A Stable Fine-Grained Webpage Fingerprinting: Aiming at the Unstable Realistic Network

Songtao Liu[1], Hua Wu[1,2(✉)], Hao Luo[1], Guang Cheng[1,3], and Xiaoyan Hu[1]

[1] School of Cyber Science and Engineering, Southeast University, Nanjing, China
{stliu,hwu,hluo}@seu.edu.cn, {gcheng,xyhu}@njnet.edu.cn
[2] Purple Mountain Laboratories for Network and Communication Security, Nanjing, China
[3] Jiangsu Province Engineering Research Center of Security for Ubiquitous Network, Nanjing, China

Abstract. Website fingerprinting enables attackers to snoop on users' browsing preferences for websites, even if the network connections are encrypted. Fine-grained webpage fingerprinting can reveal more sensitive privacy by precisely distinguishing pages from the same website. Recognizing similar webpages within the same website needs representative fine-grained features. However, the fluctuating realistic network makes it challenging to ensure the stability of fine-grained features. In this paper, we propose a fine-grained webpage fingerprinting method named Stable Webpage Fingerprinting (StableWPF) to obtain stable webpage features that are not affected by the unstable realistic network. We use the frequency information of the length of the TLS fragments to depict the most representative features. To eliminate the fluctuation of the features, we leverage the kernel density estimation and the Bag-of-Words model in our method. The experimental results in the closed-world and open-world scenarios show that our method outperforms the state-of-the-art approaches in accuracy, precision, and recall. The robust performance obtained on famous real-world websites with various network environments demonstrates the generalization ability of our method.

Keywords: Fine-grained webpage fingerprinting · Encrypted traffic classification · Machine learning · Realistic network

1 Introduction

Deploying end-to-end encrypted protocols in network communications has become a common practice. In September 2022, around 95% of the traffic across Google Chrome was encrypted [1]. Despite the websites' relatively sophisticated secure communication methods, potential attackers can still gain access to user privacy.

Website fingerprinting (WF) is a privacy threat that allows passive attackers to infer which website users are browsing by analyzing network traffic patterns.

H. Duan et al. (Eds.): SecureComm 2023, LNICST 568, pp. 167–188, 2025.
https://doi.org/10.1007/978-3-031-64954-7_9

Recent studies have proposed several WF methods. The works in [2–5] extracted statistical features to train traditional machine learning models. The other works [6–12] leverage deep learning for automatic feature engineering.

Webpage fingerprinting (WPF) can be a more significant threat than WF because it precisely recognizes which pages users are browsing. Webpage browsing behavior reflects more detailed privacy information. Users' consumption habits, daily hobbies or political tendencies could be leaked once an eavesdropper knows what specific webpages users are browsing.

We define the traditional WFs as coarse-grained fingerprinting methods because they focus on recognizing websites, not specific webpages. Panchenko *et al.* [3] first comprehensively evaluated the possibility of extending the fingerprinting scale to webpages. Though Panchenko argued that WPF is unfeasible [3], his work attracts researchers to keep studying WPF, which is a fine-grained fingerprinting method compared to the WF.

It is challenging to achieve fine-grained WPF because the traffic patterns for pages on the same website are relatively similar. Besides, the broad deployment of RESTful architecture and content delivery networks (CDN) among websites, especially large-scale famous websites, make the traffic pattern of webpages much more complicated. Nevertheless, some inspiring and creative WPF methods have been proposed recently. Martino *et al.* [13] achieved fine-grained WPF on 100 webpages. Shen *et al.* [14–16] proposed several fine-grained WPF models and constructed fine-grained WPF on the largest number of webpages, which is 330 webpages within a website.

However, realistic network environments are unstable, and traffic patterns vary greatly with network fluctuations. The existing WPF methods are based on packet size and timestamps extracted from raw packets. This causes webpage features to fluctuate in the unstable network environment, which may lead to insufficient generalization of the models.

To construct fine-grained webpage fingerprints in the unstable realistic network, we propose a WPF method named stable webpage fingerprinting (StableWPF). It can construct stable webpage features in unstable network environments and shows great generalization ability in real-world datasets. The main contributions of this paper are as follows.

- To obtain stable webpage features, we propose the size of the TLS fragments (SoTs) to represent webpages' resources. Further, we extract frequency characteristics from it as the fingerprint of a webpage. Besides, we design two strategies to eliminate the fluctuation caused by unstable network conditions. To the best of our knowledge, our method is the first work to extract the frequency features of TLS fragments' size as webpage fingerprints.
- Famous real-world websites with various network conditions are selected to construct fine-grained webpage datasets. The datasets we collected have the largest number of webpages within a website.
- We evaluate StableWPF on the datasets and compare our method with similar works. StableWPF achieves high accuracy and precision with fewer training instances, which outperforms existing state-of-the-art webpage fin-

gerprinting methods. Besides, the evaluation on four large-scale real-world datasets with different network environments proves the generalization ability of our method.

2 Related Work

2.1 Coarse-Grained Website Fingerprinting

We define the traditional Website Fingerprinting (WF) methods as coarse-grained fingerprinting methods because they can only recognize websites but fail to distinguish pages within the same website.

The traditional WFs leverage domain knowledge to extract features and apply machine learning methods as classifiers. Panchenko *et al.* [3] leveraged cumulative packet length as webpage features and selected K-Nearest Neighbor (KNN) as the classifier. Wang *et al.* [17] utilized packet lengths to achieve WF and selected KNN as the classifier. Hayes *et al.* [2] used packet number, transmission time, and inter-arrival time to construct WF. These machine learning methods leverage lightweight models that could achieve relatively low time costs. However, the performance of the classifiers is highly dependent on the efficiency of features.

With the fast advance of deep learning, automated feature extraction is applied in WF. DF [10] is a famous WF method that uses only the direction information of the packets. Bhat *et al.* [11] followed the feature extracted in DF and introduced timestamps as a supplement. They used the features to train the novel Convolutional Neural Network Resnet. TLFA [6] was introduced in Chen *et al.*'s paper using transfer learning to construct a WF that can scale well to extensive training data. These deep learning methods mainly focused on the design of the neural network but paid little attention to feature engineering. Besides, the nature of neural networks dictates that they require large amounts of data as training samples, which might lead to high time and space costs.

In the literature, traditional WF can only recognize different websites' homepages but did not achieve fingerprinting on similar webpages within a website.

