IEICE TRANS. INF. & SYST., VOL.E105–D, NO.6 JUNE 2022

PAPER

Gene Fingerprinting: Cracking Encrypted Tunnel with Zero-Shot Learning

Ding LI[†a)], *Nonmember*, **Chunxiang GU**[†], *Member*, *and* **Yuefei ZHU**[†b)], *Nonmember*

SUMMARY Website Fingerprinting (WF) enables a passive attacker to identify which website a user is visiting over an encrypted tunnel. Current WF attacks have two strong assumptions: (i) specific tunnel, i.e., the attacker can train on traffic samples collected in a simulated tunnel with the same tunnel settings as the user, and (ii) pseudo-open-world, where the attacker has access to training samples of unmonitored sites and treats them as a separate class. These assumptions, while experimentally feasible, render WF attacks less usable in practice. In this paper, we present Gene Fingerprinting (GF), a new WF attack that achieves cross-tunnel transferability by generating fingerprints that reflect the intrinsic profile of a website. The attack leverages Zero-shot Learning—a machine learning technique not requiring training samples to identify a given class—to reduce the effort to collect data from different tunnels and achieve a real open-world. We demonstrate the attack performance using three popular tunneling tools: OpenSSH, Shadowsocks, and OpenVPN. The GF attack attains over 94% accuracy on each tunnel, far better than existing CUMUL, DF, and DDTW attacks. In the more realistic open-world scenario, the attack still obtains 88% TPR and 9% FPR, outperforming the state-of-the-art attacks. These results highlight the danger of our attack in various scenarios where gathering and training on a tunnel-specific dataset would be impractical.

key words: *encrypted tunnel, website fingerprinting, user privacy, zero-shot learning*

1. Introduction

Website Fingerprinting (WF) is a traffic analysis attack with the potential to breach the anonymity provided by encrypted tunnels. WF exploits the unique patterns presented by each website on network traffic, i.e., packet timings and sizes, which can be learned by a classifier. The attack is performed by a local eavesdropper who monitors the user's encrypted traffic, e.g., a compromised wireless router or the user's ISP. The attacker trains a classifier with prerecorded website traces and uses the classifier to identify visited websites from observed traffic flows. Recent WF attacks utilize neural networks to automatically extract features of communication patterns, achieving more efficient and reliable classification than traditional machine learning methods.

Despite the powerful capabilities of WF attacks, applying them in practice faces two major challenges. First, the classifier should be able to identify traffic in encrypted tunnels other than the one in which the training data is collected. This is important because collecting sufficient training data in a tunnel setting is resource-consuming and prolongs the time required to produce off-the-shelf classifiers. Moreover, website traces may change over time due to content updates, so attackers need to collect new datasets frequently to avoid significant concept drift [1], [2]. The second challenge is that the classifier identifying sites of interest (*monitored sites*) in the open world should not require training samples of other sites that the user might visit (*unmonitored sites*). Such open-world attacks are not trivial, as the classifier must be able to identify classes it has never seen before.

In this paper, we consider a more realistic scenario as depicted in Fig. 1. The attacker does not need to replicate all possible encrypted tunnels in advance to collect training data, nor does he crawl unmonitored sites to build an approximate open world. To achieve WF across encrypted tunnels, the attacker only needs to train the classifier with monitored site traces collected in plain Ethernet. This scenario requires the classifier to be capable of Zero-shot Learning (ZSL), thus avoids the use of tunnel-specific samples from both monitored and unmonitored sites. In particular, our interest in ZSL is two-fold. Firstly, existing WF attacks have poor transferability between tunnels. The size, number, and timing of packets can be affected by the tunneling protocol and algorithm, causing a classifier that is effective on one encrypted tunnel not to work on another [1], [3]. The user can thus effectively mitigate the attack by regularly updating the tunnel properties. Secondly, to determine a monitored site, WF attacks require training on unmonitored traces. This setting causes the attack performance to be sensitive to the number and selection of unmonitored sites [4], [5]. Given the actual size of the world of websites, these data-sensitive attacks are virtually unusable in practice.

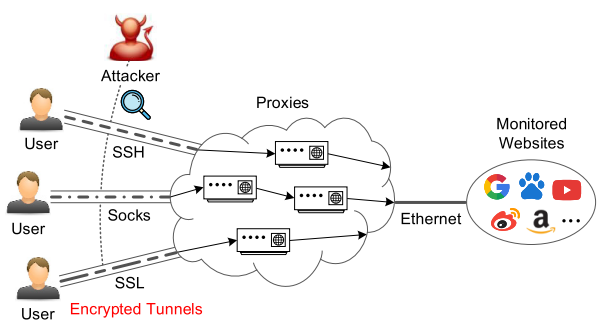


Fig. 1 Clients use various encrypted tunnels to enhance privacy, and a passive attacker tries to identify the website the user is visiting.

Manuscript received August 24, 2021.
Manuscript revised February 14, 2022.
Manuscript publicized March 23, 2022.

[†]The authors are with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, 450001 China.
a) E-mail: liding17@outlook.com
b) E-mail: yfzhu17@sina.com (Corresponding author)

DOI: 10.1587/transinf.2021EDP7179

To the best of our knowledge, this paper presents the first WF attack that leverages ZSL to reduce the labor of time-consuming data gathering and model retraining tasks. The Gene Fingerprinting (GF) attack involves a fingerprint distance measure to enable cross-tunnel identification and an adaptive thresholding function that exempts the reliance on unmonitored traces. The attacker only needs to collect monitored traces in a typical Ethernet channel to achieve open-world classification across tunnels. By design, the attack is robust to tunneling and network distortion. The main contributions of this paper are as follows:

1. Using real-world traffic, we analyze the website fetching process, which inspires the idea of gene fingerprints and explains why existing WF attacks become less effective when using trained models to classify traces collected from different tunnels.
2. We propose Gene Fingerprinting (GF), a more practical WF attack that enables cross-tunnel classification in a more realistic open-world. The attack makes full use of packet timing and size information to create website fingerprints. Based on our distance measure and adaptive thresholds, the intrinsic features are shared among different tunnels to identify the monitored sites.
3. We build seven real-world traffic datasets, including website traces collected over three well-known encrypted tunnels. Based on these datasets, we demonstrate the ZSL capability of our GF attack by comparing it with three state-of-the-art methods in both closed- and open-world evaluations.

The remainder of the paper is organized as follows. In Sect. 2, we describe the motivation of our work, including previous studies, website gene fingerprints, and zero-shot learning techniques. In Sect. 3, we introduce our Gene Fingerprinting attack, while in Sect. 4, we describe our experimental setup and report our evaluation results. Section 5 discusses countermeasures and possible future directions of our attack, and finally, in Sect. 6, we conclude the paper.

2. Motivation

In this section, we first briefly introduce related research in traffic analysis and especially website fingerprinting. Then, we anatomize the website fetching process in both typical Ethernet channels and encrypted tunnels, which motivates us to establish the intrinsic representation of websites as well as our Gene Fingerprinting (GF) attack. We present the detailed design of GF in Sect. 3.

2.1 Prior Work

Traffic analysis is the process of deducing sensitive information from communication patterns, which can be performed on encrypted network traffic. An attacker trains a machine learning classifier by feeding traffic traces of interest and then utilizes the classifier to deduce information from observed traffic, including applications [6]–[9], malware [10]–[12], keyboard input [13]–[15], and even spoken phrases [16]–[18]. Unlike research in computer vision, traffic analysis suffers from the difficulty of acquiring large amounts of labeled data. To address the problem, many studies have proposed methods based on unsupervised learning [19]–[21] or semi-supervised learning [22]–[25]. However, these methods, especially unsupervised clustering techniques, do not provide precise and clear insights into the produced output. Essentially, a (small) amount of labeled data is still required to achieve accurate classification. This paper focuses on supervised learning algorithms and addresses the data-dependent problem through zero-shot learning.

Particularly, Website Fingerprinting (WF) attack is a traffic analysis technique that enables a local passive eavesdropper to recognize the patterns of visited websites. WF was first identified as a threat in 2002 when Hintz successfully broke the privacy provided by an outdated encrypting proxy [26]. Later, more studies have applied WF attacks against specific tunneling proxies like OpenSSH [27], Shadowsocks [3], OpenVPN [28], and Tor [5], [29], [30]. For each tunnel, WF attacks have employed various features as the website representations. Early studies extract a group of fine-grained statistical features from network traces [29], [30]. However, these features provide relatively low accuracy and are susceptible to tunneling and network distortions. Subsequent work instead adopts sequence features derived from packet size [4], timing [31], or direction [5]. Based on the features, WF attacks recognize website traces by training a classifier, such as Naive Bayes [27], [28], Support Vector Machine (SVM) [4], [29], k-Nearest Neighbor (k-NN) [30], [31], etc. Recent advances in WF leverage convolutional neural network (CNN) to design classifiers that have achieved up to 98% accuracy [2], [5].

