This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3104869, IEEE Transactions on Dependable and Secure Computing

Automated Multi-Tab Website Fingerprinting Attack

Qilei Yin, Zhuotao Liu, Qi Li, Tao Wang, Qian Wang, Chao Shen, Yixiao Xu

Abstract—In Website Fingerprinting (WF) attack, a local passive eavesdropper utilizes network flow information to identify which web pages a user is browsing. Previous researchers have demonstrated the feasibility and effectiveness of WF attacks under a strong Single Page Assumption: the network flow extracted by the adversary belongs to a single web page. In reality, the assumption may not hold because users tend to open multiple tabs simultaneously (or within a short period of time) so that their network traffic is mixed.

In this paper, we propose an automated multi-tab Website Fingerprinting attack that is able to accurately classify websites regardless of the number of simultaneously opened pages. Our design is powered by two innovative designs. First, we develop a split point classification method to dynamically identify the split point between the first page and its subsequent pages. As a result, the network traffic before the split point is solely generated for the first page. Then, we propose a new chunk-based WF classifier to infer the websites based on the initial chunk of clean traffic. For both classifiers, we apply automated feature selection to select a concise yet representative feature set. We implement a prototype of our design and perform extensive evaluations using SSH and Tor-based datasets to demonstrate the effectiveness of both our system components individually and the integrated system as a whole.

Index Terms—Website Fingerprinting Attack, Machine Learning, Feature Selection, Traffic Analysis.

1 Introduction

When a client is browsing websites, it is inevitable that the website destinations are exposed to the on-path routers. Organizations owning the routers may try to collect the destinations and clients information for economic benefits, regulations, censorship, etc. Although privacy enhancing technologies, such as Tor, can mitigate those threats by encrypting client network traffic and hiding the real source and destination, an adversary, by observing the encrypted network traffic patterns and utilizing the Website Fingerprinting (WF) technique, is still able to identify the websites browsed by the client.

Website Fingerprinting is a technique used to uniquely identify a website. In general, the *fingerprint* of a website is a combination of networking traffic patterns, such as packet sequences, sizes, intervals, and directions of the traffic, when accessing the website. When applied by an adversary, WF could be used to deanonymize normal users. Yet, it could also be used to assist crime tracking on the dark web. Recently, several studies have demonstrated the effectiveness of WF attacks [2], [3], [4] under common network scenarios. However, Juarez et al. [5] highlighted a critical assumption in these works, which we refer to as the Single Page Assumption: "The attacker knows when each web page starts loading and when it ends." Unfortunately, the assumption does not always hold in practice [5], [6], [7] because users tend to simultaneously (or within a very short period of time) open multiple browsing tabs, for instance, in order to pre-fetch a list of pages on a single website or multiple websites. As a result, Juarez et al. [5] questioned the effectiveness of the WF attacks as they showed that the traditional WF attacks become ineffective without the Single Page Assumption.

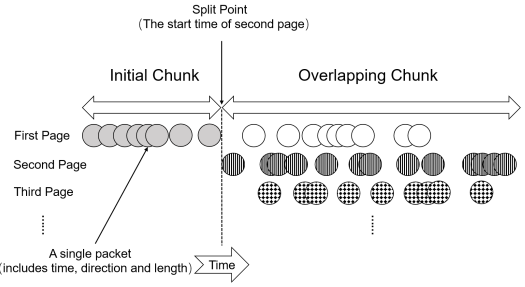

Fig. 1: The illustration of the sequential multi-tab web browsing. The grey circles are the initial chunk of network packets solely generated for the first page. Other circles are network packets generated for different pages when at least two pages are loading simultaneously.

In this paper, we propose a new WF attack that relaxes the Single Page Assumption. Our key observation is that the multiple pages of a website (or multiple websites) opened in different tabs are loaded *sequentially*. As illustrated in Figure 1, regardless of how many pages are opened by a client, until the time point when the second page starts to load, all network packets are solely generated for the first page. We refer to this crucial time point as a *split point*. Based

- *A preliminary version of this manuscript has been published in the proceedings of the 34th Annual Computer Security Applications Conference. 2018: 327-341 [1].*
- *Qilei Yin, Qi Li, Zhuotao Liu, and Yixiao Xu are with the Institute for Network Sciences and Cyberspace of Tsinghua University, Beijing, China 10084, e-mail: yinqilei@tsinghua.edu.cn, qli01@tsinghua.edu.cn, zhuotaoliu@tsinghua.edu.cn, xu-yx16@mails.tsinghua.edu.cn.*
- *Tao Wang is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, e-mail: taow@cse.ust.hk.*
- *Qian Wang is with the School of Cyber Science and Engineering, Wuhan University, email: qianwang@whu.edu.cn.*
- *Chao Shen is with the Department of Electronic and Information Engineering, Xi'an Jiaotong University of China, email: chaoshen@xjtu.edu.cn.*

on this observation, our attack is deployed in two phases. In the first phase, we accurately locate the split point of a multi-tab web browsing session to extract the *initial clean chunk* of the network packets for the first page. In the second phase, we perform website classification only based on the initial chunk of data, bypassing the complexity of analyzing or distinguishing the mixed network traffic generated for different pages or websites. Besides designing classifiers to locate the split points and fingerprint websites using initial data chunks, we further propose a feature selection method to automatically obtain the most representative features for different scenarios, yielding a wide range of advantages for our attack over existing schemes. Note that, the goal of our new WF attack is to accurately identify the first webpage visited by the client from a multi-tab browsing session, as the traffic of the following webpages is mixed and indistinguishable.

Concretely, we make the following major contributions in this paper.

1) We propose an automated multi-tab fingerprinting attack that allows an attacker to classify websites opened with multi-tabs. This is a stronger, yet more realistic, threat model than the Single Page Assumption adopted by prior literature. At the highest level, our design is powered by two innovative designs: a split point finding method to extract the clean and non-mixed network traffic from a multi-tab browsing session, and a website classification mechanism to classify websites only using a small portion of the network packets, rather than the complete packet flow generated during the session.
2) We propose an accurate split point finding method to dynamically identify the split point of the first page and its subsequent pages. We apply a feature selection method to automatically extract a brief yet useful feature set for split point profiling, and then develop a BalanceCascade-XGBoost method to detect the split points under imbalance data scenarios.
3) We come up with a new WF classifier to classify websites only based on the initial network data chunks of each website. Our WF classifier also uses the feature selection method to automatically obtain a small and representative feature set, improving both the efficiency and flexibility of our attack.
4) We implement a prototype of our design and evaluate both our system components individually and the system as a whole. The experimental results show that our split point finding method can identify the split points with high accuracy, and meanwhile demonstrate the performance advantage of our chunk-based classifier over several baseline classifiers. Finally, our integrated multi-tab WF attack can achieve the best TPR of about 0.97 and 0.9 on the SSH and Tor-based two-tab datasets, respectively. The achieved classification accuracy is comparable with the attacks launched for a single webpage, demonstrating the feasibility of multi-tab WF attacks in practice.

The rest of the paper is organized as follows. In Section 2, we present the threat model and introduce related work. We present the framework of our new WF attack in Section 3. We present our BalanceCascade-XGBoost method to split pages in Section 4 and classify pages in Section 5. We evaluate the effectiveness of our attack in Section 6. In Section 7, we discuss the real-world implications and limitations of our attack. Finally, Section 8 concludes our work.

2 Problem Statement and Related Work

2.1 Multi-tab Threat Model

In the WF threat model, the adversary records the encrypted network traffic between the victim and the proxy. To determine whether the encrypted network traffic is generated by a certain website, the adversary constructs a fingerprint database by extracting various network traffic features of the targeted website, such as the directions and sizes of packets. Then, the adversary eavesdrops on the victim's network links and classifies the victim's network traffic using a classifier trained on the fingerprint database.

Typical WF attacks are evaluated under the Single Page Assumption: only one page of the website is visited at a time and no background network traffic is generated. However, this assumption is unrealistic because network traffic generated by the same client is mixed, especially when the client opens multiple tabs simultaneously (or within a short period of time) [5], [6], [7]. As attackers are unable to accurately distinguish the network traffic generated for different pages, current WF attacks become ineffective when classifying websites based on mixed network traffic [5]. Hence, we relax this assumption and extend the threat model to a realistic setting. In the extended threat model, the client can sequentially open multiple pages from a website within a short period of time. As a result, if the first page does not finish loading before the subsequent pages start, their network traffic will be mixed. Under the sequential multi-tab threat model, only the initial chunk of network packets for the first page is clean and therefore faithfully represents the features of the webpage. Thus, in the design of our multi-tab attack, we first accurately identify the clean traffic for the first page, and then classify websites only based on the initial clean traffic rather than the complete (mixed) network flow. For clear presentation, we use the two-tab scenario for illustration. In Section 6.2, we demonstrate the effectiveness of our attack when there are more than two sequential pages.

2.2 Related work

Single Page Website Fingerprinting Attack Single-page website fingerprinting attacks identify websites visited by clients by analyzing the network traffic patterns [2], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. Based on more than 3000 features extracted from network flows, Wang et al. [20] presented a k-Nearest Neighbors (kNN) classifier with weight adjustment, which achieves TPR of 0.85 and FPR of 0.006 on Tor. Panchenko et al. [4] later presented a new approach, CUMUL, which uses SVM with only 104 features. They showed that CUMUL achieves better results than kNN. Hayes et al. [3] created the K-FP attack that utilizes random forests to extract fingerprints for each network flow and then trains a kNN classifier by the fingerprints. This attack shows better results under defenses

compared with Wang's kNN attack and Panchenko's CUMUL attack. Rimmer et al. [21] proposed an automatic Website Fingerprinting attack based on Deep Learning methods. They evaluated three different Deep Learning models including Stacked Denoising Autoencoder, Convolutional Neural Network, and Long Short-Term Memory. Unfortunately, these attacks cannot effectively identify pages under the multi-tab model.

