Luo *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2021) 10:29
https://doi.org/10.1186/s13677-021-00244-8

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

RBP: a website fingerprinting obfuscation method against intelligent fingerprinting attacks

Tao Luo*, LiangMin Wang, ShangNan Yin, Hao Shentu and Hui Zhao

Abstract

Edge computing has developed rapidly in recent years due to its advantages of low bandwidth overhead and low delay, but it also brings challenges in data security and privacy. Website fingerprinting (WF) is a passive traffic analysis attack that threatens website privacy which poses a great threat to user's privacy and web security. It collects network packets generated while a user accesses website, and then uses a series of techniques to discover patterns of network packets to infer the type of website user accesses. Many anonymous networks such as Tor can meet the need of hide identity from users in network activities, but they are also threatened by WF attacks. In this paper, we propose a website fingerprinting obfuscation method against intelligent fingerprinting attacks, called Random Bidirectional Padding (RBP). It is a novel website fingerprinting defense technology based on time sampling and random bidirectional packets padding, which can covert the real packets distribution to destroy the Inter-Arrival Time (IAT) features in the traffic sequence and increase the difference between the datasets with random bidirectional virtual packets padding. We evaluate the defense against state-of-the-art website fingerprinting attacks in real scenarios, and show its effectiveness.

Keywords: Traffic obfuscation, Website fingerprinting defense, Traffic analysis, Packet padding, Edge computing

Introduction

With the expansion of the Mobile Internet, a large number of mobile smart devices have been connected to the network which has brought unprecedented pressure to the backbone network. The centralized processing model with cloud computing will not be able to efficiently process the data generated by edge devices. To solve this problem, researchers proposed edge computing, which is a new computing model that performs computing at the edge of the network [1]. Edge computing can process a large amount of temporary data at the edge of the network and reduce the interaction with the cloud computing center, which greatly reduces bandwidth overhead and system delay.

*Correspondence: taoluo.ujs@gmail.com
School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China

However, the users are vulnerable to traffic analysis attack who located in the edge computing architecture. As illustrates in the Fig. 1, an attacker can obtain users' identity through following four steps to conduct a traffic analysis attack:

- Step1: The attacker collects traffic traces from the core cloud to conduct a train set.
- Step2: He trains the machine learning-based classifier with the collected training traces with their corresponding labels.
- Step3: The attacker eavesdrops on the mobile users to capture unlabeled trace.
- Step4: In the prediction phase, the attacker uses the trained classifier to predict users' identity.

Traffic analysis attack poses a great threat to user's privacy, and so more and more users wish to hide their identity in network activities. Tor [2], as a famous anonymity