Although existing WF attacks have achieved fairly high accuracy, it remains a challenge to (i) improve the transferability of classifiers under different tunnel settings and to (ii) identify monitored sites without the training of unmonitored traces. In 2019, Sirinam *et al.* [32] took a step forward on the first issue. Based on the DF model from their previous work [5], they effectively reduced the attack preparation time through N-shot Learning. Nonetheless, this method still requires training data collected from a specific tunnel. Rimmer *et al.* [2] tackled the second issue by introducing a confidence threshold. Specifically, they used a simplified version of the DF model to identify unmonitored sites based on the output cross-entropy and a preset threshold. However, this single threshold neglects the inherent differences between monitored sites. In this paper, we address these issues by proposing GF, a novel WF attack that extracts more intrinsic features of websites and achieves zero-shot learning in a more realistic setting. As we will show, our scheme outperforms existing WF attacks in terms of cross-tunnel classification accuracy and open-world error rates. A summary of the prior work and a comparison with our proposed GF attack is shown in Table 1.

Table 1 Summary of prior work on website fingerprinting attacks and comparison with our proposed attack. CW and OW refer to closed-world and open-world, respectively.

Author	Tunnel Type		Features	Classifier	Sites	CW Acc.	Cross Tunnel	Real OW
	Training	Testing						
Liberatore [27]	SSH	SSH	packet size count	Naive Bayes	1000	75.0%	○	○
Herrmann [28]	SSH, VPN	SSH, VPN	packet size frequency	Multinomial Bayes	100	94.9%	○	○
Wang [30]	Tor	Tor	packet size statistics, packet ordering	k-NN	100	91.0%	○	○
Panchenko [4]	Tor	Tor	cumulative packet size, packet size statistics	SVM	100	91.4%	○	○
Feghhi [31]	Ethernet	Ethernet	packet timing sequence	DTW+k-NN	100	95.0%	◐	◐
Zhao [3]	Shadowsocks	Shadowsocks	packet size/timing statistics, packet ordering	Random Forest	100	98.1%	○	○
Rimmer [2]	Tor	Tor	packet direction sequence	CNN (3 Conv, 1 Dense)	900	94.3%	○	◐
Sirinam [5], [32]	Tor	Tor	packet direction sequence	CNN (8 Conv, 3 Dense)	95	98.3%	○	◐
This work	Ethernet	SSH, VPN, Shadowsocks	packet timing sequence, cumulative packet size	DTW+k-NN	100	97.1%	●	●

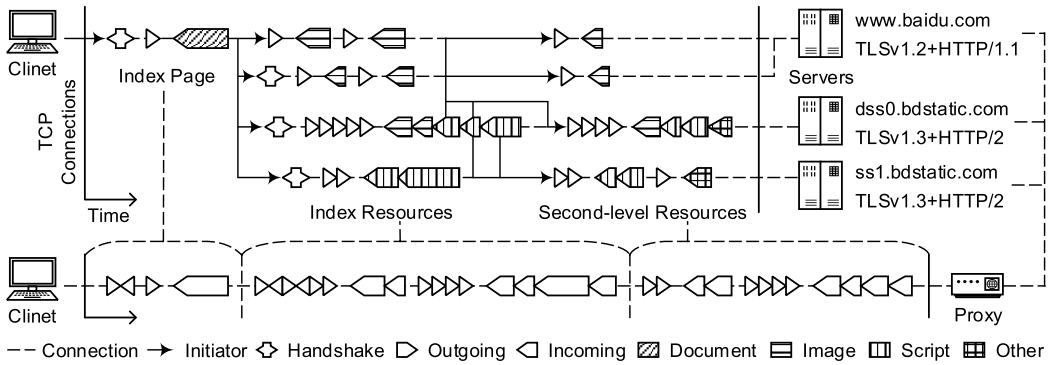


Fig. 2 Client-server interactions for fetching Baidu in an Ethernet channel (top) or through an encrypted tunnel (bottom).

2.2 Anatomy of a Website

When a user visits a website, the browser needs to request various resources, such as stylesheets, scripts, images, etc. Due to the diversity of website content, the number, size and order of resources loaded varies from site to site. We divide the fetching process into three stages based on the resource level. Taking Baidu as an example, the process is illustrated schematically in Fig. 2.

1. *Index Page*. The first stage is to request the index page of the website. To do this, the client first establishes a TLS connection with the primary server and then fetches the page file by sending an HTTP request.
2. *Index Resources*. The second stage is to request resources referenced in the index page. To improve page load speed, web developers usually place these resources on different servers and implements the HTTP/2 protocol to support multiplexing requests. As a result, the client establishes TLS connections with multiple servers and requests index resources in parallel.
3. *Second-level Resources*. The last stage is to request resources referenced in the index resources. These resources are not requested until the browser has parsed the index resources, so there is a temporal pause before this stage begins. Since there is no time conflict with the previous stage, web developers tend to put second-level resources on the same servers as the index resources.

When traffic passes directly through a typical Ethernet channel, the TLS `ClientHello` message exposes the plaintext Server Name Indication (SNI). A passive attacker can accurately identify the website by analyzing the name and number of servers connected by the client. However, when traffic is carried over an encrypted tunnel, all plaintext information, including the server name, is protected. In addition, because data transfers are multiplexed over a single connection, the address and number of servers are also hidden. An attacker sniffing traffic on the encrypted tunnel can therefore only observe the timing, direction, and size of packets, i.e., a sequence of triples $\{(t_i, d_i, s_i)\}$ for $i \in [1 : n]$ where t_i is the timestamp when the ith packet is observed, $d_i \in \{in, out\}$ indicates whether it is an incoming or outgoing packet, and s_i is the byte size of the packet, which can be the payload size of any visible packet layer.

2.3 Gene Fingerprint

To gain insight into the website fetching process, we record the HTTP Archive[†] while capturing the packet traces. Fig-

[†]A JSON file logging the browser's interaction with a website.

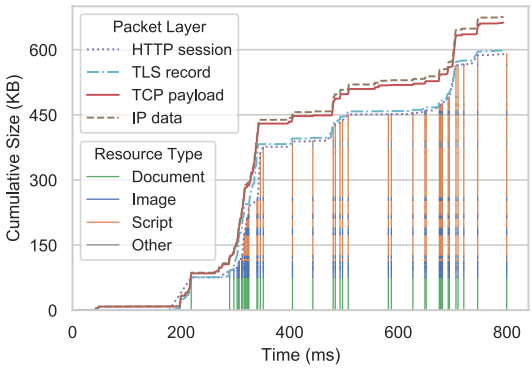


Fig. 3 Visualized correlation between website resources and traffic fingerprints extracted from packet traces.

ure 3 plots the correlation between website resources and incoming traffic. The x-axis indicates the relative timestamp of resource load completion or packet arrival, and the y-axis the cumulative size of requested resources or packet payload. It can be seen that the curves of packet cumulative size coincide with the website resources, and the growth trend of the curves highly corresponds to the three stages of the website fetching process. This observation motivates our interest in whether these curves, which we call *gene fingerprints*, can de-anonymize website traffic in encrypted tunnels.

Prior research [4] has made use of both outgoing and incoming traffic to extract cumulative representations of traces. However, our experiments on the use of outgoing, incoming and bidirectional traffic suggest that outgoing traffic provides no performance boost regarding trace patterns over incoming traffic. The reason is that most HTTP requests contain no payload data (HTTP message body), resulting in extremely similar sizes between websites. Moreover, client diversity factors, such as browsers, operating systems, and devices, have been shown to have significant impacts on the performance of WF attacks [33]. We therefore zero the outgoing packet size to eliminate the client-side differences while preserving the timing information.

The question now is which packet layer is most suitable for extracting *gene fingerprints*. As shown in Fig. 3, the profile of TLS record best matches the loaded resources, with the subtle difference being the 5-byte header; the profile of TCP payload is higher because the payload contains additional handshake data for establishing a TLS connection; the profile of IP data differs from the inner layer by the extra TCP header. In an encrypted tunnel, an attacker cannot obtain the size of the TLS record because the TLS header containing the length field is protected. On the other hand, proxies working at the session layer do not encapsulate TCP headers. Considering the above constraints, we finally take TCP payload as the source to derive fingerprints.