2.2 Fine-Grained Webpage Fingerprinting

As the research of WF continues, webpage fingerprinting (WPF) has gained more attention. Though Panchenko *et al.* [3] considered WPF unfeasible a few years ago, novel WPF methods have been proposed recently.

Mitra *et al.* [18] comprehensively evaluated the factors that affect the performance of WPF and gave an insightful solution. However, his method is restricted in the experimental scenario that requires exerting influence on the websites' transmission quality, which is hard to implement in the real world. The traffic patterns of the realistic network can hardly be simulated in the experimental scenario because of the broad deployment of RESTful architecture and CDN servers among websites. Therefore, to ensure the WPF method's generalization ability,

datasets used for training and evaluation should be collected from real-world websites.

There are several studies constructing fine-grained WPF in realistic network environments. Martino *et al.* [13] focused on fingerprinting social media webpages and evaluated several WF methods in the fine-grained webpages dataset with 100 Twitter pages. In 2019, Shen *et al.* [14] first achieved fine-grained WPF and built a fine-grained webpage dataset with a maximum of 100 webpages of a famous online shopping website named *Jingdong* [19]. Their following work enlarged the dataset [15,16]. Shen's dataset in [15] is the largest WPF dataset with the greatest number of webpages. However, Martino *et al.* and Shen *et al.* used the packet sequence information to construct webpage features. These features are unstable when the network condition fluctuates. As a result, classifiers trained by unstable features might achieve high performance on a specific dataset but show little generalization ability on other real-world datasets.

In this paper, we are devoted to constructing stable webpage features to achieve fine-grained webpage fingerprinting in the realistic network.

3 Motivation

This section first discusses the threat model for fine-grained webpages fingerprinting in a more realistic scenario. Then, we evaluate how different complex network conditions affect features obtained by existing methods.

3.1 Threat Model

In a typical threat model of WPF, attackers passively observe the traffic patterns between clients and the website's servers. Attackers can identify the webpages the users are accessing, even if the traffic is encrypted.

Figure 1 shows the threat model in this paper. We assume a greedy attacker who tries to monitor multiple victims' preferences for webpages. There are two main challenges for the greedy attacker. First, the attacker needs to construct a fine-grained WPF model that can infer which webpages in a website are visited. Second, due to the spatial-temporal differences between each victim, their network conditions are diverse. The attacker needs to ensure the stability of the webpage features in various situations, such as network congestion, router redirection or transmission error.

3.2 Complex Network Conditions Affect Traffic Pattern

The network conditions in the real world are complex and always change. It is because the real-world transmission link between clients and servers is commonly long, and numerous factors will affect the traffic pattern. For example, the network congestion causes packet delay or packet loss; the transmission error leads to packet corruption; the router redirection brings packet duplicate or packet reordering.

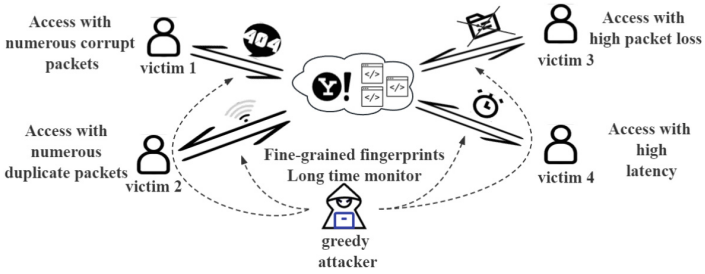


Fig. 1. A greedy attacker could snoop on victims' privacy through observing the traffic patterns

Existing methods directly extract statistical features from the traffic pattern of webpages' fetching process, such as packet number, packet size or arrival time. However, the fluctuation of traffic patterns will lead to the fluctuation of statistical features. We evaluated the features of existing methods in five different network conditions. The network conditions are measured by the following metrics: delay, loss, corruption, duplication and reorder. We deployed a Linux tool named TC on a soft router to simulate these situations. The settings of the experiments are in Table 1. We accessed the same webpage on different network conditions and visualized each feature in Fig. 2 and Fig. 3.

Table 1. Parameters of the experiment.

Network Condition	Parameters Setting
Packet Delay	All the packets will delay 100 ms
Packet Loss	loss rate of the network will be set to 16%
Packet Corrupt	5% of the packets will be corrupted
Packet Duplication	15% of the packets will be duplicated
Packet Reorder	25% of the packets will be reordered

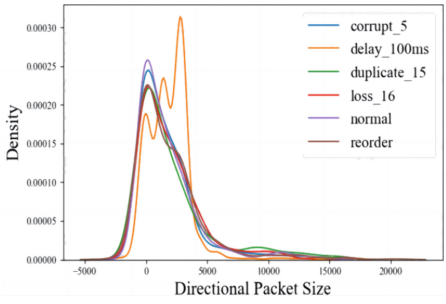


Fig. 2. Distribution of directional packet size in different network conditions

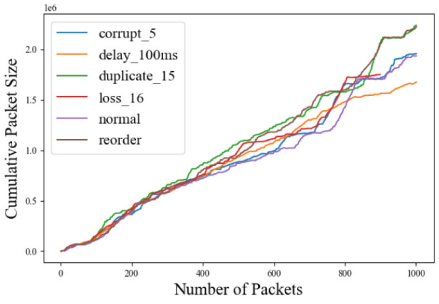


Fig. 3. Cumulative packet size in different network conditions

Directional packet size is widely used in existing WF and WPF methods [7,8,10,11,16,20]. This feature includes packet size and its direction. The sign of packet size indicates the direction of the packet: positive means outgoing, and negative means incoming [10]. As shown in Fig. 2, the distributions of directional packet sizes show various patterns in the face of complex network conditions, especially in the delay situation.

Cumulative packet size is another commonly used feature [3,15,21]. This feature is based on the sum of the first n packet sizes. As shown in Fig. 3, the patterns of the cumulative packet size are similar in the early stage, but the difference among them gets more obvious when the number of packets exceeds 400.

Leveraging machine learning and deep learning classifiers to construct WPF requires establishing a stable mapping relationship between the traffic pattern and webpage labels. Based on the above analysis in the unstable network, existing methods can not obtain stable features in complex network conditions. Models that leverage the fluctuating features can perform well on a specific dataset but show insufficient generalization ability on other datasets.