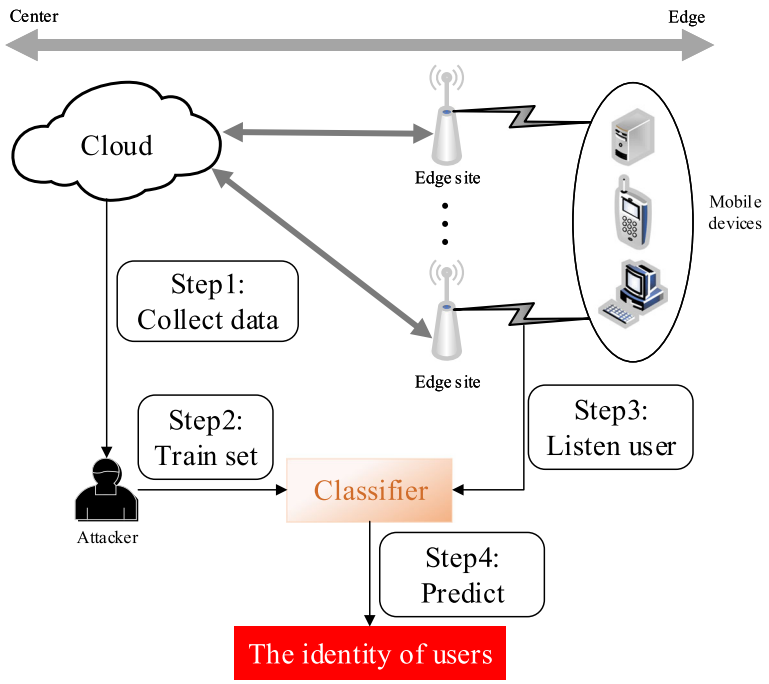


Fig. 1 The traffic analysis attack in cloud/edge computing architecture

network, can provide privacy preserving by defending web-browsing users from network eavesdroppers. To do so, it forwards user packets across multiple proxies, so that network surveillance cannot see both the true source and destination of the packets.

However, multiple studies have shown that Tor is also vulnerable to *Website Fingerprinting* (WF), a kind of traffic analysis attack. WF attacks can passively listen to the network traffic between the client and the first hop node of the Tor network, and then uses machine learning algorithm to identify or predict the website accessed by the client. On the one hand, the attacker collects encrypted packets transmitted between the client and the server, extracts traffic patterns and features, and then uses machine learning algorithms to perform traffic analysis to predict the target website accessed by the user. On the other hand, defenders (such as Tor) have been developing various defenses to thwart various attacks by disguising and morphing packets.

To thwart various WF attacks, a number of WF defenses have been proposed over the years. Recently, Marc Juarez et al. [3] proposed a lightweight defense, WTF-PAD. WTF-PAD has been successfully deployed as a Pluggable Transport (PT) in Tor network due to its advantages of effectiveness against ML-based WF attacks, zero latency and low bandwidth overhead. However, the latest WF attack proved the ineffectiveness of WTF-PAD. Deep Fingerprinting (DF) [4] and Triplet Fingerprinting (TF) [5] achieved 91.9% and 86.6% accuracy for WTF-PAD in the closed-world scenario. To the best of our knowledge, none of defenses are effective against *Deep Learning* (DL)-based WF attacks or adopted by Tor. This is because their bandwidth overhead may be too high; they may delay packets too much, hurting user experience; they may be too hard to implement realistically; or they may simply be ineffective against the best attacks. Therefore, a defense against the WF problem grows increasingly urgent as more powerful attacks are found.

In this paper, we propose a novel WF defense based on time sampling and random bidirectional packet padding. The definition of packet padding is inject packets into traffic sequence to extend the trace and conceal the traffic features. The proposed defense algorithm obfuscates website traffic patterns through the use of random time sampling and random bidirectional padding of dummy packets (dummy packets are general web traffic packets). The dummy packets are not only sent from the client, but also sent by the server, that is bidirectional padding. Through randomization and bidirectional packet padding, the features of the original traffic sequence are completely disguised. The main contributions of this paper are summarized as follows:

- We propose a novel WF defense, called Random Bidirectional Padding (RBP). The proposed defense leverages the technologies of direct time sampling and dummy packets padding to obfuscates the distinctive features in traffic sequence. Specifically,

RBP considers the Inter-Arrival Time (IAT) feature of packets in opposite directions.
- We prove the ineffectiveness of the deep learning-based classifiers under the different packet distributions, because RBP can expand the difference between packet distributions. The accuracy of the state-of-the-art WF attacks such as DF [4] and TF [5] is reduce from 98% (no defense) to 28% and 92% (no defense) to 38%.
- We show that it is difficult for an attacker to identify traffic depend on a single traffic sequence against our defense. We also implement and evaluate our approach against a Tor dataset and show its performance on bandwidth overhead and latency delay in practical scenario.

The rest of the paper is organized as follows. In "Background" section, we present relevant background information and related studies about website fingerprinting. In "Our defense methodology" section, we present and discuss our defense methodology. The model is evaluated in "Experimental evaluation and discussion" section. Finally, we conclude our paper in "Conclusions" section.