Based on the above analysis, the *gene fingerprint* of a given website trace $\boldsymbol{b} = \{(t_i, d_i, s_i)\}$, $i \in [1 : n]$ can be described as $G(\boldsymbol{b}) = \{g(t_i)\}$, where

$$g(t_i) = \begin{cases} 0, & \text{for } i = 0, \\ g(t_{i-1}), & \text{for } i > 0 \text{ and } d_i = out, \\ g(t_{i-1}) + s_i, & \text{for } i > 0 \text{ and } d_i = in. \end{cases} \quad (1)$$

Distinguishability Across Websites. In Fig. 4 (a), we visualize the fingerprints of five popular websites. As we can see, different websites can be well distinguished even by the human eye, and the fingerprints of the same website are consistent in terms of growth trend and pause position, with slight differences in temporal progression. The loading behavior of Dangdang, a shopping site that exhibits various recommended products, shows a more significant variation due to several resources of dynamic images.

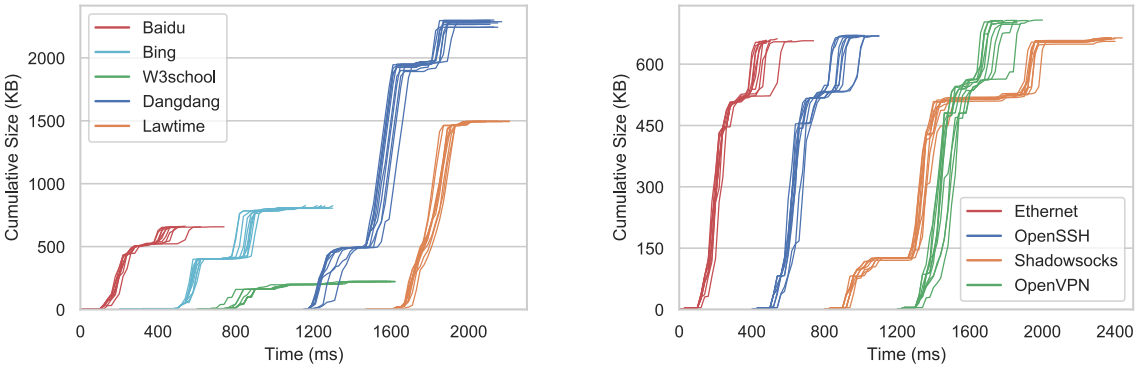
Uniformity Across Tunnels. In Fig. 4 (b), we visualize the fingerprints of the same site generated with three well-known tunneling tools. The fingerprints remain highly consistent in each respective tunnel. The cross-tunnel variations stem from the data encapsulation format and the request method: OpenSSH and Shadowsocks work at the session layer (Layer 5 of the OSI model), but SSH pads packets to an eight-byte boundary [34], which hides the exact size of the encapsulated data; Shadowsocks prolongs the request because it creates new TCP connections for every SOCKS5 address (including ports) [35]; OpenVPN works at the network layer (Layer 3 of the OSI model) and encapsulates the entire IP packet. The difference between tunnels varies the amount of payload that a packet of the same size can carry, resulting in a change in packet size and number. This is the root reason why WF attacks that use statistical features [4], [30] or sequence features [5], [31] fail to classify websites across tunnels.

2.4 Zero-Shot Learning

WF attacks have proven to be effective in de-anonymizing website traffic in encrypted tunnels [27]–[31]. However, traditional WF attacks require a large number of labeled samples used for training the classifier, and a model can only make predictions for classes observed during training. This form of learning contrasts with what we usually think of as intelligence. For example, a human fingerprint can be identified among thousands of different fingerprints after learning a few instances, and this ability responds to all angles and distortions. As we analyzed in Sect. 2, current WF attacks are unable to do this. This is a key challenge for WF: How can we build a model that does not rely on unmonitored traces and remains effective under various tunnel settings? This challenge has sparked our interest in implementing the Zero-shot Learning (ZSL) technique [36], [37].

ZSL is an emerging ML procedure in which the classes covered by training samples and the classes intended for classification are disjoint. ZSL tries to recognize an unseen class by combining features of trained classes, which requires intrinsic features that can be shared among similar classes. In terms of WF, the *gene fingerprints* we designed adequately meet this requirement. Based on ZSL, we aim to achieve two improved attack goals, as shown in Fig. 5. We summarize the key differences between ZSL and traditional WF attacks as the following:

- Traditional WF attacks focus on traffic features in spe-



(a) Website Distinguishability (b) Tunnel Uniformity

Fig. 4 Visualized fingerprints of (a) Ethernet samples of five websites, and (b) Baidu samples in Ethernet and three tunnels. Different groups are shifted to avoid overlap.

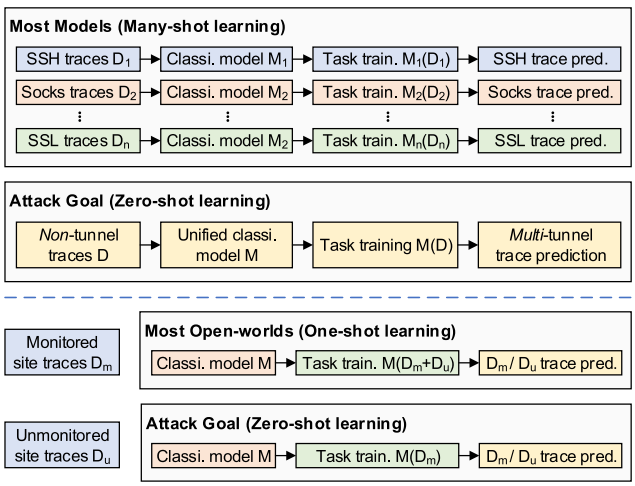


Fig. 5 Two improved attack goals based on zero-shot learning (ZSL).

cific tunnels. Although achieving high accuracy, such a classifier is less portable under different tunnel conditions. In contrast, our ZSL model is trained to learn the traffic features of websites shared across tunnels.

- Traditional WF attacks aim to predict a class within the set of training samples. This requires the training set for an open-world classifier must include unmonitored traces. By contrast, the model in ZSL is trained to measure the similarity between objects and thus can determine a monitored site by setting a threshold.

3. Methodology

In this section, we implement our Gene Fingerprinting (GF) attack by exploiting the consistent nature of *gene fingerprints*. The attack leverages ZSL to achieve the two improved attack goals described in Sect. 2.4. Suppose we have two sequences of fingerprints $G(\boldsymbol{b}) = (g(t_1), g(t_2), \cdots, g(t_n))$ and $G(\boldsymbol{b'}) = (g(t'_1), g(t'_2), \cdots, g(t'_m))$ (defined in Sect. 2.3) of length n and m respectively. To achieve ZSL, we need to measure the similarity between non-tunneled and tunneled fingerprints effectively.

3.1 Tunneling and Network Distortion

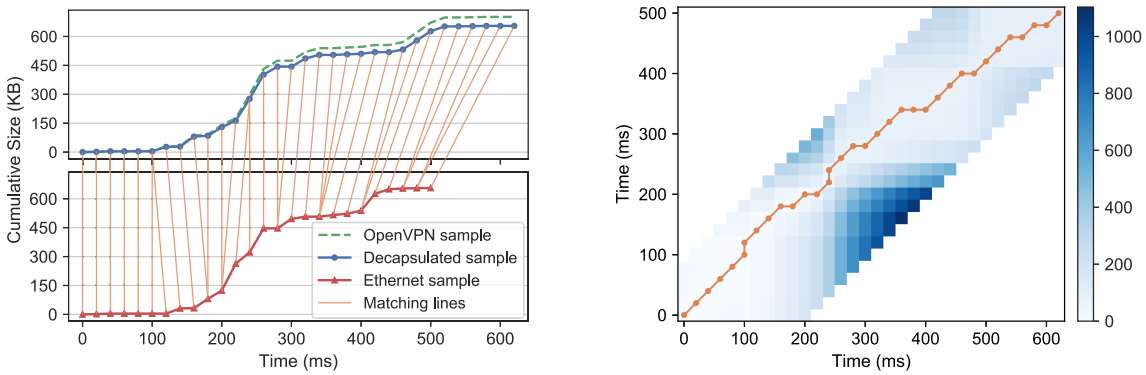
As analyzed earlier, when loading a website through an encrypted tunnel, the network traffic is distorted by the tunnel proxy. The size and number of tunnel packets are affected by the data encapsulation format. There are primarily two types of data encapsulation. The first is *session*-oriented encapsulation, where the proxy encapsulates objects of a session request or response, such as messages in SSH channels [38] and data streams in Shadowsocks [35]. Due to encryption, we cannot determine the session boundaries from tunneled traffic. However, the increased number of bytes is usually negligible compared to the resource size, thus causing little distortion to the fingerprint.