We propose StableWPF to construct fine-grained webpage fingerprints that can remain stable in unstable network environments. The implementation of StableWPF will be introduced in Sect. 4.

4 Methodology

4.1 Overall Architecture

The overall architecture of our method is depicted in Fig. 4. There are three key steps in our method:

Extract Stable Features from the Original Traffic: We group flows by the servers and reconstruct the sequences of TLS fragments in the flows. It calculates SoT and sends the SoT sequences to the features processing module.

Construct Fluctuation-Resistant Features: To handle the size fluctuation of the SoTs, we find similar sizes that might represent the same SoT by leveraging a clustering method. Then, we extract SoTs that can describe the characteristics of webpages based on frequency information. After that, a bag of words model is applied to solve the order fluctuation of the SoTs. Finally, the processing results of the bag of words model are sent to the classifiers for training.

Pages Recognition: We construct classifiers that use features extracted from the former step to predict which webpage is being visited. It learns the frequency characteristics of TLS fragments and achieves fine-grained webpage fingerprinting.

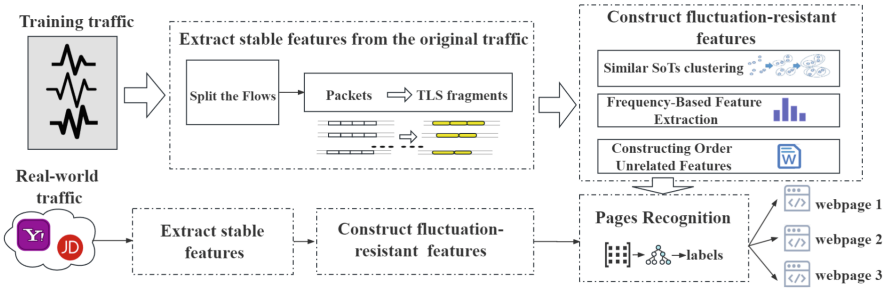


Fig. 4. Overall architecture.

4.2 Extract Stable Features from the Original Traffic

It is the resources of a webpage that make the page unique. A webpage consists of an HTML page and various other resources. An application layer resource can be represented by an application data unit (ADU). Figure 5 illustrates the process of encapsulating an application data unit into a series of TCP packets. The header information of HTTP, TLS, and TCP is added to ADUs, and rearrangement occurs due to the different split strategies of each layer. Compared to packets in the transport layer, fragments in the TLS layer are closer to the ADU. As a result, the patterns of the fragment in the TLS layer are more in line with the characteristics of ADU.

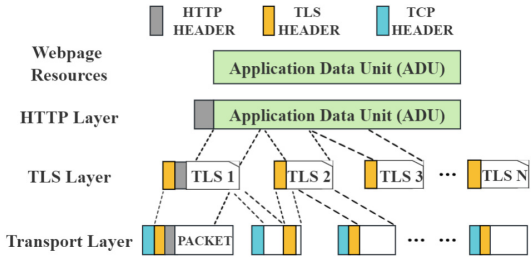


Fig. 5. Process of encapsulating an application data unit to a series of encrypted TCP packets.

Webpage resources could be represented by their sizes because different resources usually have different sizes. Therefore, in this paper, we propose reconstructing TLS fragments and calculating the size of TLS fragments (SoTs) to construct the webpage feature. Figure 6 gives a simple example of how a SoTs sequence represents a webpage resource. An image resource is segmented and transmitted by 4 TLS fragments. The SoT are 500 bytes, 500 bytes, 500 bytes

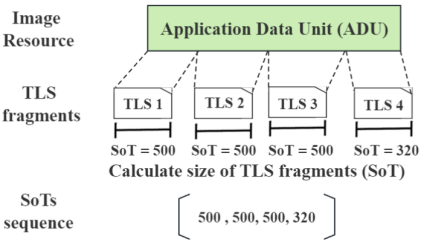


Fig. 6. Process of constructing SoTs sequence.

and 320 bytes. The SoTs sequence that could represent this image is [500, 500, 500, 320].

We group the flow by servers and extract SoTs sequences as basic features. First, we group the flows based on 5-tuples: source IP and port, destination IP and port, and protocol. Second, by analyzing the server name indication (SNI) in TLS Client Hello message in each flow, the flows will be grouped by SNI. To better portray the ADU, we reconstruct packets into TLS fragments using the unencrypted header information of the TLS suite defined in RFC 5246 [22]. Finally, an SoT sequence is the preliminary feature of a webpage and is sent to the next module for further processing.

SoT sequences obtained by StableWPF are distinguishing enough to recognize webpages because they precisely depict the divergences between different webpages, even from the same website. An example in Fig. 7 exhibits the patterns of SoTs from different webpages within a website.

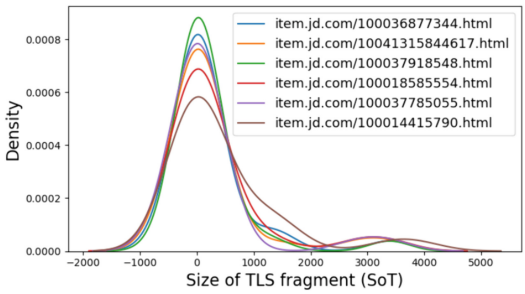


Fig. 7. SoT distributions for different webpages.

4.3 Construct Fluctuation-Resistant Features

As discussed in Sect. 3.2, complex network conditions affect the traffic pattern; SoTs of the same webpage resources are subject to size fluctuations and arrival order fluctuations in the realistic network. To construct fluctuation-resistant

features, we first leverage the clustering algorithm to gather similar SoTs that might represent the same resources. Then, we merge SoTs into SoCs (size of the chunks) and construct features with frequently occurring SoCs. Third, we leverage the bag of words (BoW) model to eliminate the order fluctuation of the features.

Similar SoTs Clustering. The encapsulation of an ADU leads to the size fluctuation problem of SoTs because of the additional information, rearrangement, and compression algorithm of different protocols. Similar SoTs may be from the same ADU. The possible size of TLS fragments (SoTs) could be different in multiple capture rounds. Therefore, we need to find similar SoTs and smooth out these size fluctuations before extracting effective features.

StableWPF adopts a 1-D clustering strategy based on kernel density estimation (KDE) to solve this problem. KDE is the application of kernel smoothing for probability density estimation. The probability density reveals the place where the data is grouped. It can be used for 1-D clustering by leveraging the highest density as the cluster center and the lowest density as the separating position. We accessed a real-world website *Jingdong*, then captured the traffic of the same webpage five times. Figure 8 and Fig. 9 show that the clustering method narrows the gap between different distributions of SoTs for the same page.

