet size count* and several additional features, such as *total transmitted bytes* and *HTML Marker*, with Support Vector Machines (SVM) [18]. Yu et al. took a fundamentally different approach by utilizing *browsing time intervals* to infer the length of web page and managed to identify 1,000 accessed pages using a Hidden Markov Model (HMM) and the Viterbi algorithm [19]. Cai et al. employed the *size and direction of packets* as features along with distance-based SVM [20]. Based on Cai et al. [20], Wang et al. focused on *cell* as a unit of data (rather than TCP/IP packets) with 100 websites, to define a new metric for characterizing the similarity between two traffic instances [21]. Apart from general size and timing features, Wang et al. added several features, such as *concentration of outgoing packets* and *packet ordering*, to identify 100 monitored web pages with k-NN [22]. Panchenko et al. combined four basic features identified in [18] together with features extracted from *cumulative sum of packet sizes* for analyzing 100 websites using Tor [23]. More recently, based on comments made by Juarez et al. [60], Wang et al. attempted to remove assumptions made in previous website fingerprinting and gap the bridge between laboratory setups and realistic conditions [24]. Finally, Abe et al. [25] and Rimmer et al. [26] investigated the application of deep learning for website fingerprinting in Tor traffic.

New Informative Features Table 8 lists the most informative features for scenario *S4*. With even packet sizes being hidden in encrypted tunnels, the importance of features that are accessible is increased (compared to *S2*)—the ranking of importance remains relatively stable, though. For example, importance of *initial (incoming/outgoing) packets* is increased from 6.33 to 16.58, and that of *inter-arrival time of first 20 (incoming/outgoing) packets* is increased from 1.83 to 6.12. In total, the importance of features that have not been discovered in prior studies is 22.18[20] (compared to 6.28 for *S2*), suggesting that the state of the art in Tor traffic analysis was quite far from having discovered most powerful features.

20 This is computed as the sum of importance of all feature categories marked with '**' in Table 8.

Table 8. Most informative features in *S4: S2+PadToMTU*. '**' indicates features that have not been discovered before.

1	preposition of first 300 incoming packets	24.039
2	concentration of outgoing packets in first 2,000 packets	7.417
3	initial 30 incoming packets	5.906
4	alternative concentration of outgoing packets	5.673
5	** cumulative size with direction of first 100 packets	5.65
6	initial 30 packets	5.611
7	position of first 300 outgoing packets	5.424
8	position of first 300 incoming packets	4.413
9	initial 30 outgoing packets	4.197
10	preposition of first 300 outgoing packets	4.196
11	** average inter-arrival time of first 20 packets	2.38
12	unique burst size	1.978
13	** average inter-arrival time of first 20 incoming packets	1.896
14	** average inter-arrival time of first 20 outgoing packets	1.824
15	** initial 30 outgoing bursts	1.761
16	** initial 30 bursts	1.3
17	number of outgoing packets per second	1.205
18	** # of packets in incoming burst count	1.163
19	** # of packets in a burst count	1.108
20	alternative outgoing packets per second	0.934
21	** outgoing burst duration	0.878
22	# of outgoing packets per TCP conn.	0.864
23	** initial 30 incoming bursts	0.862
24	ratio of incoming packets # per TCP conn.	0.842
25	concentration of first 30 outgoing packets	0.815
26	** burst duration	0.812
27	burst size count	0.785
28	** # of packets in outgoing burst	0.65
29	size of incoming bursts	0.591
30	alternative packets per second	0.558
31	concentration of last 30 incoming packets	0.463
32	interpolant of cumulative packet size	0.438
33	** # of packets in each burst	0.432
34	concentration of last 30 outgoing packets	0.428
35	number of packets per second	0.428
36	number of incoming packets per second	0.372
37	** # of packets in outgoing burst count	0.358
38	** incoming burst duration	0.34

Classification Accuracy Gains We evaluate the website classification accuracy achieved using the newly discovered features, and compare it to features used in prior work (Table 6). With packet size being camouflaged in encrypted tunnels, classification accuracy of classifiers that use burst size as features (*P* and *Vng++*) is further decreased. In this case, *Wfin* still outperforms the best-performing feature sets from the state-of-the-art by around 9%. Furthermore, the overall classification accuracy of *Wfin* is only slightly lower than that achieved in *S2*—this indicates that despite the absence of the powerful features related to packet sizes, our analysis methodology helps uncover several other features that are collectively nearly as informative.

7 Unidirectional Traffic Headers

In some cases, incoming and outgoing traffic may be routed through different links between client and server due to asymmetric routing or security concerns (e.g., [61–63]). Thus, only the incoming or outgoing traffic may be accessible on the link being monitored. We next consider the scenarios with only incoming (*S6:S0+Incoming Only*) and outgoing (*S7:S0+Outgoing Only*) HTTPx traffic available and study website fingerprinting with features derived from unidirectional traffic headers—due to space constraints, *S7* is included only in [42]. Note that while incoming traffic contains *responses* sent from servers to clients for displaying objects on a website, outgoing traffic contains the *requests* from clients to servers and reveals the length of *url* associated with each object. In prior work, Feghhi et al. demonstrated the efficiency of using uplink time information of packets for website fingerprinting [15]. However, to the best of our knowledge, *there is no prior work that studies/targets website fingerprinting in unidirectional scenarios.*

7.1 Incoming Traffic Only

Informative Features Table 9 lists the most informative feature categories for scenario *S6*. Due to the absence of *request* traffic, features related to burst size (such as *unique burst size* and *burst size count*) are no longer based on segmentation of *request/response* patterns in TCP transfers. Instead, they simply denote the *total* incoming bytes per TCP connection. Additional timing-based features are significant in this scenario:

1. ***Duration of each TCP connection.*** These features compute the duration (time gap between last and first packets) of each TCP connection, relative to the duration of first TCP connection. It is likely to be influenced by object sizes, server-side transfer protocol configuration, and the inter-arrival time between packets in each TCP connection.
2. ***Relative end time of each TCP connection.*** This feature computes the end time of each TCP connection, relative to the start time of the *first* TCP connection.