The second type is *packet*-oriented encapsulation: for session-layer proxies, each data stream segment is packed with a few bytes of metadata; for network-layer proxies, the entire IP packet is wrapped into a new header. Regardless of the proxy type, we can roughly determine the number of extra bytes for encapsulation based on a priori knowledge of the tunneling protocol. If packets are encapsulated with random padding, such as SSH, we use the median of possible padding sizes as an approximation. From this, we decapsulate each packet (minus the number of encapsulation bytes) before extracting fingerprints.

The timing of tunneled packets is affected by network conditions. Network jitter may cause packets to experience greater delays or be more crowded together [39]. The proxy's tunneling behavior (e.g., multi-connections in Shadowsocks) and bandwidth limitations can also affect packet timing, tending to extend the time between packets. We therefore need a measure of distance between fingerprint sequences that is insensitive to stretching/compression in the time dimension and can be expected to accommodate temporal differences across tunnels.

3.2 Dynamic Time Warping

Dynamic Time Warping (DTW) was initially used to com-



(a) Sequence Alignment (b) Warping Path

Fig. 6 DTW alignment and warping path example of two Baidu fingerprints, with sampling interval $\delta = 20$ ms and window scale $w = 0.2$.

pare different speech patterns in automatic speech recognition [40]. It has the advantage of automatically coping with time distortions of sequence data. Intuitively, DTW is well suited to measure the similarity between *gene fingerprints*. However, a practical obstacle is that the length of fingerprint sequences is related to the packet number (which can be enormous for a content-rich website). Since DTW needs to calculate the cost between each element pair in two sequences, a direct comparison of fingerprints can be very time-consuming.

To address the issue, we borrow the idea of signal processing and define a new representation of *gene fingerprints* by sampling them at a fixed time interval. This significantly improve the computational efficiency while preserving the timing and size information. Let δ denotes the sampling interval for the fingerprint $G(\boldsymbol{b}) = (g(t_1), g(t_2), \cdots, g(t_n))$, then the new representation can be described as the sequence $\overline{G}(\boldsymbol{b}) = \{\overline{g}(t)\}$ of length $\lceil t_n/\delta \rceil + 1$, where

$$\overline{g}(t) = \begin{cases} 0, & \text{for } t = 0, \\ \max\limits_{t_i \le t} g(t_i), & \text{for } t = \delta, 2\delta, \cdots, \lceil t_n/\delta \rceil \delta. \end{cases} \tag{2}$$

Unless otherwise stated, we use $\overline{G}(\boldsymbol{b})$ to represent the *gene fingerprint* of website trace $\boldsymbol{b}$. The concrete process of gene fingerprint generation is described in Algorithm 1.

We define a sequence alignment that maps points from one fingerprint to another such that the start and end points match and the points are monotonically increasing. Such an alignment can be described as a warping path, which is a sequence $\boldsymbol{p} = (p_1, p_2, \cdots, p_l)$ with $p_k = (x_k, y_k) \in [1 : n] \times [1 : m]$ for $k \in [1 : l]$ satisfying (i) boundary condition: $p_1 = (1, 1)$, $p_l = (n, m)$, and (ii) step-wise condition: $(x_{k+1} - x_k, y_{k+1} - y_k) \in \{(1, 0), (0, 1), (1, 1)\}$ for $k \in [1 : l - 1]$. Further, we use the Sakoe-Chiba Band [40] with the window scale w to constraint the warping path. That is, the warping path $\boldsymbol{p}$ must run along the main diagonal with $|x_k - y_k| \le w \min\{n, m\}$ for $k \in [1 : l]$.

The degree of alignment is assessed by the cost of a warping path, defined as $C(\boldsymbol{p}) = \sum_{k=1}^{l} c(x_k, y_k)$ where $c(x_k, y_k)$ is the element-wise cost for aligning points $\overline{g}(x_k \cdot \delta)$ and $\overline{g}(y_k \cdot \delta)$. We follow the convention and use the Euclidean distance as the element-wise cost. On this basis, we define the distance between fingerprints as the cost of the optimal warping path, i.e., $D(\boldsymbol{b}, \boldsymbol{b}') = \min_{\boldsymbol{p} \in P} C(\boldsymbol{p})$, where P is the set of all warping paths between fingerprints $\overline{G}(\boldsymbol{b})$ and $\overline{G}(\boldsymbol{b}')$. As shown in Fig. 6, the optimal warping path runs along a low-cost "valley" that corresponds to the optimal alignment between two fingerprints. Note that we also tried the F-distance [31], which is based on the estimated curve derivatives. However, we found this distance unsuitable for *gene fingerprints* because the download rate is susceptible to network distortion, especially in encrypted tunnels.

Algorithm 1: Generating Gene Fingerprints

Input : $\boldsymbol{b} = \{(t_1, d_1, s_1), \cdots, (t_n, d_n, s_n)\}$– website trace
δ – sampling interval
Output : $\overline{G}(\boldsymbol{b}) = \{\overline{g}(t)\}$ – gene fingerprint of $\boldsymbol{b}$

$P \leftarrow$ data encapsulation protocol of $\boldsymbol{b}$
if P is tunneling protocol **then**
  $\Delta_P \leftarrow$ number of packet encapsulation bytes
  for packet (t_i, d_i, s_i) in trace $\boldsymbol{b}$ **do**
    $s_i \leftarrow s_i - \Delta_P$
original fingerprint $g(t_0) \leftarrow 0$
for packet (t_i, d_i, s_i) in trace $\boldsymbol{b}$ **do**
  if $d_i = out$ **then**
    $g(t_i) \leftarrow g(t_{i-1})$
  else if $d_i = in$ **then**
    $g(t_i) \leftarrow g(t_{i-1}) + s_i$
sampled fingerprint $\overline{g}(0) \leftarrow 0$
for sample point $j \in \{1, 2, \cdots, \lceil t_n/\delta \rceil\}$ **do**
  $\overline{g}(j\delta) \leftarrow \max\limits_{t_i \le j\delta} g(t_i)$
return $\{\overline{g}(t)\}$

3.3 Exemplar and Adaptive Thresholding

Based on the distance measure $D(\boldsymbol{b}, \boldsymbol{b}')$, we use the k-Nearest Neighbors (k-NN) classifier to identify website traces. Although we can use other classifiers such as Naive Bayes, SVM, CNN, etc., a preliminary study indi-

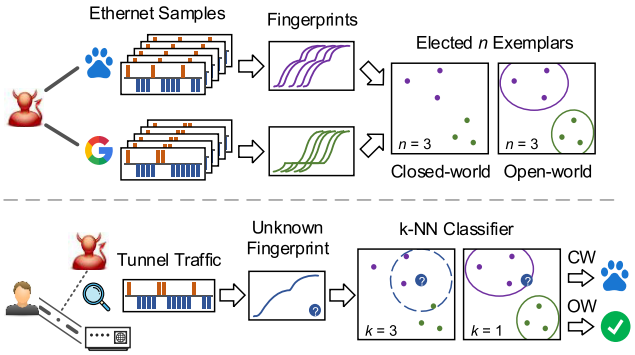


Fig. 7 GF training (top) and inference (bottom) phases.

cated that k-NN outperforms other classifiers for this purpose. For each website w and its Ethernet trace set B_w, we rank the training samples $\boldsymbol{b} \in B_w$ based on sum-distance $\sum_{\boldsymbol{b}' \in B_w} D(\boldsymbol{b}, \boldsymbol{b}')$ and elect n exemplars with the lowest sum-distance to represent this site. Given a testing sample, we sort its distance to the exemplars of all websites in ascending order. The class of the sample is then determined by the majority vote among the top k exemplars. If a tie occurs, the class with the lowest total distance wins.

Note that the closed-world scenario assumes that the user can only visit websites that the attacker has trained on, i.e., monitored sites. In the more realistic open world, however, the user is allowed to visit any website that the attacker has not been able to train on, i.e., an unmonitored site. This leads to a high probability that the attacker will get incorrect labels predicted by the k-NN classifier because he does not have prior knowledge of the complete set of unmonitored sites.

To achieve ZSL in the open world, we additionally define an adaptive threshold to distinguish between monitored and unmonitored sites. For each website w, we calculate the within-class min-distance $D_{in}(\boldsymbol{b}) = \min_{\boldsymbol{b}' \in B_w} D(\boldsymbol{b}, \boldsymbol{b}')$ and out-of-class min-distance $D_{out}(\boldsymbol{b}) = \min_{\boldsymbol{b}' \notin B_w} D(\boldsymbol{b}, \boldsymbol{b}')$ for each training sample $\boldsymbol{b} \in B_w$ and take the average of the ρ quantile of $\{D_{in}(\boldsymbol{b})\}$ and the $(1-\rho)$ quantile of $\{D_{out}(\boldsymbol{b})\}$ as the threshold for this site, where $\rho \in [0, 1]$ is a hyper-parameter. When the distance from a testing sample to the nearest exemplar is less than the corresponding threshold, we classify it as a monitored site; otherwise, it is an unmonitored site. The detailed GF training and inference processes are described in Algorithm 2 and Algorithm 3, respectively.