Background

Data encryption of network packets provides users with privacy by hiding plain text content when transmitting data between two devices in the network, such as TLS [6], HTTPS [7] and other network transmission protocols. However, these protocols cannot hide the identity of users, and the source IP and the destination IP of this transaction can be easily obtained by using the traffic analysis attack. Generally, we can use a proxy server to hide the real IP address, that is the client builds an indirect connection with the server through this proxy service. At this time, the network traffic seems to come from the proxy server instead of the client. Therefore, the combination of encryption and proxy server can better hide the user's identity information. In addition, the multiple proxy methods provided by anonymous networks can better hide identity, such as Tor can hide user's information by providing low-latency anonymity and randomization of transmission channels.

The main problem of WF attacks is to identify the website browsed by a client through encrypted and anonymized network connections that exploit meta information of encrypted packets transmitted between the user and an anonymity network. Figure 1 illustrates an attacker achieves the purpose of WF attack by listening the network activities of the first hop node and the client of the Tor network. In this paper, we use the term "website" and "webpage" interchangeably

Network packets have attributes such as IP address, packet size, and transmission time. Machine learning or deep learning classifiers can exploit these attributes and output the probability of user accesses a website. These attributes can be used by the classifier called website fingerprinting. The attacker first captures the network packets exchanged between the client and the server to form a traffic dataset, and then extracts the attributes of the packets from the traffic dataset and expresses them as a feature vector which is used to train the classifier.

WF attack

As shown in Fig. 2, we assume that the client accesses the website through the Tor network, and the attacker is a local listener located between the client and the Tor network entry guard. We also assume that the attacker already knows the identity of the client, and the purpose of the attack is to detect the website accessed by the client. Moreover, the attacker is passive, which means that the attacker can only observe and record traffic information and cannot drop, delay, modify packets or inject new packets. Due to the layered encryption of Tor network, we also assume that the attacker cannot learn anything about packet payload.

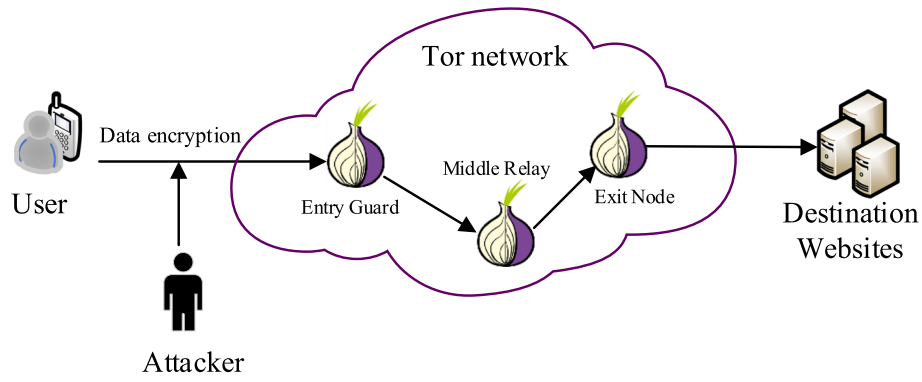


Fig. 2 A Tor anonymity network example showing a user connecting to the Internet via three Tor nodes. The WF passive attack occurs between the user and the Tor entry guard

Numerous studies [4, 5, 8–13] have proposed techniques to perform website fingerprinting attacks. Most of them are first extract features such as packet length, packet direction and time are collected from traffic flow at the user's end, and then uses these features to train machine learning or deep learning models. Then the classifier can predict the class of each website the user accesses based on the traffic generated by the user. The quality of the features determines the quality of the classifier, so a distinctive feature set for training is required. In the literature [3], besides using packet length histograms, the authors defined a burst as a sequence of packets that has been sent in a short time period. Conversely, a gap is a sequence of packets that are spread over a long-time span. To clarify the notation adopted in this paper, we use outgoing which is a positive sign to refer to the direction from the client to the web server, and conversely, incoming is the direction from the web server to the client, and expressed with the negative sign. DF [4] achieved unprecedented accuracy only using the direction of the packets. The accuracy for no-defensed traffic is as high as 98.3%, which is much better than other WF attacks such as k-NN [8], CUMUL [9] and AWF [10] (e.g. k-NN: 83.85%, CUMUL: 91.02%, AWF: 94.9%), DF can still reach accuracy of more than 90% against the traffic that has been defensed by WTF-PAD. This research shows that the features of the direction of the packets is a distinctive feature for the attacker.