Multi-tab Website Fingerprinting Attack Juarez et al [5] showed that known WF attacks cannot identify two pages that are loaded simultaneously. Two major works have attempted to address this issue. Gu et al. [22] relaxed the assumption about browsing behavior and presented a WF attack on the multi-tab scenario. Using the same extended threat model as ours, they selected fine-grained features such as packet order to identify the first page and utilized coarse features to identify the second page. With a delay of two seconds between two pages, when accessing the top 50 websites using SSH, according to Alexa, their attack can classify the first page with 75.9% TPR, and the second page with 40.5% TPR in the closed-world setting where all the pages are monitored. Compared with this work, our new attack achieves better performance by finding the accurate split points of two pages and selecting a more suitable feature set for website classification.

The work of Wang and Goldberg [7] is most closely related to our approach. They attempted to separate network flows based on noticeable time gaps or a time-based KNN split point finding algorithm. Then, they classified the pages that have been split using the kNN algorithm from [20]. In this work, we make multiple improvements, including proposing an automated feature selection mechanism to select a concise yet representative feature set from a wide range of features to profile the complex and mixed network traffic and an XGBoost based website classifier that excels at classifying websites only based on the initial packet chunks. We have qualitatively demonstrated the superiority of our new design over [7] based on extensive experiments. For instance, our method improves the accuracy of identifying split points on Tor-based multi-tab datasets by up to 32.3%.

3 Overview of Multi-tab Attacks

The Single Page Assumption is widely used in the existing proposals on WF attacks. However, this assumption is unrealistic as users tend to open multiple pages simultaneously. In this work, we relax this assumption to propose a more realistic WF attack. The key observation of our design is that if clients open a website with multi-tab pages or multiple websites within a short period of time, they typically open the second page (and subsequent pages) after some delay, i.e., multi-tab pages are loaded sequentially. For instance, a client may spend some time on reading the contents of the first page before selecting subsequent pages [6]. Therefore, if our attack was able to accurately locate the point where the second page starts to load, we could use the initial networks packets before the split point as the *clean* data to classify the website, without handling the complexity of analyzing and separating mixed network traffic generated for different websites.

Based on this observation, our multi-tab attack is designed with two major phases, as illustrated in Figure 2. The first phase is to dynamically split the clients' multi-tab web page browsing traffic into two parts so as to obtain the initial clean network data chunks. The second phase is in charge of classifying the initial chunks to infer the categories (websites) of the first web page browsed by clients.

At a very high level, both phases are classification problems and we use machine learning based algorithm to solve them. First, considering the large amount of potential split points and dynamic network conditions, we propose a richer feature set to accurately profile each packet and then utilize a feature selection method to automatically obtain a smaller but useful feature set under different scenarios. Second, we develop a classification method improved by ensemble undersampling approach to effectively handle the class imbalance problem among split points. Finally, we review a large set of website fingerprinting features to pick the ones applicable to initial packet chunks, and we also use feature selection method to automatically obtain a concise yet representative feature set so that the cost of feature extraction and classifier training time will be reduced significantly.

Note that our design is not limited by the number of tabs opened during a multi-tab web browsing because we only need to find the split point of the first web page and its subsequent web pages. We have experimentally demonstrated the applicability of our attack against different numbers of sequential tabs in Section 6.2.

4 Dynamic Page Split

4.1 Challenges in Identifying True Split Points

In this section, we highlight several challenges for dynamic and accurate split point identification in sequential multi-tab web browsing. Since a client often needs to send an outgoing packet to request a new page from the web server, we could identify an outgoing packet as the split point in one multi-tab browsing session. In our preliminary work [1], we characterize each outgoing packet to exhibit the difference between the start point and other outgoing packets based on the features proposed by [7]. However, such a static feature set has two drawbacks. First, given the complexity and unpredictability of networking conditions, using a fixed set of features may not yield consistently good performance across different browsing sessions. Second, considering the vast amount of network packets that an attacker may observe, being able to reduce the number of required features for classification would further improve the efficiency and flexibility of our attack. Therefore, to overcome these two drawbacks of our preliminary work [1], we propose a dynamic feature generation method to generate a new set containing 110 features and apply an automated feature selection method to obtain a concise yet representative feature set, for ensuring a good performance across different browsing scenarios and improving the efficiency as well as flexibility of our attack.

The second challenge we face is the classical data imbalance issue because there is only one "true split" and potentially many "false splits" in one instance of network flow data. For example, according to our original study [1],

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3104869, IEEE Transactions on Dependable and Secure Computing

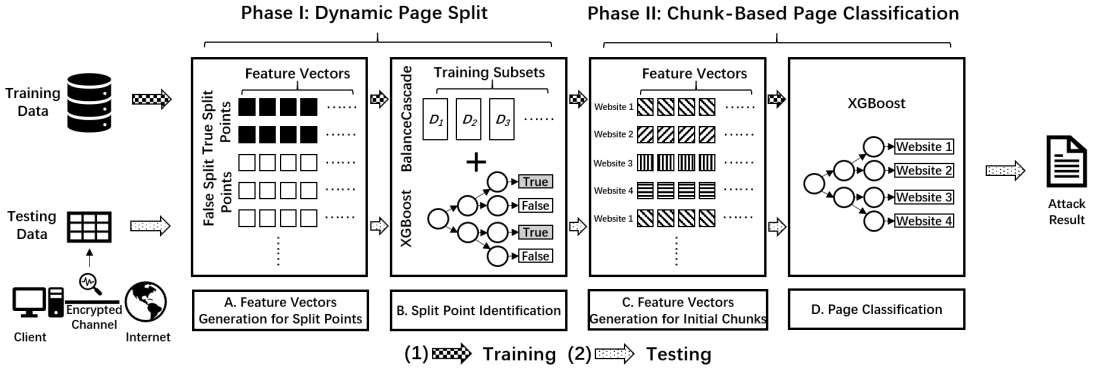

Fig. 2: The overview of our multi-tab website fingerprinting attack.

the proportion of "true splits" instances size and "false splits" instances can reach 1:461. As a result, even though a dumb classifier predicts all the testing instances as "false splits", it can still achieve an accuracy rate as high as 99.78%, yet such a classifier is completely useless. Therefore, how to train a classifier for split point identification under very imbalanced data sources is another technical challenge.

4.2 Dynamic Feature Generation and Selection

To automatically obtain a smaller yet useful feature set for the split point identification, we first enrich the original features in [7] to acquire a more comprehensive and detailed characterization of each outgoing packet. Then, we adopt a feature selection method to filter out the redundant or less effective features while keeping the most significant ones.

Our feature enrichment design works as follows. For each original feature studied in [7], we abstract one feature generation method with a specific parameter. For instance, for the feature 'the five inter-packet times around the candidate packet'[1], 'the number of inter-packet times' is a feature generation method, and 'five' is a parameter for the method. Then we expand the values of the parameters in each feature generation method to obtain a wider range of features than the original feature set. Thus, instead of adding unjustified new features, our feature enrichment design does preserve the original features that have demonstrated reasonably good performance in [1] and meanwhile transform the originally fixed set into a dynamic feature set through parameterization. We determine the parameter ranges based on two considerations: 1) Compared with a single value, a wider range represents a more detailed characterization of each outgoing packet. 2) The packets far away apart (e.g., the 1st and 100th packets in one session) are typically less correlated since they may attribute to different HTML elements in one webpage. Therefore, it is unnecessary to characterize a packet with the distant ones (i.e., we do not need to increase the parameter ranges monotonically).

Specifically, for each kind of feature, we start from a small range around the parameter value given in [7], take our method to enrich new features, select representative features, and then identify split points. Next, we increase the parameter range for another iteration. We stop this procedure when we observe a stable performance and then use the corresponding parameter range. After feature enriching, we extract the following features for each outgoing packet:

1) 20 inter-packet times around the current packet. (20 total features)
2) The mean, standard deviation, and maximum inter-packet times for the 50, 45, 40, 35, 30, 25, 20, 15, 10, and 5 packets before and after the current packet, and the times between the current packet and the packets 50, 45, 40, 35, 30, 25, 20, 15, 10, and 5 packets before the current packet. (3*10 + 1*10 total features)
3) The time between the current packet and the next 1st, 2nd, 3rd, 4th, and 5th incoming packet. (5 total features)
4) The time interval between the packet two packets after the current packet and the packet two packets before the current packet; the packet four packets after and four packets before; and so on, up to 50 packets. (25 total features)
5) The number of incoming and outgoing packets in the 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50 packets before and after the current packet, respectively. (2*10 total features)

In total, we extract 110 features for each outgoing packet, a 4x increase compared with the original 23 features used in [7]. Next, we apply the Recursive Feature Elimination [23] (RFE) technique to select the most significant features such that a smaller number of features are utilized for split point identification. RFE takes a recursive step to eliminate the minor features. Given a training set and a specific estimator, RFE initially utilizes all features to build a classification model, evaluates its performance, ranks each feature based on the current result, removes the least significant feature(s), and then uses the remaining features to start the next loop. However, in the basic version of RFE, it is challenging to decide the right time to stop the iteration in RFE. Thus, we use an improved version of RFE named RFECV, which applies the cross-validation method to automatically determine the number of final features. For the estimator, we choose the Decision Tree algorithm due to its high efficiency and applicability to nonlinear data. Finally, considering that the imbalance of our split point dataset may have a negative effect on the optimal feature number determination of RFECV, we use the AUC metric to assess the estimator

1. To unify terminologies, we change the original "cell" in [7] to "packet".

performances under different number of features, as its effectiveness has been justified in Chen et al. [24].