3.4 Attack Implementation

We now explain the step-by-step implementation of our GF attack. As shown in Fig. 7, this includes the ZSL training and inference phases for both closed- and open-world scenarios. The workflow of the training phase is as follows:

1. The attacker collects traffic samples from each of the monitored sites in a common Ethernet channel.
2. According to Algorithm 1, the collected samples are processed by feature extraction and time sampling to generate *gene fingerprints*.
3. According to Algorithm 2, the extracted fingerprints are elected to generate n exemplars for each website, and these exemplars are used to train a k-NN classifier. In the open world where the victim can access any unmonitored site, the attacker also computes the adaptive threshold for each monitored site.

With the elected exemplars and the derived adaptive thresholds, the inference phase proceeds as follows:

1. The passive attacker captures a traffic sample on the link between the victim and the proxy.
2. According to Algorithm 1, the tunnel traffic sample goes through packet decapsulation, feature extraction, and time sampling to generate an unknown *gene finger-*

Algorithm 2: Gene Fingerprint Training

Input : B – set of Ethernet traces of monitored sites
W – set of monitored sites
n – number of exemplars
ρ – thresholding quantile for the open world
$Q(\cdot,\cdot)$ – quantile function
Output : E – set of elected exemplars
T – set of adaptive thresholds for the open world

$E \leftarrow \emptyset, T \leftarrow \emptyset$
for website $w \in W$ **do**
  for website trace $\boldsymbol{b} \in B_w$ **do**
    $D_{sum}(\boldsymbol{b}) \leftarrow \sum_{\boldsymbol{b}' \in B_w} D(\boldsymbol{b}, \boldsymbol{b}')$
    $D_{in}(\boldsymbol{b}) \leftarrow \min_{\boldsymbol{b}' \in B_w} D(\boldsymbol{b}, \boldsymbol{b}')$
    $D_{out}(\boldsymbol{b}) \leftarrow \min_{\boldsymbol{b}' \notin B_w} D(\boldsymbol{b}, \boldsymbol{b}')$
  $E_w \leftarrow$ top n traces with the smallest D_{sum}
  $T_w \leftarrow (Q(\{D_{in}\}, \rho) + Q(\{D_{out}\}, 1-\rho))/2$
  $E \leftarrow E \cup E_w$
  $T \leftarrow T \cup \{T_w\}$
return E, T

Algorithm 3: Gene Fingerprint Inference

Input : $\boldsymbol{b}'$ – tunnel trace to be classified
E – set of elected exemplars
k – number of nearest neighbors
T – set of adaptive thresholds for the open world
Output : c – predicted class label of $\boldsymbol{b}'$

for website trace $\boldsymbol{b} \in E$ **do**
  $D_{min}(\boldsymbol{b}) \leftarrow D(\boldsymbol{b}, \boldsymbol{b}')$
if in the closed world **then**
  $E' \leftarrow$ top k traces with the smallest D_{min}
  $c \leftarrow$ majority of trace labels in E'
  if $|c| > 1$ **then**
    $c \leftarrow$ label of traces in E' with the smallest $\sum D_{min}$
else
  $\boldsymbol{b} \leftarrow$ trace with the smallest D_{min}
  $w \leftarrow$ trace label of $\boldsymbol{b}$
  if $D(\boldsymbol{b}, \boldsymbol{b}') < T_w$ **then**
    $c \leftarrow$ monitored site
  else
    $c \leftarrow$ unmonitored site
return c

Table 2 Evaluation datasets. Each traffic sample in CSV format consists of raw packet features of timestamp, direction, and payload size. The number of features is taken from the sampled fingerprints.

Dataset	Tunnel	Site Type	Sites	Visits	File Size		# of Features (p=20ms)			Purpose		Scenario	
					PCAP	CSV	Max.	Min.	Ave.	Train	Test	CW	OW
MS-ETH	Ethernet	Monitored	100	10	5.28 GB	151 MB	459	10	88.6	●	○	●	●
MS-SSH	OpenSSH	Monitored	100	10	5.38 GB	159 MB	462	10	93.8	○	●	●	●
UMS-SSH	OpenSSH	Unmonitored	100	10	7.28 GB	210 MB	471	6	151.6	○	●	○	●
MS-SS	Shadowsocks	Monitored	100	10	5.44 GB	174 MB	439	15	135.4	○	●	●	●
UMS-SS	Shadowsocks	Unmonitored	1709	1	7.45 GB	238 MB	461	13	181.2	○	●	○	●
MS-VPN	OpenVPN	Monitored r	1709	1	5.73 GB	184 MB	451	10	99.2	○	●	●	●
UMS-VPN	OpenVPN	Unmonitored	1709	1	8.04 GB	262 MB	471	1	151.5	○	●	○	●

print.

3. According to Algorithm 3, the attacker uses the trained k-NN classifier to predict a label for the unknown fingerprint. If the attacker is uncertain whether the target is a monitored site, he first uses the open-world classifier to predict a (un)monitored label.

4. Evaluation

In this section, we investigate the performance of the GF attack with respect to the desired attack goals. To this end, we create a series of traffic datasets and compare our approach with the state-of-the-art WF attacks.

4.1 Data Collection

To the best of our knowledge, existing publicly available datasets of web traffic are collected from specific tunnels, so we need to re-collect data from different tunnels to verify our attack goals. We select websites from the top 2000 sites ranked by Alexa Top in May 2021 [44]. After pruning sites that fail to load, there are 1809 valid websites, among which 952 use HTTPS by default. Prior works [1], [2], [45] have proved that it is difficult for WF attacks to classify websites that contain dynamic content, such as updating videos and randomized recommendations. For this reason, we take the top 100 sites without dynamic content as monitored sites and the remaining 1709 as unmonitored sites. Note that unmonitored sites in our setup are only used for testing, and therefore are not required to reach the million level to support open-world training.

The data collection was performed in a setting where the client is a Windows 10 20H2 desktop machine, and the proxy is an Ubuntu 20.04 cloud server with kernel version 5.8.0 located in a different city. We created encrypted tunnels using three well-known tunneling tools, with the default settings as shown in Table 3. On the client, we used `Selenium` WebDriver to drive the Chrome browser (version 90.0) to visit websites. This allows for more realistic crawls than using command-line tools because the WebDriver simulates the browsing behavior of a real user. We captured traffic generated by each visit separately using `tshark`. The tunnel packets in each traffic sample are represented as a sequence of triples (*timestamp, direction, payload_size*) and saved into a CSV file. Finally, we created seven traffic datasets, the detailed descriptions of which are shown in Table 2.

Table 3 Tunnel settings for collecting website traffic.

Tunnel	Client	Proxy	Ciphersuite
OpenSSH [41]	windows_8.1p1	8.2p1	chacha20-poly1305
Shadowsocks [42]	windows 4.4.0	libev 3.3.4	chacha20-ietf-poly1305
OpenVPN [43]	connect-3.2.3	as_2.8.7	AES-256-GCM

Note that our datasets differ from existing WF datasets collected from specific tunnels [2], [5], [27], [30]. Classifiers trained and tested on data with the same distribution are susceptible to overfitting, a phenomenon in which a classifier performs well on the training data but has poor performance when new data arrives. Current studies address the problem by creating non-overlapping validation sets using stratified splitting [2], [5] or cross-validation [4], [31]. In contrast, our training and testing datasets are independent, i.e., the training data come from the Ethernet channel, and the testing data are from different visits in three non-Ethernet tunnels. Such datasets inherently avoid overfitting due to similar data distributions and are designed to evaluate the ZSL capability between tunnels. Therefore, we do not need to split the training set to obtain exclusive validation sets.

4.2 Compared Methods

To comprehensively understand the performance of our GF attack, we have selected three representative WF attacks for comparison and re-evaluate these methods on our datasets.