Classification Accuracy Gains We evaluate the classification accuracy yielded by the features we have discovered for scenario *S6*—although no prior work has targeted this scenario before, we use features proposed by others for other communication scenarios. Table 6 summarizes the results. We find that compared with *S0*, the availability of only incoming traffic does not severely degrade performance of classifiers that rely on packet sizes—indeed, packet-size based features alone carry enough information for website fingerprinting. However, as expected, the performance of classifiers that rely on features describing the interleaving pattern between incoming and outgoing packets, such as *k-FP* and *CU-*

MUL, degrades significantly. Overall, *Wfin* outperforms other feature sets and the gap is around 2.7% compared with the state of the art (*L*).

Table 9. Most informative features in *S6: Incoming Traffic Only*. '**' indicates features that have not been discovered before.

1	unique packet size	44.566
2	unique burst size	12.356
3	unique server IP address	10.365
4	** initial 30 incoming in first TCP conn.	5.704
5	packet size count	5.425
6	initial 30 incoming packets	4.16
7	** incoming bytes per TCP conn. w.r.t. hostname	2.404
8	** cumulative size of first 100 packets	2.183
9	** incoming bytes per TCP conn.	1.669
10	** average inter-arrival time of incoming packets per TCP conn.	1.267
11	** relative duration of each TCP conn.	1.181
12	** count of packet number in incoming burst	0.928
13	server port count	0.917
14	** incoming bytes per TCP conn. w.r.t. Port 443/80	0.909
15	** average inter-arrival time of first 20 incoming packets	0.84
16	** incoming bytes per TCP conn. w.r.t. server IP address	0.71
17	** relative end time of each TCP conn.	0.651
18	hostname count	0.576
19	** initial 30 incoming bursts	0.483
20	# of TCP conn.	0.481
21	** 20 largest bytes per TCP conn. w.r.t. hostname	0.457
22	** # of packets in incoming burst	0.434

8 Open-world Scenario Evaluation

It is important to consider "open-world" evaluation scenarios, in which a client may visit a large world size that includes unmonitored web pages that have never been seen by the classifier during training. We use the open-world Tor dataset collected by Wang et al. [22] and compare *Wfin* with one of the state-of-the-art classifiers—*k-FP* [33]. This dataset is composed of 100 monitored web pages, each with 90 instances, and 8,900 non-monitored websites each with a single instance. With the same experimental setup as in [33], the classifier is trained on 60 instances for each of the 100 monitored web pages and 3500 unmonitored web pages, and the client can browse to any of those monitored web pages or to 5,000 unmonitored ones.

The aim of the classifier is to determine whether the client is visiting one of the monitored web pages and establish which one. Since the dataset is imbalanced, the performance is measured in terms of true positive rate[21] (TPR), false positive rate[22] (FPR), and Bayesian detection rate[23] (BDR), and is shown in Table 10: *Wfin* is able to correctly classify a monitored web page 92% of the time (compared to 88% with *k-FP*).

21 *Pr* that a monitored page is classified correctly, a.k.a. recall.

22 *Pr* that an unmonitored page is classified as monitored.

23 *Pr* that a page is correctly classified when the classifier recognized it as a monitored page, also called precision.

Table 10. Open-world performance (%) of *k-FP* and *Wfin* with Wang et al. dataset [22].

	TPR	FPR	BDR
***k*-FP**	0.88±0.01	0.005±0.001	0.997
Wfin	0.92±0.01	0.006±0.001	0.996

9 Recent Countermeasures

Several countermeasures have been proposed over the past decade to camouflage informative features such as packet sizes. We next evaluate the performance of *Wfin* in the presence of several of these, including traffic morphing [30], Decoy [18], BuFLO [14], Tamaraw [28] and Walkie-Talkie [31]. It is important to note that the *Wfin* features used in this section are the same as those that were derived in the corresponding scenarios in Sections 4-7—Wfin has not been re-derived with these new countermeasures in place.[24] More detailed evaluations will be considered in future work.

9.1 Our Dataset

Encrypted Tunnel (S2) First, we evaluate the performance of *Wfin* against countermeasures with the 2,000 websites in our dataset in *S2* and compare with *CUMUL* and *k-FP*, which are two state of the art classifiers. The outcomes are shown in Table 11. As can be seen, *Wfin* outperforms both *CUMUL* and *k-FP* in face of countermeasures due to its high diversity in the feature set. The difference ranges between 0.2% to 20%.

Table 11. Closed-world accuracy (%) against countermeasures with our dataset in *S2*.

	CUMUL	*k-FP*	*Wfin*
Morphing	74.78	77.41	82.17
Decoy	8.18	12.93	33.24
BuFLO (τ=0, β=0.08)	6.97	6.63	7.19

Encrypted Tunnel + PadToMTU (S4) We next evaluate performance of *Wfin* against countermeasures in *S4* (Table 12). *Wfin* still outperforms both *CUMUL* and *k-FP* and the gap ranges between 0.5% to 7%.