Wang et al. [8]. proposed a WF attack using the k-NN classifier, which includes features such as packet size, packet direction, and burst number. They use k-NN classification with weighted L1 distance and a page is classified as belonging to particular class only if all k neighbors belong to this class.

Panchenko et al. [9]. implemented a WF attack based on an Support Vector Machine (SVM) classifier with a Radial Basis Function (RBF) kernel which is called CUMUL, and devised a novel feature set based on the cumulative sum of packet lengths constructed as follows: the first coordinate in the feature vector is the length of the first packet in the traffic trace and the i^{th} coordinate is the sum of the value in the $i-1^{th}$ coordinate plus the length of the i^{th} packet, where lengths for incoming packets are negative.

Rimmer et al. [10]. proposed to apply deep learning for automated feature extraction in WF attacks. This attack is called Automated Website Fingerprinting (AWF) which uses a CNN-based classifier to conduct WF attack. The results show that the adversary can use deep learning to automate the feature engineering process to effectively create WF classifiers. Thus, it can eliminate the need for feature design and selection.

Sirinam et al. [4]. proposed a deep fingerprinting (DF) attack based on convolutional neural networks (CNN) which uses only the direction of the packets, and through a large amount of data for training, it reaches a maximum accuracy of about 98% in a closed-world scenario. DF also analyzed many WF defenses, such as WTF-PAD [3], which has been deployed in Tor network, and achieved an accuracy of more than 90%.

Sirinam et al. [5]. proposed a WF attack is called Triplet Fingerprinting (TF) based on the triplet network in N-shot learning (NSL), which can identify webpage class with a few training samples. In large-scale WF attack, TF can significantly reduce the training overhead of deep learning models and still achieve a high accuracy for different distributions of datasets.

Closed-world and Open-world Scenario. The closed-world scenario assumes that users can only access a finite set of websites as shown in Fig. 3(a), and these websites are under the listen of the attacker, and the attacker can analyze the traffic trace generated by users access the website. But closed-world scenario is unrealistic, there are a huge number of websites that users can access, and attackers

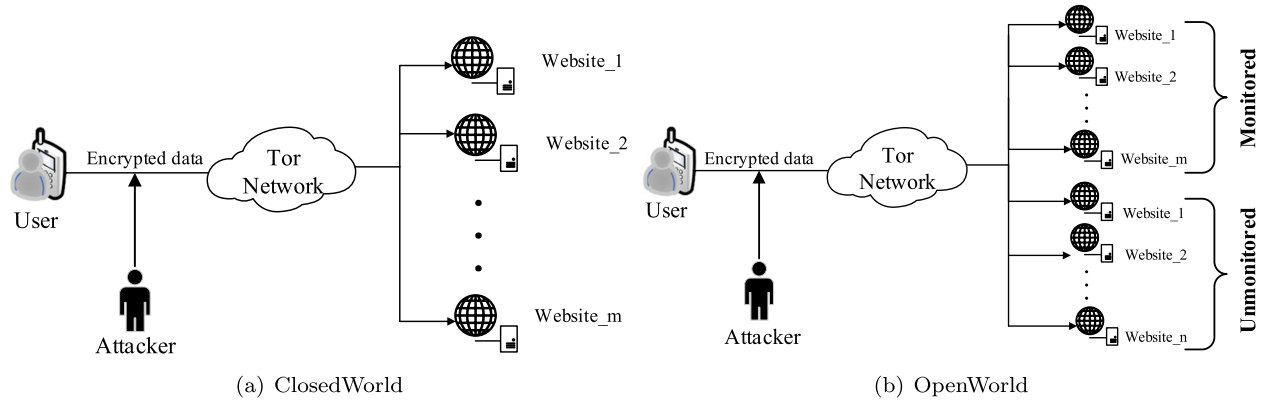


Fig. 3 In the Closed World scenario (a), the user visits one page among those monitored by the attacker. In Open World (b), the user is allowed to visit unmonitored pages