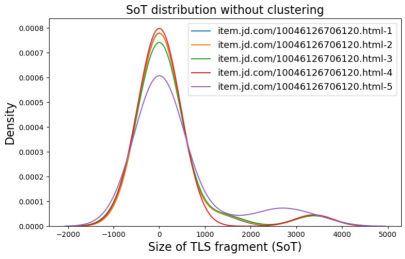


Fig. 8. SoT distributions for the same webpage without KDE

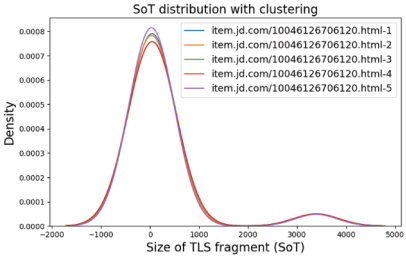


Fig. 9. SoT distributions for the same webpage with KDE

The process of 1-D clustering in this paper can be summarized as follows:

Step 1: Let F be a set that contains SoTs from the same SNI. Sort the values in F in ascending order, and put them into a sequence P.
Step 2: We leverage KDE to compute density among those values. Values in P denote x_i where $1 \leq i \leq N$, N is the total number of x_i. Since P is a sorted sequence, $\forall i \leq j, x_i \leq x_j$. y is a point in the interval between x_1 to x_N. The density is given by Eq. (1):

$$p_K(y) = \sum_{i=1}^{N} K\left(y - x_i; h\right) \tag{1}$$

$K(x;h)$ in Eq. (1) is a kernel function, which is positive mathematically, and the bandwidth parameter h controls it. The kernel function selected in this paper is the Gaussian kernel and is defined as Eq. (2):

$$K(x;h) = e^{-\frac{x}{2h^2}} \tag{2}$$

The local minima of $p_K(y)$ are found and denote m_u, where $x_1 < m_u < x_N, 1 \leq u \leq n$, n is the total number of local minima. We further define $m_0 = x_1$ and $m_{n+1} = x_N$.

Step 3: The interval $[x_1, x_N]$ is divided into $n+1$ intervals by m_u. We denote the v^{th} interval as I_v and $I_v = [x_b, x_e]$ where $1 \leq v \leq n+1$ and $m_{v-1} \leq x_b \leq x_e \leq m_v$.

Step 4: We construct a function Z as Eq. (3):

$$Z(x_i) = \sum_{k=b}^{e} \frac{x_k}{e-b+1}, x_i \in I_v = [x_b, x_e] \tag{3}$$

Equation (3) calculates the mean value of the elements that belong to the same interval as x_i. Our method modifies the value f_i in set F to $Z(f_i)$, which means those similar SoTs are modified to their mean value.

Figure 10 is an example of similar SoTs clustering. First, SoTs of the same SNI are sorted in ascending order and placed in a sequence. Second, we calculate the kernel density of the SoTs and find the local minima of the function, which are 261, 572 and 1014, respectively. Third, we leverage the local minima as the split points and split the sequence into several intervals. For example, 430, 431, and 432 are greater than 261 and less than 572, so we put them in the same group. Those SoTs in the same group are considered similar SoTs. Fourth, we modify the SoTs by Eq. (3), which calculates the mean value of a group. Since similar SoTs have been modified into the same value, the negative effect of size fluctuation is eliminated.

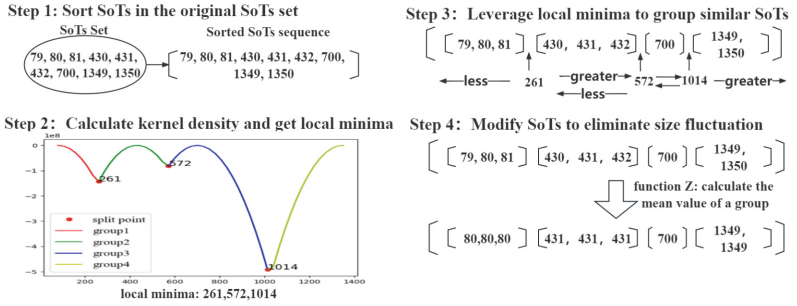


Fig. 10. Similar SoTs clustering.

Frequency-Based Feature Extraction. We have proposed that the resources of a webpage make the page unique. Some of the resources are static, such as images of a commodity. But those embedded dynamic resources, such as rankings, trending topics, or advertisements, might change over time. If our feature engineering takes the dynamic resources into consideration, their fluctuation trait yields some random number in feature vectors, making the classifiers less effective. Hence, only static resources should be considered as the representation of the webpage.

Since the relatively stable SoTs have been obtained by similar SoTs clustering, the frequency of SoTs could represent the occurrence of the resources. Therefore, StableWPF leverages a frequency-based method to find and represent those static resources that could precisely fingerprint webpages.

First, we merge the SoTs into SoCs (size of chunks) to better portray ADUs on a webpage. When an ADU is encapsulated into HTTP data, the website might apply a fragmenting mechanism to split the HTTP data into several consecutively isometric TLS fragments with a smaller one. We define the combination of these TLS fragments as a TLS chunk, and the size of TLS chunks (SoCs) might represent the size of HTTP data. We can recognize this pattern by finding the SoTs that frequently appear in a TCP connection because every ADU in a webpage obeys the same fragmenting rule. As an example in Fig. 11, three consecutive TLS fragments of length k and a smaller one of length m are merged into a TLS chunk.

Second, we find the frequent SoCs in the sequences. The SoCs frequently appear in the same browsing behavior, which means visiting the same webpage many times, can be seen as the fingerprints of a webpage. We calculate the frequency of SoCs in the sequences of multiple visits to a webpage. Then we define the SoCs whose frequency surpass a threshold θ_1 as the webpage's frequent SoCs.