4.3 BalanceCascade-XGBoost Method

To address the dataset imbalance challenge, we develop a split finding method, named BalanceCascade-XGBoost, which is a binary classifier consisting of the BalanceCascade method [25] and the XGBoost algorithm [26].

Balanced Training Data Generation. BalanceCascade is an ensemble undersampling method that is able to incrementally construct a series of adjusted training subsets from the original dataset. Denoting the original training dataset extracted from all the training network traffic as D, the subset of *true-splits* instances in D as P, the subset for *false-splits* points as N and the ratio of $|N|$ to $|P|$ as b:1, in the i-th round of BalanceCascade, it randomly selects N_i instances from N to create a training subset D_i including N_i and P, where $|N_i|=|P|$. Then, it trains a kNN classifier [27] with default parameter $k=1$ using D_i and removes the instances in N which can be correctly classified by this kNN classifier [2]. In the next round, BalanceCascade continues to create another training subset D_j from the updated N and P. In the end, we obtain a collection of $D_i, i=1...n$ training subsets, $D_i=\{(x_j, y_j)\}(|D_i|=2|P|, x_j \in R^m, y_j \in \{0,1\})$, where x_j is a feature vector extracted from its corresponding outgoing packet, $y_j=0$ for the *false-splits* class, 1 for the *true-splits* class, and m is the dimension of the feature vector.

Typically, the number of outgoing packets in the *false-splits* class is much higher than that of the *true-splits* class. As a result, when generating D during attack preparation, we take random sampling in advance to change its b value, and we will evaluate the performance of our method under varying b values in Section 6.2. Then, we can utilize D and the BalanceCascade method to generate a collection of training subsets $D_i, i=1...n$ for split point classifier training.

XGBoost-based Classifier. Moreover, we utilize the XGBoost classifier [26] to classify websites. XGBoost is a massive parallel boosted tree classifier, which is a widely used machine learning algorithm and is much faster than other methods. The hypothesis function of XGBoost is an ensemble of regression trees [28], whose leaf nodes store class values representing the average values of each leaf node's instances. The ensemble of regression trees will output a real number value and then XGBoost uses the Sigmoid function to convert it to be a value between 0 or 1, where 0 is the probability of belonging to the *false-splits* class while 1 is the probability of being in the *true-splits* class. The XGBoost classifier is able to collect the statistics of each feature in parallel to find the split of regression tree [26]. In particular, for each training subset D_i, we build an individual weak XGBoost classifier f_i and then combine all the weak classifiers to compose the final classifier $F(x)$. The hypothesis function of our final classifier can be computed as follows.

$$F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x). \quad (1)$$

2. As the kNN classifier performs better in our study, we use it in our method to replace the AdaBoost classifier proposed by original BalanceCascade method.

In the testing phase, our classifier checks each outgoing packet from the client and calculates the probabilities for them to be true split points. Finally, our classifier assigns the outgoing packet with the highest probability to be the actual true split point. Note that only the network dataset used for training is sampled and processed by the BalanceCascade method. All the outgoing packets in the testing dataset are processed by our classifier.

5 Chunk-Based Website Classification

In this section, we elaborate on the second phase of our multi-tab WF attack: classifying websites based only on the initial network packets collected before the split point.

5.1 Automated Feature Selection

Because the number of packets in the initial chunk is much smaller than those of the entire network flow, it is necessary to profile them through a rich feature set so as to reveal the subtle uniqueness of the initial chunks from different websites. However, employing a rich feature set comes at a cost. On the one hand, extracting a large number of features from a vast number of packets in high-speed networks is expensive. On the other hand, due to the dynamic changes in websites, an attacker may need to periodically add new training instances, and adjust or even re-train the classifier to retain its accuracy. Thus, we propose an automated feature selection method to obtain a concise yet representative feature set for the chunk-based website classification, improving training agility without sacrificing accuracy.

Specifically, we choose 302 features from prior WF works as the initial feature set based on two principles. 1) Since previous works [3], [4], [20] have demonstrated the significance of data size, intervals, transmission speed, and the number of packets in website fingerprinting, we choose the features relevant to these patterns. 2) For each pattern, we choose different kinds of features to enrich its representation ability. Then we also apply the RFECV method discussed in Section 4 to automatically select a concise and useful feature subset for chunk-based classification.

Compared with our preliminary work [1], there are two significant improvements in this paper. First, we exclude the similarity-based features (e.g., the Fast Levenshtein-like distance (FLLD) [29] and the Jacquard similarity with unique packet length) from our initial feature set. Similarity-based features are computed by measuring the distance between pairs of instances from the training and testing dataset. As a result, their computation is more heavyweight than features that can be directly computed based on the testing instances themselves (e.g., the statistics of packets). Furthermore, they lack transferability as a change in training data will cause the regeneration of these features for all testing data. Second, we replace the original feature selection method in [1] as the feature independence of Naïve Bayes classifiers is unlikely to be true in a complex and practical problem, e.g., website fingerprinting in an encrypted channel. Meanwhile, applying our automated feature selection method in two different tasks further demonstrates the effectiveness and applicability of its design. Eventually, all the features selected during our evaluations are listed as follows.

- First Request Content Size and RTT. We consider the delay between the first outgoing packet and the first incoming packet as RTT. And we compute the First Request Content Size by counting the size of incoming packets between the first outgoing packet and follow-up outgoing packets [16].
- Statistics of packets size and number. The outgoing packets sizes. The ratios of incoming and outgoing packets sizes to total packets sizes respectively. The incoming, outgoing and total packet numbers. The ratios of incoming and outgoing packets numbers to total packets numbers.
- The number of incoming, outgoing packets, the fraction of the number of incoming packets, and the fraction of the number of outgoing packets in the first 20 packets of the network flows.
- We generate two lists by recording the number of packets before every incoming and outgoing packet. Then we compute the average and standard deviation values of these two lists respectively.
- Statistics of packet inter-arrival time. We extract three lists of inter-arrival times between two packets of the network flow for total packets, incoming packets, and outgoing packets. We collect the statistics: maximum, minimum, average, standard deviation, and the third quartile features from each list. Note that, the maximum, minimum, average and standard deviation of inter-arrival times of incoming packets, the minimum, average and the standard deviation of inter-arrival times of outgoing packets, the maximum, average and the standard deviation of inter-arrival times of total packets are selected.
- Statistics of transmission time. We extract all three quartiles from the total, incoming and outgoing packet time sequences. Among them, the first, second and third quartiles of incoming sequence, the first and second quartiles of the outgoing sequence, the first and third quartiles of the total sequence, and the last time in outgoing sequence are selected.
- The quantity and the transmission speed of incoming, outgoing and total packets sequences. For instance, for a packets sequence, we extract a list using 1 to divide each inter-arrival time, and then sample the list to generate 20 features. The 14th sampled features of incoming packets sequence. And the 1-6th, 8th and 11th sampled features of outgoing packets sequence are selected.
- The quantity and the transmission size speed of incoming, outgoing and total packets sequences. For instance, for a packets sequence, we extract a list using each packet size to divide each corresponding inter-arrival time, and then sample the list to generate 20 features. The 1st sampled feature of incoming packets sequence, and the 2-5th, 8th, 13th sampled feature of outgoing packets sequence are selected.
- The cumulative size of packets (CSOP) [4]. We sample 100 CSOP features as recommended by [4]. Note that, the 2-5th, 8th, 15-16th, 37-39th, 43rd, 52st, 56th, 65th, 98-99th CSOP features are selected in the feature subset.
- Burst sizes and quantity. In [20], they defined *burst* as a sequence of outgoing packets, which is triggered by one incoming packet. We sample 20 bursts. We select the size sequence of 20 bursts as the bursts' size features (BSF) and the quantity sequence of 20 bursts as the bursts' quantity features (BQF). Note that, the 1-6th, 9th BQF and the 1-4th BSF are included in the feature subset.

5.2 Classifier Design

With the selected features, we build our classifier based on the XGBoost algorithm since it is an implementation of gradient boosting and incorporates multiple techniques to improve its performance, e.g., the column (feature) subsampling used in Random Forest to prevent overfitting. We replace the original Random Forest classifier in [1] with XGBoost due to the reason that we also want to demonstrate the effectiveness and applicability of XGBoost in two different tasks (i.e., split point identification and chunk-based classification). Unlike the XGBoost classifier used in the split point identification, there are multiple classes (websites) in our training and testing dataset so that we change the objective function of current XGBoost classifier into softmax [30]. Besides, in the preparation stage of website fingerprinting attack, we can collect roughly the same number of training instances (network flows) for each website to create a balanced training set. Therefore, it is sufficient for our chunk-based website fingerprinting attack to utilize a single multi-class XGBoost classifier to infer which website the victim is browsing.

6 Evaluation

In this section, we present our experimental results. Our evaluation centers around the following questions.