- **CUMUL [4]:** CUMUL attack uses an SVM to classify a set of 104 manually selected features. Specifically, a sequence of the cumulative sum is computed by adding the incoming packet sizes and subtracting the outgoing packet sizes. The final feature set contains 100 points interpolated from the sequence and four statistical features of the total number of bidirectional packets and bytes. Although the number of features is comparable to our GF attack (as shown in Table 2), the features selected by CUMUL tend to characterize the website traffic in specific tunnels.
- **DF [5]:** Deep Fingerprinting (DF) is a recent attack using a deep CNN model. The model takes the sequence features of the first 5000 packet directions as input and utilizes convolutional layers to extract latent features automatically. We follow the hyperparameter selection process for DF to achieve its best performance. While

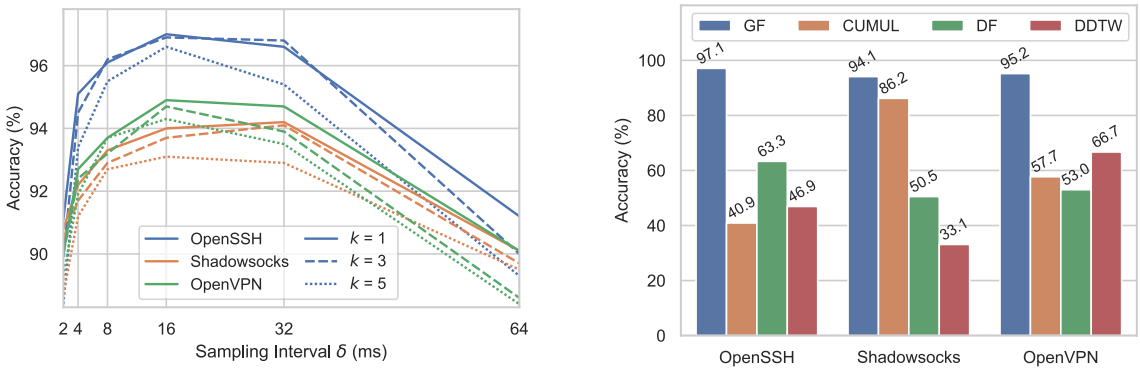


Fig. 8 Closed-world. (a) Impact of sampling interval δ and the number of nearest neighbors k on classification accuracy. (b) Comparison of classification accuracy.

the studies [2] and [32] take a step forward on the two improved attack goals, they use a CNN model similar to DF. Therefore, we compare these two works by evaluating the DF attack instead.

- **DDTW [31]:** This attack makes use only of packet timing information. The timestamp sequences of outgoing packets are first aligned using the Derivative DTW algorithm [46]. Based on the F-distance derived from the optimal warping path, the attack uses a k-NN classifier to identify websites. On our MS-ETH dataset, DDTW extracts an average of 2118 features from each traffic sample, which makes inference time-consuming. Nevertheless, since the attack discards packet size features, it is impervious to various padding mechanisms and has the potential to transfer between different tunnels.

4.3 Closed-World Evaluation

We start by evaluating the attack performance in the closed-world scenario, where the user can only visit the monitored sites. We use classification accuracy as the performance metric. The classifier for each attack is trained on the MS-ETH dataset and tested on the MS-SSH/SS/VPN datasets.

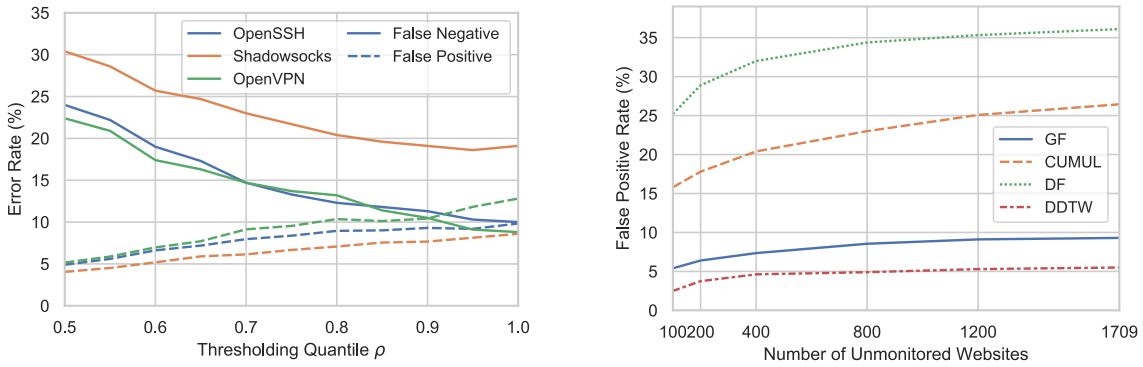
We first conduct an extensive search for the hyperparameters in GF to achieve the best performance. The model is less prone to overfitting during hyperparameter tuning because the training and testing data are independent, i.e., collected from different tunnels and have varying data distributions. The impact of sampling interval δ and the number of nearest neighbors k is presented in Fig. 8 (a), with window scale $w = 0.6$ and exemplar number $n = 6$. The convexity of the curves reflects the fact that a too-small δ can lead to overfitting, while a large δ will lose fingerprint information. We finally determine $\delta = 20$ ms (50 samples per second) and $k = 1$ to reach a balanced accuracy for each tunnel.

Figure 8 (b) shows the accuracy results. Our GF attack attains at least 94.1% accuracy, which is better than the other attacks and shows good cross-tunnel transferability. These results expose the limitations of previously reported attacks that achieved high accuracy in the same tunnel where they trained the classifier. Although CUMUL uses similar features to GF and achieves comparable accuracy in Shadowsocks, the statistical features degrade the transferability across tunnels. As analyzed in Sects. 2.3 and 3.1, data encapsulation in OpenSSH and OpenVPN significantly changes network traffic statistical features such as the total number of packets, resulting in WF attacks employing these features that fail to achieve ZSL in the corresponding tunnels. The deep learning-based DF attack only achieves an average accuracy of less than 60%, highlighting that packet direction sequence features alone are insufficient to support cross-tunnel identification. While DDTW that uses packet timing sequence features achieves 66.7% accuracy in OpenVPN, it has even worse performance (<50%) in other tunnels. This indicates that the timing information of the tunneled packets is severely distorted and cannot capture the website's inherent features. Overall, the experimental results demonstrate that the features exploited by existing WF attacks are inconsistent across different tunnel settings; our proposed GF attack utilizes the intrinsic website traffic features to achieve cross-tunnel identification of monitored sites.

4.4 Open-World Evaluation

We now evaluate the performance of the WF attacks in the more realistic open-world scenario, where the user can visit all websites on the Internet, and the attacker tries to determine whether the web trace is monitored or unmonitored. Considering this is a binary classification task, we use true positive rate (TPR), false positive rate (FPR), and precision as performance metrics. Classifiers are trained on the MS-ETH dataset and tested on a combination of monitored and unmonitored datasets (e.g., MS-SSH+UMS-SSH). It is important to note that unmonitored site samples, in contrast to the setup in previous studies [4], [5], [30], are not included in the training set.

In the open world, our adaptive thresholds naturally turn the model into a 1-NN classifier, i.e., prediction based



(a) (b)

Fig. 9 Open-world. (a) Impact of thresholding quantile ρ on FNR and FPR. (b) Impact of the number of unmonitored testing websites on FPR.

Table 4 Open-world attack performance. WF attacks are tested on a combination of monitored (MS) and unmonitored (UMS) datasets for each tunnel.

Attack	OpenSSH			Shadowsocks			OpenVPN		
	TPR	FPR	Pre.	TPR	FPR	Pre.	TPR	FPR	Pre.
GF	88.7%	9.3%	84.9%	80.9%	7.7%	86.1%	89.5%	10.5%	83.3%
CUMUL	40.6%	26.5%	47.3%	75.1%	32.7%	57.3%	51.1%	17.2%	63.5%
DF	51.5%	36.1%	45.5%	42.6%	34.0%	42.3%	45.5%	51.1%	34.3%
DDTW	29.9%	5.6%	75.9%	12.6%	4.9%	60.3%	51.2%	9.8%	75.4%

on the nearest neighbor. We have verified the good performance of the classifier when $k = 1$. Like hyperparameter tuning in the closed world, the training and testing datasets are independent and thus avoid overfitting. Figure 9 (a) demonstrates the impact of the optimal thresholding quantile ρ on the error rates, indicating that thresholding with $\rho = 0.9$ yields balanced false negatives and false positives, where the FPR for each tunnel is kept at about 9%.

We examine the cases in which classification fails, i.e., false negatives and false positives. A false negative occurs when a monitored site trace is missed, and a false positive occurs when a traffic trace is incorrectly labeled as a monitored site. We find that false negatives in OpenSSH and Shadowsocks are due mainly to some high-volume websites that generate a large number of short sessions, such as 3dmgame and Zhibo8. Since session boundaries are invisible in tunnel traffic, the accumulated session header size distorts the gene fingerprint in tunnels. There are also quite a few failures in Shadowsocks due to retransmitted packets triggered by tunneling delays. Another part of false negatives regardless of the tunnel is resulted from inconsistent website content over multiple visits, such as Sina and Kdslife. For false positives, we find that the incorrectly predicted sites are highly overlapping across tunnels, and these failure cases are due to the fingerprint distance $D(\boldsymbol{b}, \boldsymbol{b}')$ being too close from the unmonitored site to the exemplar. Although the DTW algorithm can handle fingerprint differences introduced by time distortions, it is difficult for GF to distinguish unmonitored sites with similar fingerprints to a particular monitored site.