Table 12. Closed-world accuracy (%) against countermeasures with our dataset in *S4*.

	CUMUL	*k-FP*	*Wfin*
Morphing	84.19	82.05	91.82
Decoy	8.35	11.63	16.61
BuFLO (τ=0, β=0.08)	4.54	4.9	5.46

24 Some evaluations that re-derive Wfin are included in [42].

9.2 Wang et al. Tor Dataset [22]

Next, we compare performance of *Wfin* and *k-FP* against four types of countermeasures shown in Table 13 in both closed-world and open-world scenarios with the Tor dataset [22].

Closed-world Scenario In closed-world scenario, we use 90 instances from each of the 100 monitored web pages for training and testing with 10-fold cross validation. In general, *Wfin* outperforms *k-FP* in face of countermeasures due to its high diversity in feature types. The most efficient countermeasure is Tamaraw [28], which aims at hiding packet size and ordering by sending packets in fixed size and injecting junk packets.

Table 13. Closed-world accuracy against countermeasures with Wang et al. dataset [22].

	k-FP	*Wfin*
Morphing	0.91	0.93
Decoy	0.35	0.53
BuFLO (τ=0, β=0.08)	0.21	0.24
Tamaraw (β_{out}=0.04, β_{in}=0.012)	0.10	0.11

Open-world Scenario In open-world scenario, we measure the TPR, FPR and BDR of *k-FP* and *Wfin* against countermeasures under the same setup in Section 8. The outcomes are shown in Table 14. In face of *BuFLO*, *Wfin* is able to correctly classify a website 24% of the time, while *k-FP* is able to correctly classify with 5% probability. In the worst case against *Tamaraw*, *Wfin* still outperforms *k-FP* around 8% in terms of *TPR*. Overall, *Wfin* is able to achieve higher TPR, lower FPR and higher BDR in face of different countermeasures in open-world scenario compared with *k-FP*.

Table 14. Open-world performance of *k-FP* and *Wfin* against countermeasures with Wang et al. dataset [22].

	k-FP			*Wfin*		
	TPR	FPR	BDR	TPR	FPR	BDR
Morphing	0.86	0.006	0.996	0.92	0.007	0.996
Decoy	0.18	0.033	0.915	0.51	0.010	0.991
BuFLO	0.05	0.0	1.0	0.24	0.0	1.0
Tamaraw	0.02	0.0	1.0	0.10	0.0	1.0

9.3 Walkie-Talkie [31]

Finally, we evaluate the performance of *Wfin* and *k-FP* against a fairly recent countermeasure—Walkie-Talkie [31]—with two Tor datasets collected by Wang et al. [31] in open-world scenario (shown in Table 15). The experimental setup is the same as in previous sections and the datasets consist of 100 websites, each with 100 instances, and 10,000 websites each with one instance.[25] Without Walkie-Talkie, *Wfin* outperforms *k-FP* by around 7% in terms of TPR. The gap is further increased to 15% with the Tor dataset protected by Walkie-Talkie.

Table 15. Open-world performance of *k-FP* and *Wfin* against Tor (Undefended) and Tor with Walkie-Talkie [31] (Defended).

	Undefended			Defended		
	TPR	FPR	BDR	TPR	FPR	BDR
k-FP	0.76	0.113	0.902	0.20	0.027	0.899
Wfin	0.83	0.018	0.989	0.35	0.012	0.982

Furthermore, compared with the results obtained in our dataset (*S4* in Table 6), the performance gain achieved with *Wfin* is less obvious with the Tor datasets. One possible explanation is the number of classes for classification: 2,000 websites are considered in our dataset while the Tor datasets have only 100 websites. As Dyer et al. [14] pointed out, performance of classifiers degrades with the increment of number of classes. Specifically, performance of classifiers with well-defined features is expected to degrade slower compared with those use coarse-grained features. Thus the gap between *Wfin* and *k-FP* may be further enlarged when we consider more websites. We leave it as a future work to collect a large-scale Tor dataset and evaluate the performance with *Wfin* and other classifiers.

10 Discussion & Conclusions

In this paper, we conduct an exhaustive feature importance analysis in eight different communication scenarios for website fingerprinting. We (i) discover several previously-unknown informative feature categories, and (ii) outperform the features used in the state of the art with the feature set derived from our methodology across *all* eight communication scenarios. More specifically, by analyzing the importance of different feature categories across all scenarios, we discover the following:

– When available in HTTPx (*S0/1*), encrypted tunnel (*S2*) and unidirectional scenario (*S6/7*), features derived from actual packet size alone, such as *packet size count* and *unique packet size*, are informative enough to achieve performance comparable to state of the art involving complex feature sets.

25 The reason why we do not evaluate Walkie-Talkie in our dataset and Wang et al. [22] dataset is because Walkie-Talkie requires browsers to work in half-duplex mode and both datasets are collected in the normal full-duplex mode.