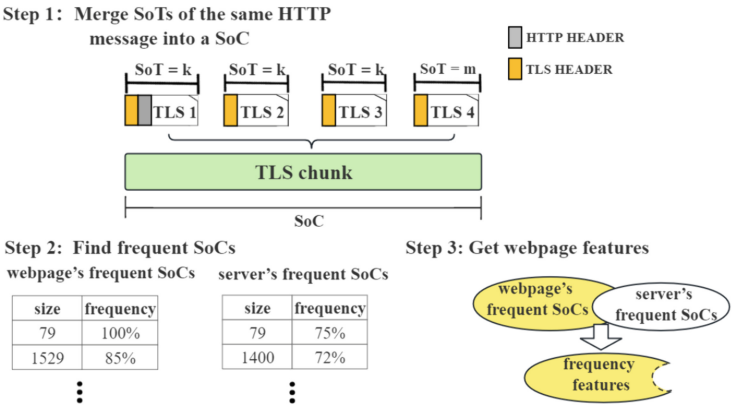


Fig. 11. Frequency-based features extraction.

Although the webpage's frequent SoCs do reflect the pattern of a webpage, they may also include SoCs related to servers. For example, every webpage within a website needs to fetch the logo of the website through an image server. Consequently, SoCs associated with this resource will be regarded as the webpage's frequent SoCs, but they are apparently interfering with webpage fingerprinting. This paper aims to construct a high-precision webpage fingerprinting method, so this kind of SoCs should be removed. Therefore, we further calculate the frequency of SoCs in the sequences of multiple visits to servers. Then we define the SoCs whose frequency surpass a threshold θ_2 as the server's frequent SoCs.

Eventually, we extract the SoCs that are the webpage's frequent SoCs but not the server's frequent SoCs. As illustrated in Fig. 11, the frequency features are the difference between the set webpage's frequent SoCs and the set server's frequent SoCs.

Constructing Order Unrelated Features. Due to the unstable realistic network conditions and the various distances between the end-users and servers, the arrival time of webpage resources is unpredictable. Most existing methods did not consider the transmission order fluctuation problem. It leads to their model could only recognize pages in a stable network condition and could not be applied to monitoring users in different network conditions.

In this paper, we use a strategy that makes feature vectors unrelated to the transmission order. The BoW model is introduced as the solution. A typical bag of words model is a representation of text that describes the occurrence of words within a document. In this paper, we use the frequency features obtained in the previous step and construct a list that contains all the non-repeating SoCs in the frequency features of webpages.

As depicted in Fig. 12, the original feature vector of webpage 1 is [79, 1350, 430, 700, 700, 1458], and each number represents a SoC that could portray the frequency features of webpages. The big circle represents the full dataset. In this example, there are three webpages in total. We drop the repeated SoCs; the rest consist of a non-repeating SoCs list. The original feature vectors are transformed into BoW feature vectors, which consist of the count of each value in the list. 79, 1350, 430, and 1458 have appeared once in the feature vector of webpage 1, so the positions that represent these numbers are set to 1. For the same reason, the position representing 700 is set to 2, and the rest are set to 0. After that, the feature vector of webpage 1 is transformed into the BoW vector and can be fed into the deep learning or machine learning classifiers. Note that feature vectors in Fig. 12 are only for reference; the actual vectors are much longer.

Stability of the Features. Based on the above feature engineering, the features obtained by StableWPF can remain stable in unstable network conditions. We used the same experimental setup as in Sect. 3 and visualized the distribution of our features of the same webpage in different network conditions. Compared with the performance of directional packet size in Fig. 2 and cumulative packet

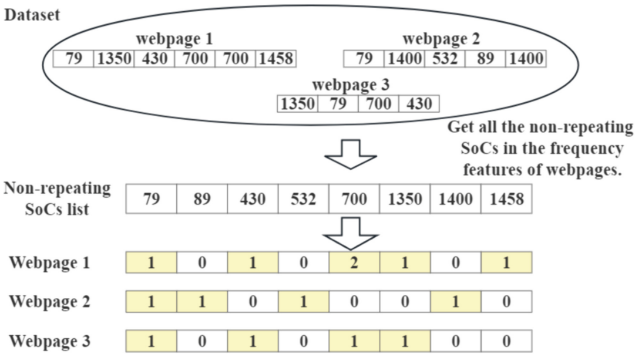


Fig. 12. Constructing order unrelated features.

size in Fig. 3, the result in Fig. 13 shows that our features achieve outstanding stability in all network conditions mentioned in Sect. 3.

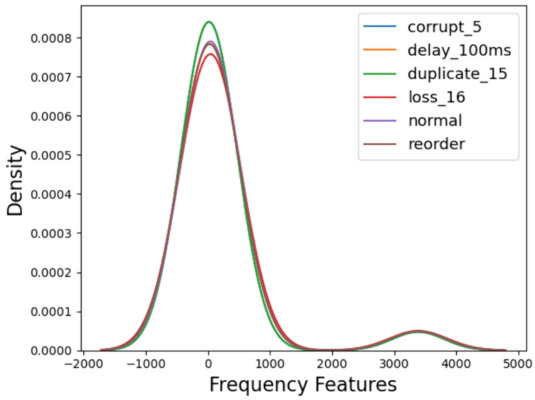


Fig. 13. Distribution of frequency features in different network conditions

4.4 Pages Recognition

We leverage the features extracted from the former step to train the classifier and then recognize webpages. To achieve better recognition performance, we construct the classifier based on both traditional machine learning methods and deep learning methods. We first design a deep learning model based on CNN. Then, we select K-Nearest Neighbor (KNN), Random Forest (RF) and Naïve Bayes (NB) as potential classifiers. The comparison between the classifiers will be presented in Sect. 5.

After the classifier is well-trained, StableWPF is able to recognize fine-grained webpages in the encrypted traffic scenario. It receives encrypted traffic from the real world and predicts which webpage is being visited.

5 Evaluation and Analysis

In this section, we first introduce the datasets used in our experiments. Then, we compare the performance of our method StableWPF to the state-of-the-art approaches in both the closed-world and the open-world scenarios.

5.1 Experimental Settings

Dataset Description. Existing open-access datasets are designed for website fingerprints and only provide processed statistical features of websites' homepages, not raw packets. To better verify if our method could achieve fine-grained webpage recognition, we captured the traffic from the real-world networks and constructed fine-grained datasets whose instances are represented as stand-alone webpages.

To the best of our knowledge, the datasets used in Shen *et al.*'s paper [15] are the largest existing fine-grained closed-world datasets, which contain the largest number of pages that belong to a website. In this paper, we chose the same websites as Shen *et al.* and enlarged the datasets with more pages and websites.