- In a multi-tab browsing session, can we accurately locate the split points between the first web page and the subsequent web pages started with arbitrary delays? According to our experiments in Section 6.2, we find that our split point finding method achieves the best accuracy of about 0.902 and 0.959 on SSH and Tor-based two-tab datasets, respectively, outperforming the kNN based approach [7] by non-trivial margins (up to about 32% improvement over TPR). Meanwhile, we achieve almost identical identification accuracy to our preliminary work [1] while we use much fewer features due to our automated feature selection design. Finally, we demonstrate that our method is not limited by the number of simultaneously opened pages: even for the dataset with five tabs opened, our method still delivers identification accuracy higher than 0.74 for both SSH and Tor-based datasets.
- Is it possible to classify web pages only based on the initial chunks containing a relatively small number of network packets? Based on our evaluation in Section 6.3, we observe that our new chunk-based website classifier automatically chooses a concise yet representative feature set and acquires the best TPRs of about 0.957 and 0.843 on SSH and Tor-based single-tab datasets, respectively. Even when the duration of the initial chunk is only around two seconds, the TPRs on SSH and Tor-based single-tab datasets are 0.955 and 0.721, respectively.
- With our integrated multi-tab fingerprinting attack design including dynamic page splitting and chunk-based website classification, is it possible to precisely infer the first website visited by clients based on the dynamically identified split points? In Section 6.4, we demonstrate that our multi-tab WF attack gains the best TPRs of about

0.97 and 0.9 on SSH and Tor-based two-tab datasets, outperforming existing WF attacks (up to about 167.2% improvement over TPR).

6.1 Experimental Setup

Single-Tab Datasets. We collect two datasets: SSH_normal, and Tor_normal. Each data instance is a network flow collected when fetching one website. We select websites based on Alexa[3], a popular website collecting the most visited URLs across the globe, which is widely used in WF studies. The SSH_normal dataset consists of 50 monitored web pages over SSH with 50 training instances and 50 testing instances for each page. There are a total of 100 instances for each page without any background network flow. The SSH_normal data also contains 2500 non-monitored pages chosen from Alexa's top 5,000 websites. We collect the SSH_normal dataset using a headless browser, PhantomJS[4], and use tcpdump to record the network traffic when accessing these websites. Similar to [29], pages are retrieved without caching and all the processes that may generate background traffic are stopped at first. Then we perform the following steps to collect each instance: (i) start the tcpdump; (ii) open the browser and visit the chosen webpage (i.e., URL); (iii) close the browser and related processes after the webpage has been loaded and the loading time is greater than our maximum chunk size in Section 6.3; (iv) close the tcpdump and regard the recorded pcap file as one single-tab instance for the chosen webpage (i.e., label this instance according to the accessed URL); (v) wait for 2 seconds and then start the next loop. By adding a time gap between two consecutive sessions and enforcing check rules (e.g., all flows in the prior session are terminated), we ensure that each collected instance is related to single webpage browsing.

The Tor_normal dataset is collected by programmatically visiting pages using Tor Browser 6.5.1[5]. To collect Tor_normal, we apply the same collection steps used by SSH_normal dataset. We close the browser and related processes at the end of one instance collection, label this instance based on the accessed URL, wait for 2s and then start another iteration. The remaining packets of the last instance will also be removed by checking whether they belong to the flows in the new browsing session. Besides, Tor_normal contains the same set of websites as the SSH_normal. Specifically, it consists of three subsets of web pages: (i) about 50 instances from each of 50 monitored web pages without background noise as training subset, (ii) another 50 instances from each of 50 monitored web pages without background noise as testing subset, and (iii) 2500 instances for non-monitored web pages.

Two-tab Datasets. We collect two datasets, SSH_two and Tor_two, where each data instance contains the network traffic for loading two pages instead of one. In each instance, we visit two pages sequentially and the second website, randomly selected from the monitored websites, is requested after a predefined gap after the first page starts to load.

3. http://www.alexa.cn/
4. PhantomJS is a headless WebKit scriptable with JavaScript: http://phantomjs.org/.
5. https://www.torproject.org/projects/torbrowser.html.en

Since delays of most page retrieval are larger than two seconds [6], we set the minimal gap time as two seconds. In addition, based on our previous observation [1], six seconds' worth of network data is sufficient. Therefore, we choose five different time gaps (two, three, four, five and six seconds) to create in total ten different two-tab datasets (five for SSH and five for Tor). Moreover, as our multi-tab website fingerprinting attack is designed to dynamically identify the split point rather than depending on a static initial chunk size, we add a random delay (0-1s) to the pre-selected time gap to all the instances of the two-tab datasets. For example, when creating the SSH_two dataset with a predefined two-second gap (referred to as SSH_two_2s), the actual gap of each instance is a random value between two and three seconds. Note that, the random delay for each instance in a two-tab dataset is different. Since the actual gap of each two-tab instance is a specific time gap value (2,3,4,5 or 6s) added with a random delay between 0-1s, we can divide the two-tab instances into 5 categories whose time gaps are in the range of 2-3s, 3-4s, 4-5s, 5-6s and 6-7s, respectively.

For each two-tab dataset with a specific initial gap, we randomly choose 50 monitored web pages, collect 50 instances for each of them so that in total the number of two-tab instances is 2500. Since our multi-tab WF attack is designed to identify the first webpage visited by client, we label each two-tab instance based on the first accessed URL. And we also use the method applied in time-kNN [7] to label the "true" and "false" split points. In particular, for each two-tab instance, we modify PhantomJS and Tor to output the exact time when the request for the second webpage is sent, and then mark the first outgoing packet at or after this time as the start of the second webpage, i.e., the "true" split point in this instance.

Data Preprocessing. The essence of the WF attack is a page classification problem, and the effectiveness of the attack is affected by noise. Thus, we perform several operations to remove some of the identifiable noise generated by the underlying networking protocols. In the SSH dataset, if a flow has fewer than 20 packets before the second page loads, we treat it as failed page loading and throw away the instance. In addition, we exclude packets with the lengths of 100, 44, 52, or 36, because these are likely to be SSH control packets. We also exclude TCP ACK packets whose lengths are 0. As for the Tor dataset, we throw away instances with fewer than 75 Tor cells. In practice, an adversary can easily perform similar data preprocessing to reduce noises so that our preprocessing is not an unrealistic enhancement of the adversary's power.

Performance Metrics. In this paper, we use true positive rate (*TPR*) and false positive rate (*FPR*) to measure the effectiveness of our attack. TPR measures how often monitored instances are correctly classified and FPR measures how often non-monitored instances are incorrectly classified as monitored ones.

6.2 Evaluation of Dynamic Page Split

Now we evaluate the performance of our dynamic split point finding method (RFECV+BalanceCascade-XGBoost) under different situations. We compare it with our preliminary work [1] (BalanceCascade-XGBoost) and the time-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3104869, IEEE Transactions on Dependable and Secure Computing

kNN method used in [7] on each two-tab dataset to demonstrate the advantages of our new method. For each two-tab dataset, we randomly choose half of the instances as the training dataset and the rest as testing dataset. We keep three decimal places for all our numerical results. We also utilize the dataset published in [7] to show the generality of our new method.

Accuracy Evaluation with Varying Imbalance Metrics. As we discussed in Section 4.1, the binary classification needs to handle the imbalance between the sizes of the *true-splits* and *false-splits* classes. The BalanceCascade method used in our split point finding method resolves this issue by balancing the quantity of two classes in an ensemble under-sampling method. In Section 4, we define an imbalance metric b to measure the ratio of the *false-splits* class size and *true-splits* class size in training set D. As D is generated by a given b from the original training network traffic, we evaluate how this imbalance metric may affect the split accuracy at first.

We use the split accuracy proposed in time-kNN [7], i.e., if the predicted split point is within the 25 packets before and after the true split point, it is considered as correct. In [7], the authors have demonstrated that extra or fewer packets within this range (i.e., 25) do not significantly influence the following webpage classification. Hence, they used this accuracy in [7]. We choose it for two reasons: 1) We also observe, in our own experiments, that a few missing or extraneous packets around the true split points do not significantly affect the following chunk-based classification. 2) We want to make the performance of our work comparable with time-kNN. Specifically, we evaluate the split accuracy under varying b values on SSH_two_2s and the result is shown in Figure 3.

According to the result above, we can conclude that for each imbalance metric b, the split accuracy of our method is better and much more stable than that of time-kNN. In general, the accuracy of time-kNN decreases as imbalance metric increases, indicating performance degradation for unbalanced datasets. Furthermore, our new method achieves almost the same accuracy as the method used in our preliminary work [1], except for a slightly noticeable degradation when b is 100. However, our new method uses less features: for most imbalance metrics, our new method automatically selects and uses only 5 to 10 representative features, while the number of features used in [1] is 23. Therefore, in the case where the number of network packets is large, our new method has better efficiency.

Figure 3 also shows the trend of accuracy with regards to imbalance metrics. We notice that a very small b may result in reduced accuracy due to the lack of sufficient *false-splits* instances. Meanwhile, continuously enlarging b does not result in further performance while adding more training latency. Thus, we select b=10 as a proper value for the rest of experiments.

Page Split Accuracy with Different Datasets. With the selected b value, we conduct more comprehensive evaluations on all of our two-tab datasets. We list the page split accuracy of our new method, our original method in [1], a combination of Random Undersampling and XGBoost, XGBoost, and time-kNN on five SSH-based two-tab datasets in Table 1, and the results on five Tor-based two-tab datasets are shown in Table 2.