simply cannot monitor all websites. Therefore, the open-world scenario is a more practical attack scenario. In the open-world scenario, an attacker can only collect a portion of traffic generated by users accessing the website in Fig. 3(b). The techniques developed to address the open-world problem design a binary classifier that requires traces from both the finite monitored set and an infinitely large (rest of the universe) set of non-monitored websites. As shown in Fig. 3, we assume that the number of websites monitored by the attacker is m, while the number of websites not monitored is n, and the value of n is much greater than m.

WF defense

Website fingerprinting attacks are mainly designed to train machine learning models by extracting useful features from encrypted packets. The more prominent and unique the features, the better the effectiveness of WF attacks. Therefore, most of the defense against attack are designed with the aim of obfuscating the patterns of the encrypted packets of the loaded website. WF defense has been an active area of research, and many defenses have been introduced in literature [3, 14–20]. These defenses include traffic morphing (make the source website distribution appear to come from another website distribution) [17] and time deforming (the time required for packets exchange between the client and the server). The following introduces several common WF defenses.

BuFLO. Dyer et al. [14]. proposed Buffered Fixed-Length Obfuscation (BuFLO), which combines packet padding and time deforming methods. BuFLO sends fixed-length packets at fixed intervals, and uses dummy packets to pad and extend the transmission. Packet padding refers to hide the distribution of website traffic by increasing the packet length. One of the basic and effective padding defenses is Pad-to-MTU, which means that each individual packet is padding to the maximum transmission unit (MTU) in the TCP connection. The defense method using packet padding will make all the packets in the data transmission process the same size, so it is difficult for an attacker to obtain a useful pattern from the traffic dataset to conduct WF attack. The packet padding method is bound to increase bandwidth overhead, so the practicality of the method may not be ideal.

In order to overcome the bandwidth overhead burden caused by packet padding, literature [17] proposed a Direct Target Sampling (DTS) and Traffic Morphing (TM) are distribution-based padding defenses that use statistical sampling techniques. DTS morphs the source packet size into the packet size of the target traffic through random sampling (select normal website traffic as target traffic), which makes the source webpage as similar to the target page as possible.

TAMARAW. Cai et al. [16]. improved BuFLO by introducing a lighter defense named TAMARAW. TAMARAW sends fixed-length packets at fixed intervals as same as BuFLO, but the fixed-length packets are 750 bytes instead of the maximum transmission unit. During the padding process, TAMARAW will set a padding parameter, if the total number of packets sent in both directions in and out of the packets are multiple of this parameter, TAMARAW stops padding. In addition, the amount of padding generated depends on the total size of the webpage. Due to the asymmetry of Web browsing traffic, Cai et al. suggested using different packet sizes and padding at different rates to handle incoming and outgoing traffic independently.

WTF-PAD. Juarez et al. [3]. proposed an adaptive padding method for WF defense by improving the adaptive padding (AP) algorithm [4], which is called Website Traffic Fingerprinting Protection with Adaptive Defense (WTF-PAD). This defense uses the packet time sampling approaches to send dummy packets in the gaps of real packets without delaying actual traffic and achieve smaller bandwidth overhead than the BuFLO family.

BiMorphing. Al-Naami et al. [19]. considered the dependence of consecutive packet sequences in the opposite direction, and then proposed a novel WF defense method with bidirectional padding. BiMorphing uses a matrix to calculate packet distribution and IAT distribution to morph the bidirectional burst mode. It also uses mathematical optimization techniques to minimize the bandwidth overhead.

Our defense methodology

In this section, we introduce the model of WF defense and demonstrate our defense methodology.