- When server IP addresses are visible, more informative features can be extracted from server IP addresses and hostnames than considered by the state of the art [7, 8]—such as *transmitted bytes per TCP conn. w.r.t. server IP addresses/hostnames*—to fingerprint a website when TCP sub-flows are not hidden (*S0/3*). The performance is promising even in unidirectional scenarios (*S6/7*).
- When packet size is hidden in HTTPx (*S3*), importance of features extracted from server IP addresses and hostnames, and from TCP-level, such as *initial packets in first TCP connection*, *ratio of incoming bytes per TCP connection* increases—features extracted from first TCP connection in TCP-level are among the most informative ones.
- When TCP sub-flows are multiplexed via encrypted tunnels (*S2/4*), burst-level features that have been widely used in prior studies are no longer as informative [14, 18, 22].
- With packet size is hidden in encrypted tunnels (*S4*), features describing packet ordering become most informative—such as *preposition of incoming packets*, *initial packets* and *cumulative packet size with direction*. This suggests the potential for deploying deep learning in identifying websites in Tor [25, 26]—since deep learning methods have a powerful ability to exploit input sequence ordering.
- Top informative features across *all* communication scenarios include *position/preposition of packets*, *initial packets*, *concentration of outgoing packets* and *cumulative packet size with direction*. Those features are extracted from both packet size and direction, and reveal packet ordering. Thus they are more robust across *all* communication scenarios.

We believe this work is an important step in a new direction—that of searching for the limits of learnability. Our feature selection methodology can also be applied to other fields of traffic analysis—such as fingerprinting of applications, protocols, online user activity, and type of content [58, 64–66]—in order to identify the most informative features and better serve their goals.

However, there are several practical issues that need to be explored in future work to bridge the gap between our experimental findings and the real world:

- *Traffic segmentation.* Most Internet links aggregate traffic from multiple sources and clients. Before the traffic of a given web page download can be fed to a classifier, it must be extracted from an aggregated traffic trace—indeed, current fingerprinting techniques all assume and rely on this preprocessing step. However, there is only scant prior work on providing a solution—Feghhi et al. [15] and Wang et al. [24] have utilized timing information and machine learning techniques, respectively. Nevertheless, this still remains a challenging open issue.
- *Influence of cache.* While Miller et al. [8] studied the effect of cache on website fingerprinting with HTTPS traffic, most of prior studies have chosen to simply disable cache during data collection—when objects are cached locally, less packets are observed in the traffic trace. Thus, an open question for this paper is how will caching impact feature importance. It is also worth noting that the influence of caching depends strongly on the time gap between consecutive visits to a web page—this adds diversity to real world traffic, which must be incorporated in training and testing conditions.
- *Diverse browser platforms.* There is significant diversity in the browser platforms used by Internet clients. It is important to study the influence of browsers on website fingerprinting—especially when the browsers platforms used in the training and testing datasets are different.
- *HTTP/2.* With the growing adoption of HTTP/2 [67–69], features such as server push and pipelined/parallel download of objects are becoming increasingly commonplace. We speculate these new features will make website fingerprinting more challenging under the HTTPx scenario since they directly influence several of the important traffic features we have identified. Understanding this influence remains an important future work.
- *Dataset Bias.* When fingerprinting websites in *different* datasets, we observe different performance gains of *Wfin* over the state-of-the-art. One remaining problem that has not attracted much attention for website fingerprinting is *dataset bias*. Specifically, how can datasets that have been studied in this field before be used to generalize to new unseen samples even with different training and test collections? How will different datasets affect importance of features? What contributes more to performance gains—tuning to specific datasets or new features/learning algorithms? Understanding the impact of these questions is an important avenue for future work.

References

[1] Markus Feilner. *OpenVPN: Building and integrating virtual private networks*. Packt Publishing Ltd, 2006.

[2] Girish Venkatachalam. The openssh protocol under the hood. *Linux Journal*, 156, 2007.

[3] Tor metrics-user. https://metrics.torproject.org.

[4] Interception and disclosure of wire, oral, or electronic communications prohibited. http://www.law.cornell.edu/uscode/text/18/2511.

[5] Qixiang Sun, Daniel R Simon, Yi-Min Wang, Wilf Russell, Venkata N Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 19–30. IEEE, 2002.

[6] Xun Gong, Negar Kiyavash, and Nikita Borisov. Fingerprinting websites using remote traffic analysis. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 684–686. ACM, 2010.

[7] Martino Trevisan, Idilio Drago, Marco Mellia, and Maurizio M Munafo. Towards web service classification using addresses and dns. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International*, pages 38–43. IEEE, 2016.

[8] Brad Miller, Ling Huang, Anthony D Joseph, and J Doug Tygar. I know why you went to the clinic: Risks and realization of https traffic analysis. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 143–163. Springer, 2014.

[9] Gabriel Maciá-Fernández, Yong Wang, Rafael Rodríguez-Gómez, and Aleksandar Kuzmanovic. Isp-enabled behavioral ad targeting without deep packet inspection. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[10] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted http streams. In *International Workshop on Privacy Enhancing Technologies*, pages 1–11. Springer, 2005.

[11] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263. ACM, 2006.

[12] Dominik Herrmann, Rolf Wendolsky, and Hannes 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. ACM, 2009.

[13] Liming Lu, Ee-Chien Chang, and Mun Chan. Website fingerprinting and identification using ordered feature sequences. *Computer Security–ESORICS 2010*, pages 199–214, 2010.

[14] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 332–346. IEEE, 2012.

[15] Saman Feghhi and Douglas J Leith. A web traffic analysis attack using only timing information. *IEEE Transactions on Information Forensics and Security*, 11(8):1747–1759, 2016.

[16] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[17] Steven J Murdoch and George Danezis. Low-cost traffic analysis of tor. In *Security and Privacy, 2005 IEEE Symposium on*, pages 183–195. IEEE, 2005.

[18] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 103–114. ACM, 2011.

[19] Shui Yu, Wanlei Zhou, Weijia Jia, and Jiankun Hu. Attacking anonymous web browsing at local area networks through browsing dynamics. *The Computer Journal*, 55(4):410–421, 2012.