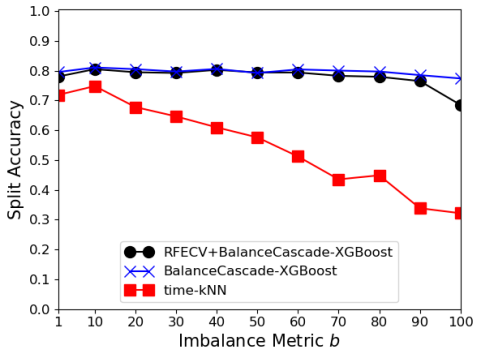

Fig. 3: The split accuracy under various imbalance metrics. Our method achieves similar performance with BalanceCascade-XGBoost while using fewer features, and meanwhile outperforms time-kNN by non-trivial margins.

TABLE 1: Accuracy of dynamic split point identification of five methods on SSH-based datasets. (BX, RU and XG are the abbreviations for BalanceCascade-XGBoost, Random Undersampling and XGBoost, respectively)

Dataset	RFECV+ BX	BX	RU+ XG	XG	time-kNN
SSH_two_2s	0.805	0.810	0.785	0.792	0.748
SSH_two_3s	0.818	0.815	0.758	0.793	0.750
SSH_two_4s	0.832	0.833	0.794	0.826	0.778
SSH_two_5s	0.873	0.869	0.850	0.866	0.810
SSH_two_6s	0.902	0.913	0.879	0.889	0.841

From the results, we can see that the time-kNN method achieves the lowest accuracy for all datasets. Our new method can improve the page split accuracy by about 6.9% to 9.1% on SSH-based datasets and about 22.4% to 32.3% on Tor-based datasets. Besides, our new method outperforms the combination of Random Undersampling and XGBoost (by about 2.5% to 8.7%), and XGBoost (by about 0.2% to 3.2%) on all SSH-based datasets and 4 Tor-based datasets. Further, compared with our original work in [1], our method in this paper achieves the best results in two SSH-based datasets and all the Tor-based datasets, while in the other three SSH-based results, its accuracy is slightly lower than that of BalanceCascade-XGBoost. However, the number of features used in our method is much smaller, as shown in Tables 3 and 4. This is because our automated feature selection design is able to retain the most representative features from a rich original feature set (much larger than the feature set used in [1]). Besides, our smaller feature set can save the training time of BalanceCascade-XGBoost (e.g., on SSH_two_2s, its training time based on the original 23 features and our selected 6 features are about 10s and 5s, respectively). Although RFECV consumes more time of feature selection (about 85s on SSH_two_2s), note that it ensures the effectiveness of selected features and the adversary only needs to perform it when the training set is significantly changed, which is not a frequent operation. Hence, the additional time cost of RFECV is acceptable in practice. Besides, for all the split point identification experiments, we list all the features selected by our RFECV method in Appendix A.

To further demonstrate the generality of our method, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3104869, IEEE Transactions on Dependable and Secure Computing

TABLE 2: Accuracy of dynamic split point identification of five methods on Tor-based datasets. (BX, RU and XG are the abbreviations for BalanceCascade-XGBoost, Random Undersampling and XGBoost, respectively)

Dataset	RFECV+ BX	BX	RU+ XG	XG	time-kNN
Tor_two_2s	0.959	0.958	0.882	0.956	0.751
Tor_two_3s	0.930	0.926	0.887	0.924	0.757
Tor_two_4s	0.880	0.874	0.830	0.872	0.718
Tor_two_5s	0.880	0.879	0.838	0.878	0.665
Tor_two_6s	0.846	0.842	0.797	0.849	0.691

TABLE 3: The number of features selected/used by our work on SSH-based datasets.

Dataset	RFECV+ BalanceCascade-XGBoost	BalanceCascade-XGBoost
SSH_two_2s	6	23
SSH_two_3s	6	23
SSh_two_4s	6	23
SSH_two_5s	11	23
SSH_two_6s	5	23

apply our new method on the dataset used in [7] and the results are shown in Figure 4. In this dataset, the two-page instances are in three classes (Class 1: Positive-time separated, Class 2: Zero-time separated and Class 3: Negative-time separated). The third class is the most difficult one as there is neither a noticeable gap nor a clear pattern of packets indicating the split point [7]. In these classes, our method consistently outperforms time-kNN. Even in the most difficult Class 3 instances, our method still achieves an accuracy of about 0.695.

Evaluation on Datasets with Over Two Tabs. When the client opens more than 2 webpages sequentially, there will be a split point between each webpage and its subsequent webpages. However, recall that the goal of our multi-tab attack is to accurately identify the first webpage visited by client so that it always has to locate the first split point, i.e., the start of the second webpage, no matter how many tabs exist in one instance. To demonstrate the effectiveness of our new split point finding method in those scenarios, we collect datasets with more than two tabs. The collection procedure is similar to that of SSH_two and Tor_two datasets. To create a multi-tab instance, we load the first page, wait for a specific initial time gap plus a random delay, start to load the second page, wait for another random delay, and start to load the third page and so on. In total, we collect 6 different datasets named SSH_three_2s, SSH_four_2s, SSH_five_2s, Tor_three_2s, Tor_four_2s, and Tor_five_2s, where SSH_three_2s means three tabs with an initial gap of 2 seconds (similar meanings applied for other datasets). For each dataset, we randomly select half of the instances as the training set and test on the second half. Note that only the start point of the second web page is in the *true-splits* class and all of the other outgoing packets are in the "false-splits" class, even for the start points of the third, fourth and fifth web page. We list the page split accuracy results in Table 5. Although the split accuracy slightly decreases as the number of loaded web pages increases, our method still achieves decent accuracy of about 0.743 and 0.781 for SSH_five_2s and Tor_five_2s, respectively.

TABLE 4: The number of features selected/used by our work on Tor-based datasets.

Dataset	RFECV+ BalanceCascade-XGBoost	BalanceCascade-XGBoost
Tor_two_2s	5	23
Tor_two_3s	10	23
Tor_two_4s	18	23
Tor_two_5s	4	23
Tor_two_6s	5	23

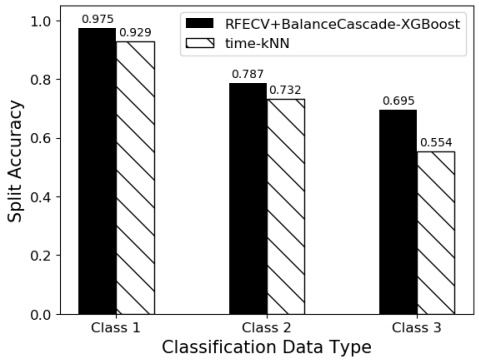

Fig. 4: The split accuracy of our method compared to time-kNN on [7] dataset.

6.3 Evaluation of Chunk-Based Classification

In this section, we evaluate the performance of our new chunk-based WF classifier that only uses the initial and clean network packet streams before the split point. For each dataset, we evaluate five different initial chunk sizes (or durations) ranging from two seconds to six seconds. Furthermore, we perform experiments under the closed-world and open-world scenarios. In the closed-world setting, we test our WF classifier with the dataset where all web pages are monitored, while in the open-world setting, the dataset consists of both monitored and non-monitored web pages. Finally, we compare the performance of our classifier with multiple prior proposals. For a fair comparison, all classifiers are trained and tested using the same data.

Website Classification Under the Closed-world Scenario. We compare our new WF classifier with k-FP [3], CUMUL [4], Deep Fingerprinting (DF) [21] and the WF classifier in our preliminary work [1] on the single-tab datasets: SSH_normal, and Tor_normal. Note that, DF is a new benchmark classifier not compared in our previous work [1]. We clarify that whether the datasets are single-tab or multi-tab does not make a difference in this section because we only use the initial data chunk for classification. The integrated WF attacks, with both dynamic page splitting and chunk-based website classification, are evaluated in Section 6.4. We randomly split each dataset into two halves as the training set and testing set. We faithfully implement the k-FP, CUMUL and DF attacks referring to their published codes[6], as well as the description in their papers. Since

6. k-FP source code: https://github.com/jhayes14/k-FP; CUMUL source code: http://home.cse.ust.hk/~taow/wf/attacks; DF source code: https://github.com/DistriNet/DLWF.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3104869, IEEE Transactions on Dependable and Secure Computing

TABLE 5: The split accuracy on multi-tab datasets.

Dataset	Split Accuracy
SSH_three_2s	0.785
SSH_four_2s	0.759
SSH_five_2s	0.743
Tor_three_2s	0.86
Tor_four_2s	0.846
Tor_five_2s	0.781

the DF attack has three different deep learning models, we evaluate all of them in each experiment and record their best result as the performance of the DF attack.

The TPRs for all classifiers over ten different experiments are given in Figure 5. It is clear that our new WF classifier achieves the best TPR in most cases. Specifically, on all five experiments on SSH_normal dataset, our new WF classifier outperforms all other classifiers and achieves the overall best TPR (over 0.957) when the initial chunk size is five seconds. On the Tor_normal dataset, our new WF classifier obtains the best TPRs on four chunk sizes and only slightly falls behind the classifier in our preliminary work [1] for one size (5 seconds). Also, it is clear that Tor-based web pages are more difficult to be classified since the TPRs for all classifiers drop when classifying Tor_normal dataset. Previous work [16] has also indicated that Tor is more difficult than SSH because all Tor packets have the same size. Further, we observe that increasing the size of the initial chunk does not necessarily result in higher TPRs, e.g, the TPRs of all classifiers drop on the Tor_normal dataset when the initial chunk size increases to five seconds. This may be due to the underlying characteristics of the Tor_normal dataset, underpinned by Tor's Orion routing protocol.