To directly compare with the other attacks, we implement the same adaptive thresholding with $\rho = 0.9$ in each attack using respective similarity measures, i.e., prediction confidence in CUMUL and DF, and F-distance in DDTW. We first investigate how the error rates can be affected by the number of unmonitored testing websites. As shown in Fig. 9 (b), the FPR tends to increase as the number of unmonitored sites increases for all the WF attacks. DF and DDTW have FPRs at the two extremes, indicating that the packet direction and timing sequences have the worst and best distinguishability in the feature space, respectively. In comparison, GF exploits packet size and timing information and holds a low FPR close to that of DDTW. In Table 4, we report the attack performance in the open world. The results show that GF consistently performs the best on both TPR and precision, with at least 80.9 % TPR and 83.3 % precision for 1,709 unmonitored testing sites. Although GF has a slightly higher FPR than DDTW, it maintains a reasonable maximum of 10.5 % FPR. Overall, our GF attack employing integrated packet timing and size features achieves the highest precision while maintaining a low error rate.

4.5 Time Complexity

Additionally, we examine the time complexity by recording times for feature extraction, training, and testing. Besides our datasets, the experiments also use the publicly available Liberatore et al.'s SSH dataset [27] and Juarez et al.'s Tor dataset [47], which preserve the raw packet features of timing, direction, and size. The Liberatore dataset contains 2000 websites visited for 200 rounds over a two-month period. After removing corrupted traces with less than 20 incoming packets, we keep only the top 100 sites with 200 visits as monitored sites. The Juarez dataset consists of 40

Table 5 Comparison of time complexity. The numbers of sites and traces are indicated below the corresponding dataset. For trials using our datasets, MS-ETH is used for extraction and training, and MS-SSH is used for testing.

Dataset	Attack	Extraction	Training	Testing
MS-ETH MS-SSH (100×10)	GF	257.5s	0.8s	16.0s
	CUMUL	671.8s	0.1s	0.3s
	DF	7.7s	17.8s	2.1s
	DDTW	5.4s	857.2s	4890.4s
Juarez (100×40)	GF	389.3s	3.8s	10.9s
	CUMUL	825.5s	0.5s	0.1s
	DF	9.4s	62.9s	0.9s
	DDTW	6.8s	1165.8s	1264.6s
Liberatore (100×200)	GF	785.2s	10.5s	25.2s
	CUMUL	1213.7s	2.3s	0.5s
	DF	15.4s	307.0s	4.1s
	DDTW	11.3s	1969.4s	2143.8s

traces, collected in ten batches of four visits, for each of the top 100 Alexa sites. We set the sampling interval $\delta = 200$ ms for our GF attack to accommodate the low bandwidth and high latency of the Tor network. Since these datasets are collected from specific tunnels, we cannot utilize cross-tunnel datasets for training and testing. Instead, we use stratified splitting to create non-overlapping training and testing data in a $9:1$ ratio for time complexity trials only.

The corresponding time complexity results for WF attacks are presented in Table 5. The code for each attack runs on an Intel i7-6700 CPU with 32 GB memory. We accelerate the training and testing of the DF attack using an NVIDIA RTX 2060 with 6 GB of GPU memory and accelerate these tasks for other attacks by calling the underlying C libraries through Cython. As can be seen from the table, our GF attack has reasonable overall time complexity. While DF requires the least preparation time (extraction+training), GF has less complexity than CUMUL and DDTW. However, GF takes longer for testing than CUMUL and DF because it needs to compare the samples with each exemplar. Comparing the results on all three datasets, we find that GF adapts well to different scale datasets, indicating that it is scalable for applying to real networks.

5. Discussion

5.1 Countermeasures

The GF attack requires network packet size and timing information to construct distinguishable gene fingerprints. Therefore, preventing the side-channel leakage in website traces is sufficient to counter the attack. We consider the tradeoffs of several countermeasures and how they affect the identification of gene fingerprints.

Padding. Recall that our attack identifies websites across tunnels based on the similarity between gene fingerprints. This similarity can be weakened by padding the tunnel packets. One common padding method is to pad each packet by a random amount, such as the SSH protocol that pads packets to an eight-byte boundary [34]. Since the packet sizes before and after padding are still highly correlated, we can obtain an approximate amount of transmitted data. Another radical way is padding to ensure that all packets are the same size, such as the Tor network that fixes the cell to 512 bytes [45]. This approach pushes the gene fingerprint further away from the elected exemplar and makes classification difficult, although at the cost of increased bandwidth overhead.

Dummy Traffic. While padding aims to hide the fingerprint profile, generating dummy traffic aims to camouflage the fingerprint to look like other websites. With each loading request, the server could send several bursts of dummy packets that mimic a specific target site, similar to the generation of decoy website traffic [29]. In this way, the fingerprints collected by the attacker will be incorrectly labeled as the spoofed site. This approach can defeat the attack but comes at the cost of high bandwidth overhead, a dilemma between strong anonymity and practical usability, and requires some cooperation from the client to ignore the dummy traffic.

Shift Timing. Another way to counter the attack without increasing the amount of data transmitted is to shift the packet timings. Changing the timing of tunnel packets results in varied fingerprint growth trends and pause positions. Since our attack requires consistent fingerprint profiles, shifted timings would affect the DTW alignment and increase the fingerprint distance. This method requires caching packets and dumping them onto the network according to a specific schedule. The drawback of this approach is that the shifted packets could adversely affect usability since website content is delayed to the user.

5.2 Limitations and Future Directions

First, while our GF attack performs well against most websites, it is still challenging to identify those with dynamic content. As demonstrated in the evaluation, some false negatives in the open world are caused by inconsistent website content on different visits, resulting in these websites with multiform fingerprints. However, this does not exclude the possibility of website inference through improved algorithms. A straightforward approach is to map such a website to multiple classes and select the respective exemplars for each class in the training phase. We leave the further investigation of this technique to future work.

Second, although the proposed gene fingerprint integrates packet size and timing information, it remains a challenge to improve the efficiency of measuring the distance between fingerprints. We utilize fixed-period sampling to reduce the computational complexity of the DTW algorithm. However, this approach yields numerous feature points of the same height, making the distance measurement less efficient. One possible direction is to simplify the fingerprint representation by merging duplicate feature points, thus reducing unnecessary comparisons while preserving timing information. We leave the challenge of applying this idea to network traffic traces to potential future work.

Finally, while we achieve accurate identification using

the DTW and k-NN algorithms, this similarity-based approach cannot faithfully capture the fine-grained features of gene fingerprints, which have been illustrated in Fig. 4 with good consistency. To this end, we can adopt well-established neural networks in computer vision to automatically extract more robust local features. Previously, it was shown that the use of deep learning methods in the WF domain has the potential to defeat countermeasures [5]. Future research could fruitfully employ these methods to identify website gene fingerprints.

5.3 Reproducibility of our Results

To ensure scientific correctness and reproducibility of our results, we publish the following at `https://github.com/gene-fingerprinting/gf`:

- The source code for our GF attack. This includes feature extraction, model construction, and performance evaluation.
- Our implementations of known CUMUL [4], DF [5], and DDTW [31] attacks, which we compared and evaluated against ours.
- The datasets we used for evaluation. This includes the list of monitored and unmonitored sites, and the packet traces collected over different tunnels.

6. Conclusion

At present, WF attacks require that the adversary can deal with ever-changing tunneling attributes and assume that he can collect data and train for every unmonitored site. In this paper, we propose gene fingerprinting (GF), a more practical attack that exploits the intrinsic profile of websites and classifies websites based on the similarity between gene fingerprints. Our attack allows training the classifier without unmonitored site samples and enables a trained classifier to work across tunnels.

Although we consider this work as the first attempt to bring Zero-shot Learning to WF attacks, we are also aware of a substantial amount of work to be addressed. For example, packet padding techniques (e.g., Tor) will obfuscate the boundaries between gene fingerprints, leading to degraded attack performance. Moreover, it is still a challenge for WF attacks to identify websites with dynamic content. In the future, we hope to extend this work in the direction of designing more robust feature representations and powerful classifiers.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (grant nos. 2019QY1302 and 2019QY1305).