[20] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob 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. ACM, 2012.

[21] Tao Wang and Ian Goldberg. Improved website fingerprinting on tor. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 201–212. ACM, 2013.

[22] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *USENIX Security*, pages 143–157, 2014.

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

[24] Tao Wang and Ian Goldberg. On realistically attacking tor with website fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016(4):21–36, 2016.

[25] Kota Abe and Shigeki Goto. Fingerprinting attack on tor anonymity using deep learning. *Proceedings of the Asia-Pacific Advanced Network*, 42:15–20, 2016.

[26] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. Automated feature extraction for website fingerprinting through deep learning. *arXiv preprint arXiv:1708.06376*, 2017.

[27] Xiang Cai, Rishab Nithyanand, and Rob Johnson. Cs-buflo: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 121–130. ACM, 2014.

[28] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 227–238. ACM, 2014.

[29] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security*, pages 27–46. Springer, 2016.

[30] Charles V Wright, Scott E Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, volume 9, 2009.

[31] Tao Wang and Ian Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. 2017.

[32] Giovanni Cherubin, Jamie Hayes, and Marc Juarez. Website fingerprinting defenses at the application layer. *Proceedings on Privacy Enhancing Technologies*, 2017(2):186–203, 2017.

[33] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security Symposium*, pages 1187–1203, 2016.

[34] Andrew Ng. Machine learning and ai via brain simulations, 2013.

[35] Giovanni Cherubin. Bayes, not naive: Security bounds on website fingerprinting defenses. *arXiv preprint arXiv:1702.07707*, 2017.

[36] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[37] David L Donoho et al. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture*, 1:32, 2000.

[38] Alexa Amazon. Top 1m websites by alexa. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[39] Stefan Saroiu, Krishna P Gummadi, Richard J Dunn, Steven D Gribble, and Henry M Levy. An analysis of internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36(SI):315–327, 2002.

[40] Alexandre Gerber, Joseph Houle, Han Nguyen, Matthew Roughan, and Subhabrata Sen. P2p the gorilla in the cable. *National Cable & Telecommunications Association (NCTA) 2003 National Show*, pages 8–11, 2003.

[41] Subhabrata Sen and Jia Wang. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Transactions on Networking (ToN)*, 12(2):219–232, 2004.

[42] Junhua Yan and Jasleen Kaur. Feature selection for website fingerprinting. Technical Report 18-001, 2018.

[43] Rishab Nithyanand, Xiang Cai, and Rob Johnson. Glove: A bespoke website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 131–134. ACM, 2014.

[44] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[46] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[47] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.

[48] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[49] Hongying Jiang, Youping Deng, Huann-Sheng Chen, Lin Tao, Qiuying Sha, Jun Chen, Chung-Jui Tsai, and Shuanglin Zhang. Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. *BMC bioinformatics*, 5(1):81, 2004.

[50] Kathryn L Lunetta, L Brooke Hayward, Jonathan Segal, and Paul Van Eerdewegh. Screening large-scale association study data: exploiting interactions using random forests. *BMC genetics*, 5(1):32, 2004.

[51] Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

[52] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

[53] Mee Young Park, Trevor Hastie, and Robert Tibshirani. Averaged gene expressions for regression. *Biostatistics*, 8(2):212–227, 2006.

[54] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[55] Laura Toloşi and Thomas Lengauer. Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27(14):1986–1994, 2011.

[56] Vinicius Gehlen, Alessandro Finamore, Marco Mellia, and Maurizio M Munafò. Uncovering the big players of the web. In *International Workshop on Traffic Monitoring and Analysis*, pages 15–28. Springer, 2012.

[57] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[58] Charles V Wright, Fabian Monrose, and Gerald M Masson. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research*, 7(Dec):2745–2769, 2006.

[59] Openssh users. https://www.openssh.com/users.html.

[60] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 263–274. ACM, 2014.

[61] Malcolm Stevens and Michael Pope. Data diodes. Technical report, 1995.

[62] Jill Slay, B Turnbull, et al. The uses and limitations of unidirectional network bridges in a secure electronic commerce environment. In *Proceedings of the Fourth International Network Conference 2004 (INC2004)*, page 129. Lulu. com, 2004.

[63] Douglas W Jones and Tom C Bowersox. Secure data export and auditing using data diodes. *technology*, 6:7, 2006.

[64] Fan Zhang, Wenbo He, Xue Liu, and Patrick G Bridges. Inferring users' online activities through traffic analysis. In *Proceedings of the fourth ACM conference on Wireless network security*, pages 59–70. ACM, 2011.

[65] Sean Sanders and Jasleen Kaur. Can web pages be classified using anonymized tcp/ip headers? In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2272–2280. IEEE, 2015.

[66] Hasan Faik Alan and Jasleen Kaur. Can android applications be identified using only tcp/ip headers of their launch time traffic? In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 61–66. ACM, 2016.

[67] Matteo Varvello, Kyle Schomp, David Naylor, Jeremy Blackburn, Alessandro Finamore, and Konstantina Papagiannaki. Is the web http/2 yet? In *International Conference on Passive and Active Network Measurement*, pages 218–232. Springer, 2016.

[68] Jawad Manzoor, Idilio Drago, and Ramin Sadre. The curious case of parallel connections in http/2. In *Network and Service Management (CNSM), 2016 12th International Conference on*, pages 174–180. IEEE, 2016.

[69] Jawad Manzoor, Idilio Drago, and Ramin Sadre. How http/2 is changing web traffic and how to detect it. In *Network Traffic Measurement and Analysis Conference (TMA), 2017*, pages 1–9. IEEE, 2017.