Defense model

In addition to features such as packet size commonly used in WF attacks, Inter-Arrival Time (IAT) is also a significant feature of web traffic. WTF-PAD [3] uses a link padding method, which uses the packet time sampling approaches to pad dummy packets in the gaps of real packets, and does not cause the actual traffic delay. BiMorphing [19] analyzed the importance of the feature of burst in WF attacks and proposed the concept of bi-burst. BiMorphing also uses the principle of AP algorithm [21] to pad the gaps in the bursts, which obfuscates the feature of burst in website distribution.

In order to split the IAT into burst and gap model, WTF-PAD calculates the instantaneous bandwidth at the time of each inter-arrival time to determine if it is part of a burst or not. The padding phase selects whether to send a dummy packet and determines the waiting time according to the current mode (burst or gap mode). In burst mode, the algorithm essentially assumes there is a burst of real data and consequently waits for a longer

period before sending any padding, while in gap mode, the algorithm assumes there is a gap between bursts and consequently aims to add a fake burst of padding with short delays between packets. However, there is no gap mode in BiMorphing, only the burst mode, which has been redefined: consecutive packets in the same direction are called a burst, and two opposite bursts are called a bi-burst. Therefore, BiMorphing uses dummy packets to pad the internal gap of the burst, and the direction of the dummy packet is consistent with the burst direction.

Figure 4 shows the padding example of WTF-PAD and BiMorphing. WTF-PAD will determine whether it is currently in gap mode or burst mode according to the instantaneous bandwidth of the inter-arrival time. If in the gap mode, it will be padded dummy packets. Generally, burst mode does not padding dummy packets. BiMorphing will first calculate the matrix distribution of the original traffic and the IAT in the target traffic traces. After detecting the arrival of the real traffic packet, it calculates the interval between the packets in the burst according to the matrix distribution sampling and padding a dummy packet in the interval of traffic trace as same as the direction of the burst.

However, due to the fixity of the padding mechanism of WTF-PAD and BiMorphing, the traffic flow formed by each padding is similar, and the process of morphing feature produces new features. For example, if a user accesses the same webpage at different time, the two website distributions generated by the user are similar under the condition that there are no major changes to the webpage. DF [4] uses an improved CNN model, which can extract the key features from the traffic traces protected by WTF-PAD (or BiMorphing) and then to train the classifier with these features, DF achieved more than 90% test accuracy.

Based on the limitations of WTF-PAD and BiMorphing in traffic processing and padding mechanism, we propose a WF obfuscation method against WF attacks based on machine learning or deep learning, dummy packets are randomly padding in bi-direction between the interval of the original traffic flow in our algorithm. Our defense can extend the difference between two packet distributions generated by a user accesses a website at difference time and also disguise the distinctive features such as packet size and packet direction.

Defense design

In order to defeat the WF attacks, it is not adequate to morph the packet sequences by using size padding techniques or even more sophisticated time delay methods, because of some existing deep learning models can identify these obfuscated traffic traces with high accuracy. Inter-Arrival Time (IAT) in the traffic sequence is a distinguishing feature that can predict the class of webpages in high accuracy. Even if defense was applied, attackers can still achieve their aim by using this feature alone to train models. In this section, we introduce a novel defense against WF attacks by random bidirectional packet padding. This defense can completely disrupt the website distributions generated by the user accesses the website, and make sure zero delay so that it will not