Classification with Open-World Datasets. We further compare our new WF classifier with other classifiers in the more realistic open-world setting using 10-fold cross-validation. We vary the number of the non-monitored pages from 500 to 2500 on SSH_normal training subset and Tor_normal training subset while the initial chunk size also varies from two to six seconds. As shown in Table 9, when the amount of non-monitored pages is 2500, the TPRs of our new WF classifier on SSH_normal dataset are between about 0.934 to 0.941 while the FPR is in the range of 0.011 to 0.018. We observe that the FPRs of all classifiers decrease as the number of non-monitored pages increase. Compared with other classifiers, our method achieves much higher TPRs and only slightly falls behind the K-FP method in FPR. For Tor_normal dataset (shown in Table 10), our new WF classifier achieves TPRs of about 0.799 to 0.897 and FPRs are no greater than 0.002 when the number of non-monitored pages is 2500. Although the Tor_normal dataset is in general more difficult, our WF classifier outperforms all other methods in both the TPR and FPR metrics.

Evaluation of Automated Feature Selection. Besides the classification accuracy, we newly compare the number of selected features by our current RFECV method and our original IWSSembeddedNB method [1] in Table 6. Clearly, our RFECV method always selects a smaller feature set, which represents a faster training time for our classifier. More importantly, our automated feature reduction does not negate the accuracy. We demonstrate this by comparing

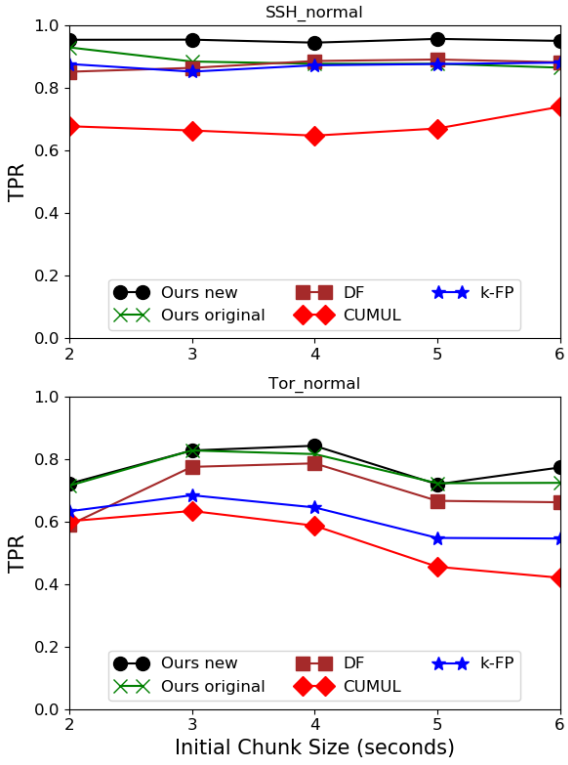

Fig. 5: TPRs of all five evaluated classifiers on five SSH_normal datasets and five Tor_normal datasets.

TABLE 6: The number of selected features for chunk-based classification based on our current RFECV method and the IWSSembeddedNB method in our preliminary work [1]

Dataset & Feature Selection Method		Initial Chunk Size				
		2s	3s	4s	5s	6s
SSH_normal	RFECV	11	11	15	17	7
	IWSSembeddedNB	58	47	40	35	44
Tor_normal	RFECV	12	17	22	10	11
	IWSSembeddedNB	27	47	32	36	38

the performance of our XGBoost classifier before and after feature selection in Table 7. At the cost of a marginal accuracy degradation, our feature selection design significantly reduces the training time of our classifier by 10x.

Another issue worth analyzing is the time cost of feature selection. In our RFECV method, there are two major parameters directly related to its run time: the number of features to be removed at each iteration (step) and the number of folds for cross-validation (cv). Hence, we test six different combinations of the two parameters to obtain their feature selection times, the numbers of selected features and the TPRs on the SSH_normal dataset with initial chunk size as two seconds. The results are shown in Table 8. Typically, with smaller step and larger cv values, our RFECV method is able to perform fine-grained analysis on each feature so that we are able to obtain a more concise feature set, without sacrificing TPR. The selection cost, however, is also higher. We clarify that a fine-grained analysis may not necessarily result in more heavyweight attack preparation over time, since the adversary does not have to repeat the feature selection process for minor or moderate training set adjustments.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3104869, IEEE Transactions on Dependable and Secure Computing

TABLE 7: The TPR and training time of XGBoost classifier before and after our feature selection (TPR,Training Time).

Dataset		Initial Chunk Size				
		2s	3s	4s	5s	6s
SSH_normal	Before	0.965/207.9s	0.957/212.1s	0.947/230.4s	0.956/234.0s	0.960/250.4s
	After	0.955/21.5s	0.955/18.6s	0.946/23.5s	0.958/29.2s	0.951/22.9s
Tor_normal	Before	0.733/139.6s	0.850/185.3s	0.838/205.1s	0.749/235.4s	0.794/245.3s
	After	0.721/15.5s	0.828/19.6s	0.843/28.2s	0.720/24.8s	0.774/23.4s

TABLE 8: The performance of our RFECV method under different parameters.

Metric	Parameter Combination (step,cv)					
	(1,5)	(1,2)	(5,2)	(10,2)	(15,2)	(20,2)
Feature Count	11	12	12	32	47	62
Selection Time	667.0s	263.7s	54.3s	23.8s	16.3s	12.6s
TPR	0.955	0.948	0.955	0.964	0.964	0.964

TABLE 9: The performance of our new WF classifier compared to K-FP, CUMUL, DF and our original WF classifier while varying initial chunk sizes and non-monitored page numbers on the SSH_normal dataset.

Open Page Number	Method	Metric	Chunk Size				
			2s	3s	4s	5s	6s
500	Ours New	TPR	0.947	0.947	0.939	0.949	0.946
		FPR	0.074	0.090	0.080	0.062	0.082
	Ours Original	TPR	0.926	0.913	0.906	0.909	0.905
		FPR	0.204	0.218	0.168	0.162	0.208
	K-FP	TPR	0.880	0.867	0.859	0.871	0.878
		FPR	0.074	0.076	0.058	0.056	0.080
	CUMUL	TPR	0.661	0.618	0.629	0.644	0.685
		FPR	0.236	0.184	0.196	0.196	0.260
	DF	TPR	0.832	0.853	0.857	0.874	0.873
		FPR	0.346	0.356	0.436	0.414	0.352
1500	Ours New	TPR	0.946	0.942	0.939	0.938	0.941
		FPR	0.026	0.025	0.027	0.031	0.029
	Ours Original	TPR	0.912	0.897	0.890	0.896	0.890
		FPR	0.089	0.083	0.087	0.077	0.089
	K-FP	TPR	0.863	0.853	0.842	0.853	0.865
		FPR	0.023	0.023	0.011	0.009	0.014
	CUMUL	TPR	0.550	0.516	0.497	0.496	0.541
		FPR	0.070	0.057	0.067	0.061	0.059
	DF	TPR	0.803	0.818	0.812	0.839	0.841
		FPR	0.184	0.175	0.195	0.196	0.197
2500	Ours New	TPR	0.941	0.936	0.938	0.937	0.934
		FPR	0.014	0.011	0.014	0.018	0.015
	Ours Original	TPR	0.874	0.867	0.872	0.879	0.875
		FPR	0.070	0.057	0.058	0.053	0.054
	K-FP	TPR	0.855	0.844	0.831	0.839	0.853
		FPR	0.010	0.011	0.005	0.007	0.004
	CUMUL	TPR	0.480	0.449	0.398	0.408	0.451
		FPR	0.042	0.029	0.024	0.017	0.022
	DF	TPR	0.786	0.797	0.786	0.825	0.811
		FPR	0.132	0.112	0.128	0.123	0.129

Also, an attacker can either choose a more concise feature set or less training time, or make a balance between them.

Attack Against Defenses. Finally, we stress test the performance of our new WF classifier against defense techniques that may be deployed by clients in the real world. Towards this end, we create SSH_normal and Tor_normal datasets that are modified based on four defense mechanisms described below[7].

- HTTPOS split [31], which utilizes HTTP range requests to obfuscate the sizes of small outgoing and incoming packets, splitting them into random sizes.

7. Our implementation is based on the codes published here: http://home.cse.ust.hk/~taow/wf/defenses/

TABLE 10: The performance of our new WF classifier compared to K-FP, CUMUL, DF and our original WF classifier while varying the initial chunk size and unmonitored page number on the Tor_normal dataset.