11 A complete list of features

In this part, we present the complete list of features have extracted in this work in a format of *<category index - description (number of features)>*. Statistical derivatives includes *sum*, *max*, *min*, *mean*, *25/50/75/90 percentile* and *standard deviation*.

Packet-level 1-packet size count (366), 2-unique packet size (366), 3-number of packets per TCP conn. (9), 4-number of incoming packets per TCP conn. (9), 5-number of outgoing packets per TCP conn. (9), 6-ratio of incoming packet number per TCP[26](10), 7-packet frequency[27] (1), 8-incoming packets frequency (1), 9-outgoing packets frequency (1), 10-cumulative size of first 100 packets[28] (100), 11-cumulative size with direction of first 100 packets (100), 12-interpolant of cumulative packet size (100) [23], 13-concentration of outgoing packets in first 2,000 packets[29] (109), 14-alternative concentration of outgoing packets[30] (28), 15-concentration of first 30 incoming packets (1), 16-concentration of first 30 outgoing packets (1), 17-concentration of last 30 incoming packets (1), 18-concentration of last 30 outgoing packets (1), 19-number of packets per second in 60s[31] (68), 20-number of incoming packets per second (68), 21-number of outgoing packets per second (68), 22-alternative packets per second [32] (29), 23-alternative incoming packets per second (29), 24-alternative outgoing packets per second (29), 25-initial 30 packets[33] (30), 26-initial 30 incoming packets (30), 27-initial 30 outgoing packets (30), 28-initial 30 packets in first TCP conn.[34] (30), 29-initial 30 incoming packets in first TCP conn. (30), 30-initial 30 outgoing packets in first TCP conn. (30), 31-position of first 300 outgoing packets[35] (309), 32-preposition of first 300 outgoing packets[36] (309), 33-position of first 300 incoming packets (309), 34-preposition of first 300 incoming packets (309), 35-average inter-arrival time of first 20 packets[37](20), 36-average inter-arrival time of first 20 incoming packets (20), 37-average inter-arrival time of first 20 outgoing packets (20), 38-average inter-arrival time between packets (1), 39-average inter-arrival time between incoming packets (1), 40-average inter-arrival time between outgoing packets (1), 41-trace duration (1), 42-duration of incoming packets (1), 43-duration of outgoing packets (1),

TCP-level 44-average inter-arrival time between consecutive packets per TCP conn.[38] (209), 45-average inter-arrival time between incoming packets per TCP conn. (209), 46-average inter-arrival time between outgoing packets per TCP conn. (209) 47-relative start time of each TCP conn.[39] (208), 48-relative end time of each TCP conn. (208), 49-relative duration of each TCP conn. (208), 50-number of TCP conn. (1), 51-incoming bytes per TCP conn. (208), 52-outgoing bytes per TCP conn. (208), 53-total bytes per TCP conn. (208), 54-ratio of incoming bytes to total transmitted bytes per TCP conn. (208), 55-20 largest transmitted bytes per TCP conn.[40] (20), 56-20 largest incoming bytes per TCP conn. (20), 57-20 largest outgoing bytes per TCP conn. (20),

Burst-level 58-burst size count[41] (4,555*), 59-unique burst size (4,555*), 60-number of incoming bursts per TCP conn. (209), 61-number of outgoing bursts per TCP conn. (209), 62-number of bursts per TCP conn. (209), 63-ratio of incoming burst number per TCP conn. (210), 64-burst duration (8), 65-incoming burst duration (9), 66-outgoing burst duration (9), 67-size of incoming bursts (9), 68-size of outgoing bursts (9), 69-ratio of incoming

26 Include overall ratio of incoming packets in a trace.

27 Ratio between *total number of packets* and *trace duration*.

28 The *nth* cumulative packet size is calculated by summing up total size of first n packets. For example, given a packet sequence of [-100, 100, -70], first 3 cumulative packet size is [100, 200, 270] and first 3 cumulative packet size with direction is [-100, 0, -70].

29 Ratio of outgoing packets in every 20 non-overlapping packets [22, 33]. We focus on first 2,000 packets.

30 This feature is calculated by splitting concentration of outgoing packets feature list (12) into 20 evenly sized subsets and sum each subset [33].

31 Number of packets in each second along with their statistical derivatives [33]. Number of features in this category determines by the maximum time it takes to load a page in training data and we focus on first 60s.

32 Create 20 even sized subsets for the number of packets per second feature list (19/20/21) and calculate sum of values in each subset [33].

33 Direction and size of first 30 packets in a traffic trace [22].

34 Direction and size of first 30 packets in first TCP connection.

35 A list of features that indicate the total number of packets seen before each outgoing packet in the sequence [33].

36 Number of incoming packets between current outgoing packet and the previous outgoing packets.

37 It is calculated by adding up inter-arrival time between *nth* and *(n+1)th* packet in each TCP connection and dividing it by the total number of TCP connections in the traffic trace. We focus on first 20 time intervals.

38 For each TCP connection, we calculate the average inter-arrival time between consecutive packets. This feature category includes calculate the overall statistical derivatives and values taken from first 200 TCP connection.