References

[1] M. Juárez, S. Afroz, G. Acar, C. Díaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," Proc. 2014 ACM SIGSAC Conference on Computer and Communications Security, pp.263–274, ACM, 2014.
[2] V. Rimmer, D. Preuveneers, M. Juárez, T. van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," 25th Annual Network and Distributed System Security Symposium, The Internet Society, 2018.
[3] Y. Zhao, X. Ma, J. Li, S. Yu, and W. Li, "Revisiting website fingerprinting attacks in real-world scenarios: A case study of shadowsocks," Network and System Security - 12th International Conference, pp.319–336, Springer, 2018.
[4] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," 23rd Annual Network and Distributed System Security Symposium, 2016.
[5] P. Sirinam, M. Imani, M. Juárez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security, pp.1928–1943, ACM, 2018.
[6] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," 2017 IEEE International Conference on Intelligence and Security Informatics, ISI 2017, Beijing, China, July 22-24, 2017, pp.43–48, IEEE, 2017.
[7] V.F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," IEEE Trans. Inf. Forensics Security, vol.13, no.1, pp.63–78, 2018.
[8] J. Kampeas, A. Cohen, and O. Gurewitz, "Traffic classification based on zero-length packets," IEEE Trans. Netw. Serv. Manag., vol.15, no.3, pp.1049–1062, 2018.
[9] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," Comput. Networks, vol.165, 2019.
[10] H. Bai, G. Liu, W. Liu, Y. Quan, and S. Huang, "N-gram, semantic-based neural network for mobile malware network traffic detection," Secur. Commun. Networks, vol.2021, pp.5599556:1–5599556:17, 2021.
[11] D.F. Isingizwe, M. Wang, W. Liu, D. Wang, T. Wu, and J. Li, "Analyzing learning-based encrypted malware traffic classification with automl," 21st International Conference on Communication Technology, pp.313–322, IEEE, 2021.
[12] S. Liu, Y. Han, Y. Hu, and Q. Tan, "Fa-net: Attention-based fusion network for malware https traffic classification," IEEE Symposium on Computers and Communications, pp.1–7, IEEE, 2021.
[13] D.X. Song, D.A. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on SSH," 10th USENIX Security Symposium, 2001.
[14] S.E. Oh, S. Li, and N. Hopper, "Fingerprinting keywords in search queries over tor," Proc. Priv. Enhancing Technol., vol.2017, no.4, pp.251–270, 2017.
[15] J.V. Monaco, "What are you searching for? A remote keylogging attack on search engine autocomplete," 28th USENIX Security Symposium, pp.959–976, 2019.
[16] C.V. Wright, L. Ballard, S.E. Coull, F. Monrose, and G.M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," 2008 IEEE Symposium on Security and Privacy, pp.35–49, IEEE Computer Society, 2008.
[17] A.M. White, A.R. Matthews, K.Z. Snow, and F. Monrose, "Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks," 32nd IEEE Symposium on Security and Privacy, pp.3–18, IEEE Computer Society, 2011.
[18] C. Wang, S. Kennedy, H. Li, K. Hudson, G. Atluri, X. Wei, W. Sun, and B. Wang, "Fingerprinting encrypted voice traffic on smart speakers with deep learning," 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp.254–265, ACM, 2020.
[19] A. Morichetta, P. Casas, and M. Mellia, "EXPLAIN-IT: towards explainable AI for unsupervised network traffic analysis," Proc. 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Ar-

tificial Intelligence for Data Communication Networks, pp.22–28, ACM, 2019.
[20] C. Gijón, M. Toril, M.S. Delgado, S. Luna-Ramírez, and L.R. Jimenez, "Encrypted traffic classification based on unsupervised learning in cellular radio access networks," IEEE Access, vol.8, pp.167252–167263, 2020.
[21] R. Hwang, M. Peng, C. Huang, P. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," IEEE Access, vol.8, pp.30387–30399, 2020.
[22] D. Li, Y. Zhu, and W. Lin, "Traffic identification of mobile apps based on variational autoencoder network," 13th International Conference on Computational Intelligence and Security, pp.287–291, IEEE Computer Society, 2017.
[23] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D.J. Dubois, M. Lindorfer, D.R. Choffnes, M. van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," 27th Annual Network and Distributed System Security Symposium, The Internet Society, 2020.
[24] L. Yang, A. Finamore, F. Jun, and D. Rossi, "Deep learning and zero-day traffic classification: Lessons learned from a commercial-grade dataset," IEEE Trans. Netw. Serv. Manag., vol.18, no.4, pp.4103–4118, 2021.
[25] P. Wang, Z. Wang, F. Ye, and X. Chen, "Bytesgan: A semi-supervised generative adversarial network for encrypted traffic classification in SDN edge gateway," Comput. Networks, vol.200, p.108535, 2021.
[26] A. Hintz, "Fingerprinting websites using traffic analysis," Privacy Enhancing Technologies, pp.171–178, Springer, 2002.
[27] M. Liberatore and B.N. Levine, "Inferring the source of encrypted HTTP connections," Proc. 13th ACM Conference on Computer and Communications Security, pp.255–263, ACM, 2006.
[28] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," Proc. first ACM Cloud Computing Security Workshop, pp.31–42, ACM, 2009.
[29] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," Proc. 10th annual ACM workshop on Privacy in the electronic society, pp.103–114, ACM, 2011.
[30] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," Proc. 23rd USENIX Security Symposium, pp.143–157, 2014.
[31] S. Feghhi and D.J. Leith, "A web traffic analysis attack using only timing information," IEEE Trans. Inf. Forensics Security, vol.11, no.8, pp.1747–1759, 2016.
[32] P. Sirinam, N. Mathews, M.S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," Proc. 2019 ACM SIGSAC Conference on Computer and Communications Security, pp.1131–1148, ACM, 2019.
[33] H.F. Alan and J. Kaur, "Client diversity factor in HTTPS webpage fingerprinting," Proc. 9th ACM Conference on Data and Application Security and Privacy, pp.279–290, ACM, 2019.
[34] T. Ylonen, "The Secure Shell (SSH) Transport Layer Protocol," RFC 4253, IETF, Jan. 2006.
[35] S.D.T, "Shadowsocks: A secure SOCKS5 proxy," Project of Shadowsocks, https://shadowsocks.org/assets/whitepaper.pdf, accessed 27 May 2021.
[36] C.H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp.951–958, IEEE Computer Society, 2009.
[37] Y. Fu, T. Xiang, Y. Jiang, X. Xue, L. Sigal, and S. Gong, "Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content," IEEE Signal Process. Mag., vol.35, no.1, pp.112–125, 2018.
[38] T. Ylonen, "The Secure Shell (SSH) Connection Protocol," RFC 4254, IETF, Jan. 2006.
[39] L. Zheng, L. Zhang, and D. Xu, "Characteristics of network delay and delay jitter and its effect on voice over IP (VoIP)," IEEE International Conference on Communications, pp.122–126, IEEE, 2001.
[40] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," IEEE Trans. Acoust., Speech, Signal Process., vol.26, no.1, pp.43–49, 1978.
[41] "OpenSSH," https://www.openssh.com/, accessed 23 April 2021.
[42] "Shadowsocks," https://shadowsocks.org/, accessed 23 April 2021.
[43] "OpenVPN," https://openvpn.net/, accessed 23 April 2021.
[44] "Site Rank," http://www.alexa.cn/siterank, accessed 15 June 2021.
[45] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," Proc. 12th annual ACM Workshop on Privacy in the Electronic Society, pp.201–212, ACM, 2013.
[46] E.J. Keogh and M.J. Pazzani, "Derivative dynamic time warping," Proc. First SIAM International Conference on Data Mining, pp.1–11, SIAM, 2001.
[47] M. Juárez, M. Imani, M. Perry, C. Díaz, and M. Wright, "Toward an efficient website fingerprinting defense," 21st European Symposium on Research in Computer Security, Lecture Notes in Computer Science, vol.9878, pp.27–46, Springer, 2016.

Ding Li received the B.S. and M.S. degrees in Information Engineering University, China, in 2015 and 2018. He is currently pursuing the Ph.D. degree at State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University. His research topics include network security, traffic analysis and machine learning.

Chunxiang Gu received his Ph.D. degree in State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University, China, in 2006. Currently he is a professor at State Key Laboratory of Mathematical Engineering and Advanced Computing. His research interests mainly include cryptography and network security.

Yuefei Zhu received his Ph.D. degree in Zhengzhou Information Science Technology Institute, China, in 1990. Currently he is a professor at State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University, China. He is the main designer of the Elliptic Curve Cryptography (ECC) Public-Key Algorithm SM2. His research topics include cryptography and network security.