Page Number	Method	Metric	Chunk Size				
			2s	3s	4s	5s	6s
500	Ours New	TPR	0.797	0.897	0.886	0.786	0.837
		FPR	0.004	0.0	0.004	0.004	0.006
	Ours Original	TPR	0.775	0.884	0.864	0.769	0.803
		FPR	0.006	0.0	0.006	0.004	0.016
	K-FP	TPR	0.666	0.739	0.726	0.617	0.630
		FPR	0.016	0.024	0.062	0.056	0.078
	CUMUL	TPR	0.615	0.651	0.556	0.440	0.396
		FPR	0.038	0.070	0.09	0.226	0.196
	DF	TPR	0.674	0.821	0.827	0.728	0.705
		FPR	0.008	0.032	0.062	0.2	0.262
1500	Ours New	TPR	0.802	0.898	0.889	0.792	0.843
		FPR	0.001	0.002	0.002	0.001	0.001
	Ours Original	TPR	0.780	0.889	0.860	0.760	0.800
		FPR	0.002	0.001	0.003	0.0	0.001
	K-FP	TPR	0.664	0.733	0.717	0.603	0.615
		FPR	0.005	0.007	0.017	0.013	0.016
	CUMUL	TPR	0.585	0.571	0.460	0.283	0.248
		FPR	0.016	0.018	0.023	0.017	0.016
	DF	TPR	0.670	0.819	0.828	0.719	0.706
		FPR	0.002	0.008	0.015	0.106	0.116
2500	Ours New	TPR	0.799	0.897	0.890	0.788	0.846
		FPR	0.0	0.002	0.0	0.0	0.001
	Ours Original	TPR	0.782	0.888	0.858	0.767	0.803
		FPR	0.002	0.0	0.001	0.0	0.001
	K-FP	TPR	0.657	0.730	0.711	0.596	0.608
		FPR	0.004	0.005	0.008	0.008	0.005
	CUMUL	TPR	0.559	0.531	0.423	0.246	0.188
		FPR	0.015	0.008	0.006	0.004	0.005
	DF	TPR	0.677	0.820	0.829	0.711	0.691
		FPR	0.003	0.005	0.010	0.072	0.083

- Traffic morphing [32], which alters the packet sizes of the client's traffic according to the packet distribution of a target web page, is used as a decoy for the real web page.
- Decoy page [16], which loads a decoy page whenever the client opens a new web page.
- BuFLO [13], which sends packets at a constant size and at regular intervals in both directions.

Table 11 shows the evaluation results. Overall, our method shows improved performance over all other classifiers. However, the performance of all classifiers dramatically decreases when the decoy page defense technique is applied, especially for the Tor_normal dataset. The reason behind this is that the packets in Tor traffic have the same size and the loaded decoy page further damages the temporal and direction patterns that are used for classification. The BuFlo seems to be the most effective defense method against all classifiers: our work can only have the TPR of 0.112 on the SSH_normal dataset and the TPR of 0.02 on the Tor_normal dataset, indicating a totally random class prediction (50 classes and probability 0.02 for each). We list the features selected during defenses deployment in Appendix B.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3104869, IEEE Transactions on Dependable and Secure Computing

TABLE 11: Comparison of the TPR of our new WF classifier with other methods against various defenses when the initial chunk size is two seconds.

Dataset	Method	HTTPOS Split	Traffic morphing	Decoy pages	BUFLO
SSH_Normal (2 seconds)	Ours New	0.961	0.831	0.822	0.112
	Ours Original	0.929	0.789	0.712	0.108
	K-FP	0.842	0.712	0.693	0.112
	CUMUL	0.733	0.605	0.459	0.071
	DF	0.815	0.721	0.598	0.067
Tor_Normal (2 seconds)	Ours New	0.678	0.709	0.159	0.02
	Ours Original	0.678	0.716	0.099	0.02
	K-FP	0.570	0.630	0.115	0.02
	CUMUL	0.591	0.601	0.091	0.02
	DF	0.577	0.589	0.061	0.02

6.4 Evaluation of the Integrated Multi-tab WF Attacks

In this section, we evaluate the integrated multi-tab WF attack with dynamic split point identification and chunk-based classification.

The Multi-tab Attack Performance. For each two-tab dataset, half of the instances are used as the training set for our split point classifier, and the remaining half instances are used as the testing set. The detected initial chunks of the testing instances will be further classified by our chunk-based classifier trained on the single-tab dataset with the same initial chunk size as the current two-tab dataset. For instance, when experimenting with the SSH_two_2s dataset, the chunk-based classifier is trained by the SSH_normal dataset with the chunk size as two seconds. We use the initial chunks of our single-tab datasets to train chunk-based classifiers because each single-tab instance contains only the network traffic generated for one website. For a fair comparison, we use the same training datasets and testing datasets for all benchmark WF classifiers. We plot the TPRs for all classifiers in Figure 6 on all ten two-tab datasets. Clearly, our method outperforms all other benchmark classifiers by large margins (up to about 167.2%). Even compared with our preliminary design in [1], our new design delivers improved TPRs over nine datasets (by up to about 22.8%) except for the Tor_two_2s dataset where our new method has slight (less than 1%) performance degradation.

We do realize that all benchmark classifiers (except for our original classifier in [1]) are designed based on the single page assumption. Thus, it is seemingly unfair to feed them the two-tab test datasets with mixed network traffic generated for multiple pages. This, however, is not unfairness. Rather, it demonstrates that these classifiers become ineffective when classifying the multi-tab, arguably more realistic, network traffic. For completeness, we set up a new evaluation where the three benchmark WF classifiers make their classification on the initial chunks detected by our split point finding method, i.e., we combine our split point method with these three WF classifiers as three new multi-tab WF attacks. We also compare them with our new WF classifier and our original classifier in [1] in Figure 7. Clearly, even enhancing the benchmark classifiers with our split point identification, our new WF classifier still has the optimal overall TPRs across different datasets.

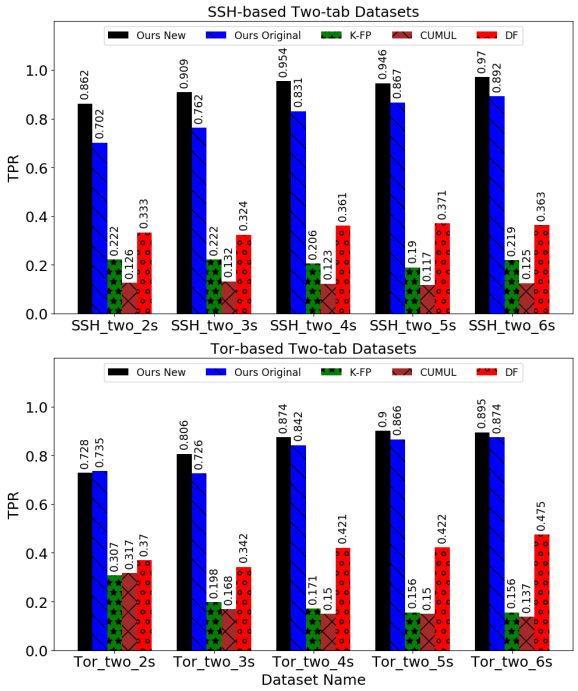

Fig. 6: The multi-tab attack performance for all classifiers.

TABLE 12: The average testing time (in milliseconds) for each instance of SSH_two_2s and Tor_two_2s.

Dataset	Ours New	Ours Original	K-FP	CUMUL	DF
SSH_two_2s	19.149	20.839	0.131	0.342	0.328
Tor_two_3s	17.909	21.601	0.126	0.049	0.247

Additionally, we show the average testing time (in milliseconds) for each instance of SSH_two_2s and Tor_two_2s in Table 12. Since both our new and original WF attacks contain split point identification and chunk-based classification, they take more testing time than the benchmark classifiers designed for single page identification. *However, we clarify that real-time deanonymization is not a must for website fingerprinting attack*. Typically, the adversary collects the encrypted traffic of clients by mirroring the user traffic and performs deanonymization offline, since the goal of WF attack is not to disrupt user browsing sessions. Further, our average testing time is fairly small compared with the typical delays for most webpage retrieval, i.e., 2 seconds [6].

Mismatch between the Initial Chunk Sizes of the Training and Testing Datasets. So far, in all our experiments, we have used the similar initial chunk size for both training and testing datasets. For instance, when experimenting with the SSH_two_2s dataset, the chunk-based classifier is also trained by the SSH_normal dataset with the chunk size as two seconds. In this section, we study the performance of our attacks when the initial chunk sizes for training and testing datasets are different. Towards this end, we apply a chunk-based classifier trained on single-tab Tor_normal datasets with initial chunk size ranging from two to six seconds (in total five classifiers) to classify the Tor_two_3s and Tor_two_5s datasets. The results are plotted in Figure 8.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3104869, IEEE Transactions on Dependable and Secure Computing

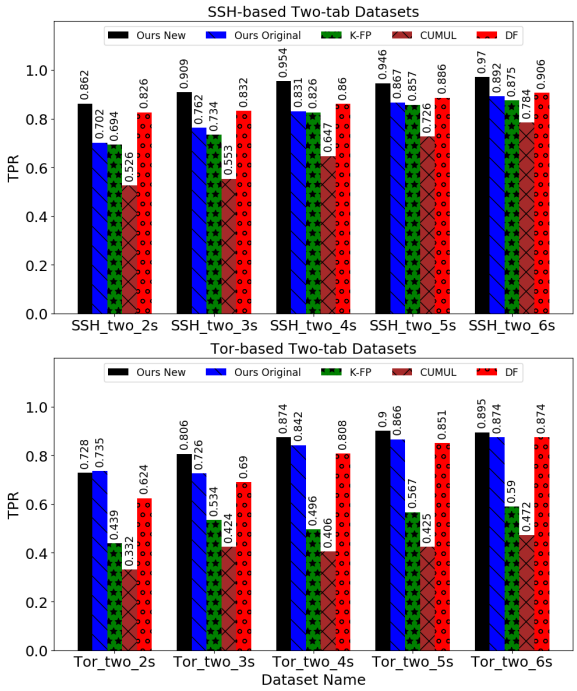

Fig. 7: The multi-tab attack performance for all classifiers. The three classifiers designed based on the single page assumption are enhanced by our split point identification design.

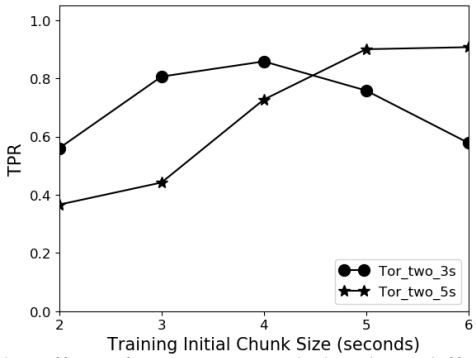

Fig. 8: The effect of training initial chunk in different sizes on both Tor_two_3s and Tor_two_5s datasets.