39 Start time and end time for each of the first 200 TCP connections, relative to the start time of the first TCP conn.

40 It is calculated by sorting total transmitted bytes in each TCP in a descending order and select the top 20.

41 Burst size is defined as the total size of consecutive packets sent in one direction. Burst size count indicates the number of bursts with size X in a traffic trace. The range of X is determined by the maximum size of consecutive packets sent in one direction. In our dataset, number of features in this category is 4,555 when rounded to an incremental of 600.

bursts size per TCP conn.(8), 70-number of packets in a burst count[42] (109*), 71-number of packets in incoming burst count (109*), 72-number of packets in outgoing burst count (16*), 73-number of packets in each burst (8), 74-number of packets in each incoming burst (8), 75-number of packets in each outgoing burst (8), 76-initial 30 bursts in first TCP conn.[43] (30), 77-initial 30 incoming bursts in first TCP conn. (30), 78-initial 30 outgoing bursts in first TCP conn. (30), 79-initial 30 bursts (30), 80-initial 30 incoming bursts (30), 81-initial 30 outgoing bursts (30), 82-size and direction of the first incoming burst (HTML size [18]) (1),

Port-level 83-number of unique server port[44] (1), 84-server port count[45] (21*), 85-unique server port[46] (21*), 86-transmitted bytes per TCP conn. w.r.t. port 80/443[47] (18), 87-incoming bytes per TCP conn. *w.r.t.* port 80/443 (18), 88-outgoing bytes per TCP conn. *w.r.t.* port 80/443 (18), 89-ratio of incoming bytes *w.r.t.* port 80/443 (2),

IP address-level 90-number of unique sever IP addresses[48] (1), 91-server IP address count[49] (8,727*), 92-20 largest server IP address count (20), 93-unique server IP address (8,727*), 94-transmitted bytes per TCP conn. *w.r.t.* server IP address[50] (180), 95-incoming bytes per TCP conn. *w.r.t.* server IP address(180), 96-outgoing bytes per TCP conn. *w.r.t.* server IP address (180), 97-ratio of incoming bytes per TCP conn. *w.r.t.* server IP address (160), 98-20 largest transmitted bytes per TCP conn. *w.r.t.* server IP address[51] (20), 99-20 largest incoming bytes per TCP conn. *w.r.t.* server IP address (20), 100-20 largest outgoing bytes per TCP conn. *w.r.t.* server IP address (20), 101-number of unique hostnames[52] (1), 102-hostname count[53] (1,141*), 103-transmitted bytes per TCP conn. *w.r.t.* hostname[54] (160), 104-incoming bytes per TCP conn. *w.r.t.* hostname (180), 105-outgoing bytes per TCP conn. *w.r.t.* hostname (180), 106-ratio of incoming bytes per TCP conn. *w.r.t.* hostname (160), 107-20 largest transmitted bytes per TCP conn. *w.r.t.* hostnames[55] (20), 108-20 largest incoming bytes per TCP conn. *w.r.t.* hostname (20), 109-20 largest outgoing bytes per TCP conn. *w.r.t.* hostname (20)

20 most common server IP address in our dataset 1-216.58.217, 2-204.85.30, 3-204.85.32, 4-31.13.69, 5-31.13.71, 6-216.58.218, 7-172.217.1, 8-54.192.19, 9-151.101.32, 10-52.85.142, 11-68.67.178, 12-8.43.72, 13-216.58.195, 14-172.217.2, 15-74.119.118, 16-199.16.156, 17-173.241.242, 18-72.21.91, 19-66.150.48, 20-199.96.57,

20 most common hostnames in our dataset 1-1e100.net. , 2-amazonaws.com. , 3-akamaitechnologies.com. , 4-cloudfront.net. , 5-fbcdn.net. , 6-facebook.com. , 7-sl-reverse.com. , 8-adnexus.net. , 9-yahoo.com. , 10-quantserve.com , 11-openx.org , 12-googleusercontent.com. , 13-aol.com. , 14-nr-data.net. , 15-turn.com. , 16-yandex.ru. , 17-hwcdn.net. , 18-btrll.com. , 19-amsedge.net. , 20-omtrdc.net. ,

12 Additional Evaluations

12.1 Performance with other datasets

We next use two datasets made available by prior work to study our feature selection methodology.

SSH2000 Dataset [11] We evaluate the performance of *Wfin* with the data on 2,000 websites in *SSH2000* [11]—for scenarios *S2*, *S4*, and *S5*. The results are summarized in Table 16. As can be seen, the difference in performance of *Wfin* and the best-performing features from the state-of-the-art is quite significant—5.65%, 17.16%, and 20.64%, respectively, across the three scenarios. We also find that the classifiers that rely on packet size do not perform as well with this dataset (compared to our dataset). We suspect this is due to less uniqueness in packet sizes in *SSH2000* (according to analysis in [11]). On the other hand, accuracy obtained with *DTW* is higher in *SSH2000* than with our dataset, which suggests that packet times in *SSH2000* are more unique for each website. This is also supported by the

42 It indicates the number of bursts contain n packets in a traffic trace. Up-bound of n determines by the maximum number of consecutive packets sent in one direction in training set, which is 109 in our dataset.

43 It indicates size and direction of first 30 bursts.

44 Number of different server ports seen in a traffic trace.

45 It indicates number of TCP connections that are sent over port X. Values of X is determined by how many different server ports have been seen in training data.

46 It indicates whether a specific server port (such as 443,80) has been used for transmitting data in a traffic trace. If yes, set it to 1 else 0.

47 It calculates statistical derivatives about packet sent over port 80/443 in each TCP connection.

48 It illustrates the number of different server addresses a client connects with to load a website.

49 It counts the occurrence of each server address in a traffic trace. Number of this feature depends on how many different server addresses are seen in the training dataset, which in our case is around 8,727.