We collected four real-world datasets for the closed-world evaluation and one dataset for the open-world evaluation by the crawler. The datasets were divided into training and testing sets using the 7/3 split ratio in both scenarios. The summary of the datasets is shown in Table 2.

In the closed-world scenario, we aim to correctly distinguish every webpage, which is a multi-class classification. To verify our method's generalization ability, we chose four websites of different countries with various network conditions, which are *Wikipedia*, *Yahoo*, *MSN* and *Jingdong* [19]. These websites are ranked high in SimilarWeb [23], which means webpage fingerprinting will reveal a high volume of user privacy. Pages on these websites are randomly selected, and each page is given an individual label. The datasets of these four websites are named JD, YH, MSN, and WIKI, respectively.

In the open-world scenario, our target is to recognize monitored webpages from a larger amount of unknown unmonitored webpages. The open-world dataset we construct is named OW. Pages in the JD dataset are selected as monitored ones. To evaluate whether our model could recognize monitored pages with the traffic of similar unmonitored pages, we randomly chose 10000 pages from the same website, *Jingdong*, as unmonitored pages. These unmonitored pages have similar designs and resources to the monitored ones. To scale up the experiment, we further add the AlexaTop 10000 [24] websites' homepages (7916 pages are still available) as the unmonitored pages.

Table 2. Summary of the datasets.

Dataset	Webpages		Labels	Instances per pages	Total instances
JD	360		360	30	10800
YH	321		321	10	3210
MSN	150		150	30	4500
WIKI	150		150	30	4500
OW	Monitored	360	1	30	10800
	Unmonitored	17916	1	1	17916

1 Closed-world datasets: JD, YH, MSN, WIKI.
2 Open-world dataset: OW. 10000 pages from *Jingdong*, 7916 pages from AlexaTop 10000.

Baseline. We evaluate our method and then compare it with three state-of-the-art methods. Few existing works claim that they achieve webpage fingerprinting (WPF). Since the comparability of the evaluation results is affected by the scale of resources, the complexity of resources' distribution and the architecture of websites, we select the baselines that construct on famous real-world websites data. We choose FineWP [15], DF [10] and Var-CNN [11] as baselines.

- **FineWP** [15] is a fine-grained webpage fingerprinting method. It classifies the most significant number of webpages from a famous real-world online shopping website with the best performance among existing studies.
- **DF** [10] is a website fingerprinting method based on deep learning. It leverages the direction information of packets and attains over 98% accuracy on Tor traffic.
- **Var-CNN** [11] is a website fingerprinting attack that leverages well designed convolutional neural network with statistical features extracted from packets.

5.2 Closed-World Evaluation

Experiment 1: Recognition Performance on Different Classifiers. In this experiment, we leverage both deep learning and machine learning classifiers to construct StableWPF. Datasets with a larger number of pages can better evaluate the performance of classifiers. Therefore, we choose the JD dataset, which has the largest number of pages on the same website. As shown in Table 3, the Random Forest performs well, its accuracy and recall rate are slightly higher than CNN, but precision is lower. CNN achieves the highest precision among all the classifiers, which is 98%. NB and KNN are underperformers, so they are unsuitable for constructing the StableWPF based on the experimental results.

We further compare the training time of these classifiers. As shown in Table 4, KNN costs the shortest time, CNN and NB cost nearly twice than RF. In terms of testing time, RF outperforms the other classifiers. Considering both time cost and performance, in this paper, we chose RF as the classifier of StableWPF.

Table 3. Performance of StableWPF on JD dataset.

	RF	CNN	NB	KNN
Accuracy	0.937	0.925	0.84	0.733
Precision	0.863	0.98	0.701	0.658
Recall	0.873	0.85	0.713	0.66

Table 4. Time Performance of StableWPF on JD dataset.

	RF	CNN	NB	KNN
Training Time	31	63	71	5
Testing Time	1.15	1.23	0.8	2

Experiment 2: Comparison with the State-of-the-Art Method. To evaluate the performance of StableWPF, we compare our method with four different fingerprinting methods on the JD and YH datasets. Results in Fig. 14 and Fig. 15 show that our method performs much better than the baselines in both datasets.

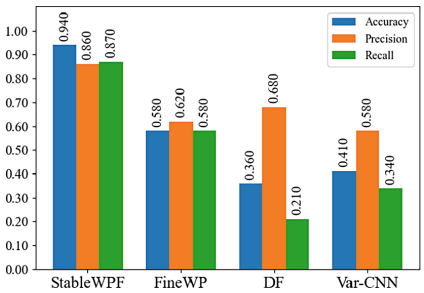


Fig. 14. Comparison with existing methods on JD dataset

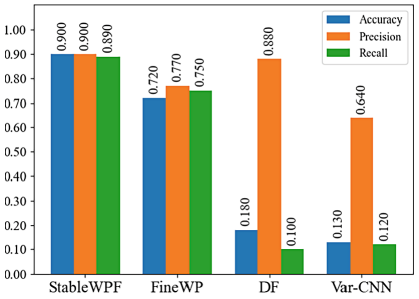


Fig. 15. Comparison with existing methods on YH dataset

As shown in Fig. 14 and Fig. 15, both DF and Var-CNN do not perform well. It is because they are designed for coarse-grained website fingerprinting. Besides, DF and Var-CNN are trained with relatively simple features that only leverage time and direction information extracted from packets. Both models have not considered the impact of unstable networks. If the network condition changes during the period of capturing procedure, features obtained by their methods will fluctuate, which could lead to inefficient classification results. Moreover, these two methods use deep learning models, which require large amounts of training data, but providing a large amount of training data for deep learning models is too costly. The experimental results indicate that these two methods cannot be applied in fine-grained webpage fingerprinting.

FineWP is designed with the goal of fine-grained webpage fingerprinting. As shown in Fig. 14 and Fig. 15, FineWP performs relatively well among baselines but can not reach the accuracy mentioned in paper [15]. Therefore, we design an experiment to further investigate the reason that makes the replication of FineWP less effective.