We observe that when the initial chunk sizes for the training dataset are close to that of the testing dataset, our chunk-based classifier tends to achieve better TPRs.

Real-World Multi-tab Attacks. However, in reality, due to heterogeneity of client behavior across different web browsing sessions, the split points of these sessions are unknown a priori. Our previous work [1] did not point out how to avoid the mismatch pitfall we just evaluated in practice. To handle this challenge, the adversary can prepare a list of chunk-based classifiers trained on different initial chunk sizes so that it can dynamically select the proper website classifier based on the split point identified by our split point classifier. Thus, by assembling a single split point classifier and multiple chunk-based website classifiers, our multi-tab attack can effectively adapt the network dynamics in real-world deployment.

7 Discussion

In this section, we discuss the real-world implications and limitations of our website fingerprinting mechanism.

Real-World Implications. At a very high level, the design of our website fingerprinting mechanism may have positive effects in the following two areas. First, the normal clients can utilize our attack to assess the effectiveness of their anonymous network proxies (e.g., Tor) using more realistic multi-tab browsing settings. Considering the heterogeneity of client network conditions, it is more preferable to perform individual assessments rather than relying on a common third-party evaluation report, which is typically outdated due to the evolution of both anonymous network proxies and defenses. Second, dark web cybercrime is on the rise[8] and imposes a great threat (e.g., terrorism[9] and illegal weapon sales[10]) to our cyberspace. They often deploy various anonymization techniques (e.g., using Tor) to hide their identities. With our website fingerprinting mechanism, the judiciary authorities have enhanced capability to track criminals on the dark web.

Limitations. Specific countermeasures would reduce the effectiveness of our mechanism. For instance, eliminating the explicit traffic patterns may make the statistical features of different websites indistinguishable. However, a recent study [17] shows that the advanced neural network model is still able to extract implicit yet useful features for WF attacks even under the deployment of these defenses. Hence, incorporating new neural network models into our work would be an interesting future work. Another factor that may limit the effectiveness of our mechanism is streaming data, e.g., video, music and torrent, which acts as a background webpage with a long duration so that we can not locate the clean chunks of other webpages. However, we recognize two caveats. First, large file transfers are discouraged on anonymous network proxies like Tor [20] due to limited bandwidth. Second, many torrent applications may leak the client's real IP address and chosen port[11], which seriously undermines the anonymity of network proxies in the first place. Further, prior work has shown that multi-label learning is able to fingerprint SDN applications in totally mixed encryption traffic [33]. We will evaluate its effectiveness for website fingerprinting in our future work.

8 Conclusion

In this paper, we describe an automated multi-tab website fingerprinting attack, which relaxes the Single Page Assumption that current WF attacks rely on. Our attack has two phases. In the first phase, we develop a method to find the split point between the first page and its subsequent web pages, extracting the initial chunk of clean network traffic for the first web page. In the second phase, we classify websites based on our chunk-based classifier. The

8. UNODC report https://www.unodc.org/southeastasiaandpacific/en/2021/02/darknet-cybercrime-southeast-asia/story.html
9. https://www.jstor.org/stable/26297596
10. https://www.vice.com/en/article/j5qnbg/dark-web-gun-trade-study-rand
11. Bittorrent over Tor isn't a good idea https://blog.torproject.org/bittorrent-over-tor-isnt-good-idea

experimental results demonstrate that our page splitting method is able to accurately identify the split points, even under imbalance data scenarios, using a small number of features, which significantly outperforms existing page splitting methods. Moreover, we compare our new chunk-based classifier with other website fingerprinting attacks under close and open-world scenarios and achieve the best fingerprinting performance. For instance, even when the split initial chunk size is only 2 seconds, our WF classifier achieves TPR of about 0.955 and 0.721 on SSH and Tor-based single-tab datasets, respectively. Finally, we show that our integrated attack, with dynamic split point identification and chunk-based classifiers, achieves the best TPR of about 0.97 and 0.9 on SSH and Tor-based multi-tab datasets, respectively, yielding non-trivial accuracy improvement over existing attack methods including the one proposed in our preliminary work [1].

REFERENCES

[1] Y. Xu, T. Wang, Q. Li, Q. Gong, Y. Chen, and Y. Jiang, "A multi-tab website fingerprinting attack," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 327–341.

[2] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a Distance: Website Fingerprinting Attacks and Defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 605–616.

[3] J. Hayes and G. Danezis, "k-fingerprinting: A Robust Scalable Website Fingerprinting Technique." in *USENIX Security Symposium*, 2016, pp. 1187–1203.

[4] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website Fingerprinting at Internet Scale." in *NDSS*, 2016.

[5] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A Critical Evaluation of Website Fingerprinting Attacks," in *Proceedings of the 2014 ACM Conference on Computer and Communications Security*. ACM, 2014, pp. 263–274.

[6] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott, "What TCP/IP protocol headers can tell us about the web," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 29. ACM, 2001, pp. 245–256.

[7] T. Wang and I. Goldberg, "On Realistically Attacking Tor with Website Fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 21–36, 2016.

[8] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical Identification of Encrypted Web Browsing Traffic," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 19–30.

[9] A. Hintz, "Fingerprinting Websites Using Traffic Analysis," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2002, pp. 171–178.

[10] M. Liberatore and B. N. Levine, "Inferring the Source of Encrypted HTTP Connections," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 255–263.

[11] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted HTTP streams," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2005, pp. 1–11.

[12] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 227–238.

[13] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 332–346.

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

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

[16] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website Fingerprinting in Onion Routing based Anonymization Networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, 2011, pp. 103–114.

[17] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1928–1943.

[18] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1131–1148.

[19] A. Shusterman, L. Kang, Y. Haskal, Y. Meltser, P. Mittal, Y. Oren, and Y. Yarom, "Robust website fingerprinting through the cache occupancy channel," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 639–656.

[20] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective Attacks and Provable Defenses for Website Fingerprinting." in *USENIX Security Symposium*, 2014, pp. 143–157.

[21] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *NDSS*, 2018.

[22] X. Gu, M. Yang, and J. Luo, "A novel Website Fingerprinting attack against multi-tab browsing behavior," in *Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on*. IEEE, 2015, pp. 234–239.

[23] P. M. Granitto, C. Furlanello, F. Biasioli, and F. Gasperi, "Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products," *Chemometrics and Intelligent Laboratory Systems*, vol. 83, no. 2, pp. 83–90, 2006.

[24] X.-w. Chen and M. Wasikowski, "Fast: a roc-based feature selection metric for small samples and imbalanced data classification problems," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 124–132.

[25] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory Undersampling for Class-imbalance Learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[26] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[27] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[28] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[29] T. Wang and I. Goldberg, "Improved Website Fingerprinting on Tor," in *Proceedings of the 12th annual ACM workshop on Privacy in the electronic society*. ACM, 2013, pp. 201–212.

[30] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

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

[32] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis." in *NDSS*, 2009.

[33] J. Cao, Z. Yang, K. Sun, Q. Li, M. Xu, and P. Han, "Fingerprinting {SDN} applications via encrypted control traffic," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*, 2019, pp. 501–515.

Qilei Yin is currently a postdoctoral in the Institute for Network Sciences and Cyberspace, Tsinghua University. Qilei Yin earned a Ph.D. degree in Information Security from University of Chinese Academy of Sciences (UCAS) in 2020 and a master degree in Computer Science and Technology from Sichuan University (SCU) in 2015. His primary research areas are networking security and mainly focus on malicious traffic detection.

Zhuotao Liu is currently an assistant professor in the Institute for Network Sciences and Cyberspace, Tsinghua University. He has worked as a tech lead at Google, managing Google's private Wide Area Network (WAN) that hyper-connects Google's massive-scale Datacenters across the globe. Zhuotao Liu earned a Ph.D. degree in Computer Engineer from University of Illinois at Urbana-Champaign (UIUC) in 2017 and a B.S. degree in Electrical Engineering from Shanghai Jiao Tong University (SJTU) in 2012. His primary research areas are systems and networking, with special interest in Blockchain infrastructure, Internet security & privacy, system security, and datacenter networking.

Qi Li (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University. He is currently an Associate Professor in the Institute for Network Sciences and Cyberspace, Tsinghua University. He has worked with ETH Zurich and The University of Texas at San Antonio. His research interests include network and system security, particularly in Internet and cloud security, mobile security, and big data security. He is currently an Editorial Board Member of IEEE TDSC and ACM DTRAP, and serves on the technical program committee of numerous prestigious conferences including IEEE S&P and ISOC NDSS.

Tao Wang is an assistant Professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. Tao Wang received a BSc Degree from HKUST in 2010 and MMath and Ph.D. Degrees from the University of Waterloo in 2012 and 2016 respectively. His primary research is in privacy and security, with a special focus on anonymity networks.

Qian Wang is a professor in the School of Cyber Science and Engineering, Wuhan University. His primary research includes AI Security and Privacy, Cloud Security, and Wireless System Security.

Chao Shen is a professor in the faculty of Electronic and Information Engineering, Xi'an Jiaotong University of China. His current research interests mainly include Data-Driven Network and System Security, AI Security, Cyber-Physical System Security.

Yixiao Xu received his master degree from Tsinghua University. His research interests are network security, particular security and privacy in anonymity networks.