50 We focus on transmitted bytes per TCP connection *w.r.t.* 20 most common IP addresses in our dataset.

51 We compute the total transmitted bytes with each server address in a traffic trace and record the 20 largest value in a descending order.

52 It computes the number of different second-level hostnames a client connects with to load a website.

53 It indicates how many TCP connections each hostname connects with to load a website.

54 We focus on transmitted bytes per TCP connection *w.r.t.* 20 most common hostnames in our dataset.

55 We compute the total transmitted bytes per hostname in a traffic trace and record the 20 largest value in a descending order.

Table 16. Performance comparison between different sets of features with 2,000 websites in *SSH2000*.

	H	*L*	*P*	*Vng++*	*DTW*	*CUMUL*	*FLSVM*	*k-FP*	*Wfin*
S2: Encrypted Tunnel	*76.02	*74.75	*74.05	*47.88	34.03	69.28	70.64	*63.13	81.67
S4: S2+PadToMTU	7.06	15.81	29.24	30.99		*58.99	27.94		80.29
S5: S4+Fixed IAT			29.17	26.62	45.69			54.91	75.55

Table 17. Top 30 informative features in *S2: Encrypted Tunnel* with 2,000 websites from *SSH2000*. '**' indicates features that have not been discovered before.

1	unique packet size	28.27
2	preposition of first 300 incoming packets	8.72
3	packet size count	6.466
4	position of first 300 incoming packets	6.173
5	unique burst size	4.678
6	** initial 30 outgoing bursts	4.045
7	concentration of outgoing packets in first 2,000 packets	2.656
8	initial 30 packets	2.519
9	size of outgoing bursts	2.401
10	position of first 300 outgoing packets	2.279
11	alternative concentration of outgoing packets	2.259
12	initial 30 outgoing packets	2.204
13	ratio of incoming bursts size per TCP conn.	2.001
14	burst size count	1.902
15	** cumulative size with direction of first 100 packets	1.897
16	** initial 30 bursts	1.741
17	** average inter-arrival time of first 20 packets	1.741
18	preposition of first 300 outgoing packets	1.695
19	** average inter-arrival time of first 20 outgoing packets	1.586
20	** burst duration	1.529
21	** average inter-arrival time of first 20 incoming packets	1.165
22	** outgoing burst duration	1.075
23	initial 30 incoming packets	1.042
24	concentration of first 30 incoming packets	0.902
25	size of incoming bursts	0.799
26	concentration of first 30 outgoing packets	0.72
27	** incoming burst duration	0.694
28	** cumulative size of first 100 packets	0.672
29	** initial 30 incoming bursts	0.671
30	ratio of incoming packets # per TCP conn.	0.659

gap between the performance of *Vng++* (which uses total transmission time as one of three features) in each dataset. When inter-arrival times are fixed (*S5*), classification accuracy obtained with features proposed in *k-FP* decreases from 63.13% to 54.91% with *SSH2000*, while in our dataset the performance gap is less than 1%. This further supports the conclusion about unique timing information in *SSH2000*, since *k-FP* uses three features extracted from packet timestamp.

Tor dataset [22] We evaluated the performance of our classifier with a public Tor dataset provided by Wang et al.[22], which is collected by visiting 100 websites each 90 times with Tor browser. With the informative features identified in *S4*, we are able to achieve an accuracy of around 92.21% with 90 instances per website (60 for training and 30 for testing) using Extra-Trees—this is comparable to the accuracy 0.91 ± 0.03 reported in [22]. Based on estimation of bayes error about samples in Tor dataset in [35], an accuracy around 91% may be the best performance we are able to obtain with this Tor dataset in a closed-world experiment.

Table 18. Performance evaluation by using classifier introduced in previous work with 2,000 websites in our dataset.

	H	*L*	*P*	*Vng++*	*CUMUL*	*FLSVM*	*k-FP*	*Wfin*
S0			93.69	39.88				97.96
S1	*92.51	92.48			82.85	76.20		97.73
S2			*93.08	*7.69			*88.18	97.41
S3			63.10	33.04				97.54
S4	0.13	31.31	*19.87	31.69	*84.18	63.15		96.83
S5			19.61	26.45			86.10	95.44
S6	94.45	84.65	83.62	13.44	33.89	83.20	73.67	96.70
S7	88.88	85.30	93.43	24.33	36.59	76.20	64.23	96.76

12.2 Influence of Classifier

In Section 4-7, we have evaluated the performance of different feature sets on classification accuracy by using the Extra-Trees classifier. In order to understand the potential impact of different machine learning algorithms on classification performance, we next evaluate classification accuracy achieved using the respective machine learning algorithm proposed in the original work. Table 18 summarizes the results. Comparing with Table 6, two observations are worth emphasizing here:

1. Classifiers do affect classification performance. For example, with the same input data samples and features, accuracy achieved with Extra-Trees is higher in *most* cases, compared to Bayes classifiers such as Naive Bayes (L) and Multinomial Bayes (H).
2. Extra-Trees does not outperform in all cases. For example, SVM, which is the original machine learning algorithm used in P, outperforms Extra-Trees in *S4* and *S5* (in which P performs poorly); while in other scenarios, such as *S0* and *S1* (in which P performs well), Extra-Trees performs better than SVM.

More fundamentally, we believe that our analysis suggests that any website fingerprinting research must separately evaluate the impact of the feature set and the machine learning algorithm being used, in order to help us better understand the improvement in the proposed work compared to others. This is especially important for prior work that combines results from either multiple layers of classifiers, or employs additional models (e.g., HMM) for improving classification performance [8, 33].