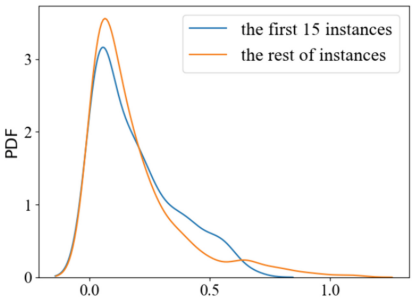


Fig. 16. RTT distribution of dataset 1

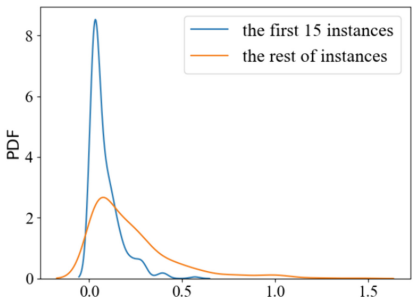


Fig. 17. RTT distribution of dataset 2

To further evaluate the performance of StableWPF and FineWP in realistic fluctuating networks, we construct two datasets with different network conditions and test the methods on the datasets. In dataset 1, all the instances are captured within 2 days. In dataset 2, the first 15 instances of a webpage were captured first, and the rest were captured one week later. We calculated the round-trip time (RTT) of each server in the two datasets by leveraging the timestamps of the ACK packet in the three-way handshake. Figure 16 and Fig. 17 show that the RTT distribution of dataset 1 is stable, while the distribution of dataset 2 is diverse. It is possible that some webpages are put on different servers for load balancing as time passes, which leads to the fluctuation of the traffic pattern. Table 5 shows the performance of StableWPF and FineWP on both datasets. The result shows that FineWP could perform well when the network condition is stable but is less effective when the network condition has changed.

Table 5. Performance of StableWPF and FineWP in different network conditions.

Dataset	Method	Accuracy	Precision	F1-score	Recall
dataset 1	StableWPF	0.98	0.99	0.98	0.98
	FineWP	0.91	0.92	0.91	0.92
dataset 2	StableWPF	0.96	0.94	0.94	0.93
	FineWP	0.68	0.71	0.67	0.73

In conclusion, our method, StableWPF, could obtain stable features and outperform all the baselines in diverse unstable network environments.

Experiment 3: Generalization Ability in the Unstable Realistic Network. To evaluate the generalization ability of our method, we test StableWPF on four different websites with various network environments and compare the accuracy of our method with state-of-the-art methods. The websites we chose have very different Round Trip Time (RTT) distributions when browsing. We

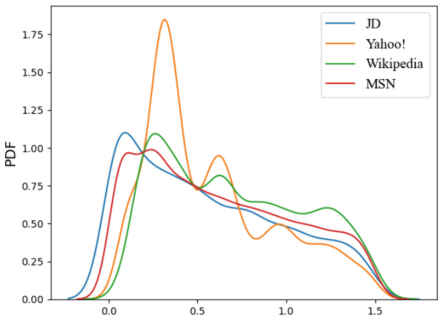


Fig. 18. RTT distribution of users visiting the monitor websites.

calculated the average RTT of the four websites and calculated their probability density in Fig. 18. The average RTT of JD is 0.18 s, 0.58 s for YH, 0.62 s for MSN, and 0.71 s for WIKI. The value for WIKI is nearly three times larger than JD.

As shown in Table 6, although the network conditions are different when accessing these websites, the classification performances of StableWPF are almost consistent. We also show how baselines perform in different websites in Table 7. StableWPF achieves the highest accuracy in all the datasets. The result of this experiment proves that StableWPF can achieve desirable generalization ability in unstable network conditions.

Table 6. Performance of StableWPF on four websites.

	Accuracy	Precision	Recall
JD	0.94	0.863	0.873
YH	0.90	0.896	0.885
MSN	0.92	0.926	0.92
WIKI	0.88	0.887	0.88

Table 7. Accuracy of state-of-the-art methods on four websites.

	JD	YH	MSN	WIKI
StableWPF	**0.94**	**0.90**	**0.92**	**0.88**
FineWP	0.58	0.72	0.73	0.46
DF	0.357	0.18	0.21	0.23
Var-CNN	0.41	0.13	0.31	0.43

Experiment 4: Evaluation on Different Numbers of Instances. In this experiment, we compare the practicability of StableWPF with FineWP, which performs best among baselines. In a practical application of fine-grained webpage fingerprinting, the fewer instances required for training, the lower cost a model takes. Therefore, we compare StableWPF with FineWP on different numbers of instances; 70% of the instances are chosen as training instances. Table 8 illustrates the impact of the number of instances on the classification accuracy. The experimental result shows that StableWPF can reach higher accuracy with fewer instances than FineWP.

Table 8. Comparison of accuracy with different numbers of instances.

Number of Instances	FineWP	StableWPF
15	0.54	0.87
20	0.58	0.91
30	0.61	0.94

The experimental results in the closed-world evaluation show that our method can precisely recognize webpages with fewer instances, and its generalization ability in diverse network environments is also desirable.

5.3 Open-World Evaluation

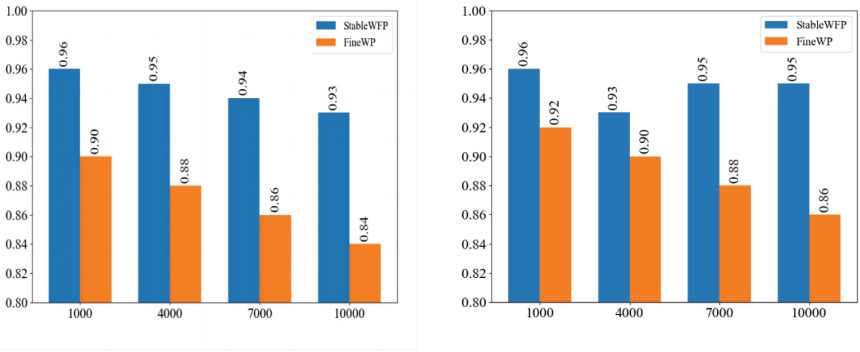
In this section, we experiment on a more realistic open-world dataset OW. Attackers in the open-world scenario need to distinguish whether the traces come from the monitored sites or unmonitored ones. 360 pages on *Jingdong* are selected as monitored pages; 10000 pages that have similar designs and resources within the same website *Jingdong* mixed with 7916 available homepages from AlexaTop 10000 are selected as unmonitored pages.

We first compare the performance of our method with the other three baselines using the full OW dataset, which contains monitored pages on *Jingdong*, similar unmonitored pages within the same website, and homepages from AlexaTop 10000. Then, we evaluate the methods using only the monitored pages and the similar unmonitored pages. As shown in Table 9, StableWPF achieves the best performance in both situations, outperforming other methods when the unmonitored pages are similar to the monitored ones.

Table 9. Open-world: Metrics of different methods

Unmonitored Pages	Performance	StableWPF	FineWP	DF	Var-CNN
similar unmonitored pages + AlexaTop 10000	precision	0.93	0.91	0.90	0.90
	recall	0.98	0.92	0.95	0.86
similar unmonitored pages	precision	0.93	0.84	0.75	0.82
	recall	0.95	0.86	0.98	0.88

To investigate how the size of unmonitored data affects the performance of webpage fingerprinting, we evaluate how our method performs on testing sets with different sizes. Figure 19 illustrates the precision and recall for increasing number of unmonitored instances. Features from unmonitored instances might be similar to monitored instances, for example, two different news from the same section of Yahoo. Such similarities confuse the classifiers, and the impact will escalate as the number of unmonitored pages increases. As a result, precision and recall in Fig. 19 show a gradually decreasing trend with increasing unmonitored



(a) Precision. (b) Recall.

Fig. 19. Comparison with varying numbers of unmonitored pages.

pages. The experiment results show that StableWPF outperforms FineWP in the open-world scenario, regardless of the unmonitored traffic size.

The experimental results in the open-world evaluation show that our method is robust in the realistic scenario.

6 Conclusion

Existing webpage fingerprinting methods are vulnerable to the fluctuation caused by the unstable network environment. In this paper, we propose a fine-grained webpage fingerprinting approach named StableWPF that obtains stable features to recognize the pages within a website. We leverage the frequency information of SoTs to construct webpage fingerprints, with two strategies to ensure their stability. StableWPF shows a great generalization ability on four closed-world datasets and achieves higher accuracy than the state-of-the-art fingerprinting method with fewer instances. In the open-world scenario, our method obtains 94.5% average precision and 94.8% average recall, which also outperforms the similar work of fine-grained webpage classification. In future work, we plan to expand the dataset and use embedding learning to improve the performance of our method.

Acknowledgment. This work is supported by the National Key Research and Development Program of China (2021YFB3101403).

References

1. Google: Google transparency report (2022). https://transparencyreport.google.com/https/overview
2. Hayes, J., Danezis, G.: k-fingerprinting: a robust scalable website fingerprinting technique. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 1187–1203 (2016)

3. Panchenko, A., et al.: Website fingerprinting at internet scale. In: NDSS (2016)
4. Shusterman, A., et al.: Website fingerprinting through the cache occupancy channel and its real world practicality. IEEE Trans. Dependable Secure Comput. **18**(5), 2042–2060 (2020)
5. Leroux, S., Bohez, S., Maenhaut, P.-J., Meheus, N., Simoens, P., Dhoedt, B.: Fingerprinting encrypted network traffic types using machine learning. In: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, pp. 1–5. IEEE (2018)
6. Chen, M., Wang, Y., Xu, H., Zhu, X.: Few-shot website fingerprinting attack. Comput. Netw. **198**, 108298 (2021)
7. Wang, Y., Xu, H., Guo, Z., Qin, Z., Ren, K.: SnWF: website fingerprinting attack by ensembling the snapshot of deep learning. IEEE Trans. Inf. Forensics Secur. **17**, 1214–1226 (2022)
8. Lu, J., et al.: GAP-WF: graph attention pooling network for fine-grained SSL/TLS website fingerprinting. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2021)
9. Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., Joosen, W.: Automated website fingerprinting through deep learning. In: 25th Annual Network and Distributed System Security Symposium. The Internet Society (2018)
10. Sirinam, P., Imani, M., Juarez, M., Wright, M.: Deep fingerprinting: undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1928–1943 (2018)
11. Bhat, S., Lu, D., Kwon, A., Devadas, S.: Var-CNN: a data-efficient website fingerprinting attack based on deep learning. Proc. Priv. Enhancing Technol. **2019**(4), 292–310 (2019)
12. Abe, K., Goto, S.: Fingerprinting attack on tor anonymity using deep learning. In: Proceedings of the Asia-Pacific Advanced Network, vol. 42, pp. 15–20 (2016)
13. Di Martino, M., Quax, P., Lamotte, W.: Realistically fingerprinting social media webpages in HTTPS traffic. In: Proceedings of the 14th International Conference on Availability, Reliability and Security, pp. 1–10 (2019)
14. Shen, M., Liu, Y., Chen, S., Zhu, L., Zhang, Y.: Webpage fingerprinting using only packet length information. In: ICC 2019-2019 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2019)
15. Shen, M., Liu, Y., Zhu, L., Du, X., Hu, J.: Fine-grained webpage fingerprinting using only packet length information of encrypted traffic. IEEE Trans. Inf. Forensics Secur. **16**, 2046–2059 (2020)
16. Shen, M., Gao, Z., Zhu, L., Xu, K.: Efficient fine-grained website fingerprinting via encrypted traffic analysis with deep learning. In: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), pp. 1–10. IEEE (2021)
17. Wang, T., Cai, X., Nithyanand, R., Johnson, R., Goldberg, I.: Effective attacks and provable defenses for website fingerprinting. In: 23rd USENIX Security Symposium (USENIX Security 2014), pp. 143–157 (2014)
18. Mitra, G., Vairam, P.K., Saha, S., Chandrachoodan, N., Kamakoti, V.: Snoopy: a webpage fingerprinting framework with finite query model for mass-surveillance. IEEE Trans. Dependable Secure Comput. **20**(5), 3734–3752 (2022)
19. Jingdong. https://www.jd.com
20. Rahman, M.S., Sirinam, P., Mathews, N., Gangadhara, K.G., Wright, M.: Tik-Tok: the utility of packet timing in website fingerprinting attacks. Proc. Priv. Enhancing Technol. **2020**(3), 5–24 (2020)

21. Yan, J., Kaur, J.: Feature selection for website fingerprinting. Proc. Priv. Enhancing Technol. **2018**(4), 200–219 (2018)
22. Rescorla, E., Dierks, T.: The transport layer security (TLS) protocol version 1.2, RFC 5246 (2008). https://www.rfc-editor.org/info/rfc5246
23. SimilarWeb: SimilarWeb ranking (2022). https://www.similarweb.com/top-websites/
24. Alexa: Alexa top one million sites (2022). http://s3.amazonaws.com/alexa-static/top-1m.csv.zip