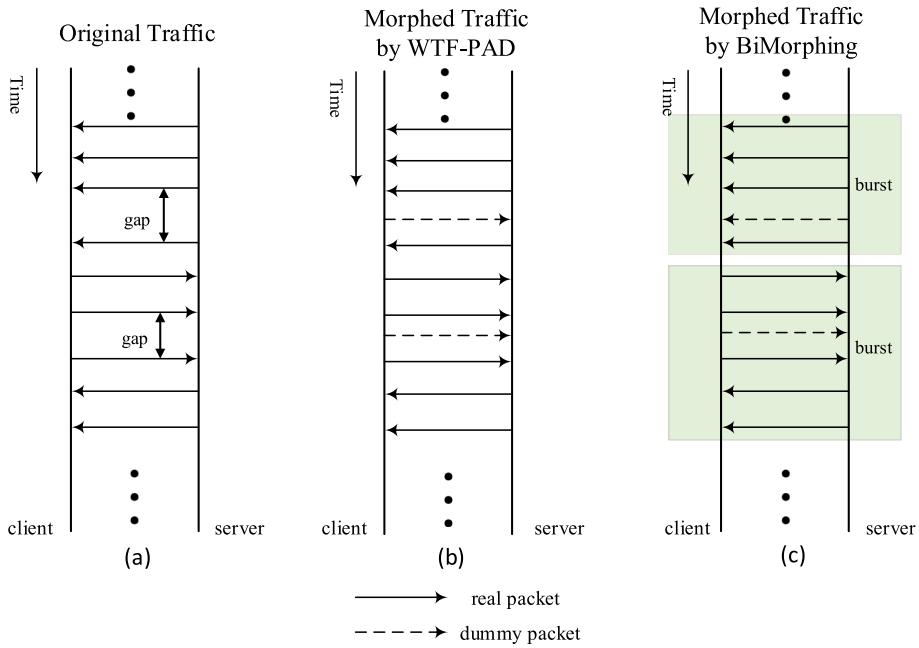


Fig. 4 The example of WTF-PAD and BiMorphing

cause a large impact on the actual user experience in web browsing.

As shown in Fig. 5(a) is the original traffic sequences generated by user accesses website while Fig. 5(b) and (c) are the traffic sequences generated by RBP against original traffic in different time. Our defense makes the traffic sequence after padding is extremely different from the original traffic sequence, which completely disrupts the packet distribution, and the direction of the dummy packet and the insert position is random. Thus, our defense can increase the attack cost of the attacker. For example, we assume that the traffic sequence generated by the user use RBP to access a certain webpage in time A is Fig. 5(b) is called $traffic_A$, which is captured by the attacker, and extract features to train a model. The traffic sequence in Fig. 5(c) is generated by the user use RBP to access the same webpage in time B is called $traffic_B$, and there are no major changes to the webpage (time A and B present a user accesses a same webpage at different time). The attacker cannot use the trained classifier to better identify $traffic_B$. If the attacker wants to identify the $traffic_B$, he needs to capture the $traffic_B$ again, extract the features, and then train the classifier, which greatly increases the attack cost of the attacker. However, even though an attacker uses $traffic_A$ or $traffic_B$ to conduct WF attack with 10-fold cross validation, the results of attack are undesirable for the attacker. For example, we perform a 10-fold cross validation with a single dataset against TF [5], the TF only reach 28.8% accuracy in closed-world scenario.

Our defense can completely disrupt the traffic distributions generated by a user accesses the website, and make sure zero delay and moderate bandwidth overhead. RBP algorithm is depicted in Fig. 6 using a finite state machine. Starting from INI Mode, it will switch between Sampling Mode and Padding Mode until receive a real packet or the time expires.

INI Mode. Starting in INI Mode, our defense waits for receive a real packet and calculates a traffic distribution X^T from target websites. **Sample Mode.** When a real packet is received, the sate machine switch to the Sample Mode. RBP algorithm sample a time t from the distribution X^T and start a timer. When the time t expires, this means no packet received for a consecutive time, the state of mode switch to the Padding Mode, if the trace is end, the state will return to INI Mode. RBP algorithm match the traffic sequence with target traffic without any delay, ensure the good user experience in low latency network.

Padding Mode. In Padding Mode, the algorithm generates a dummy packet which is similar to normal website packet except for packet size, and then inject it in the traffic sequence. After padding, the algorithm returns to Sample Mode, restart a new timer.

Figure 7 depict the workflow of RBP, which consists of an *initialization* phase and a padding phase. The *initialization* phase is responsible for building distribution that

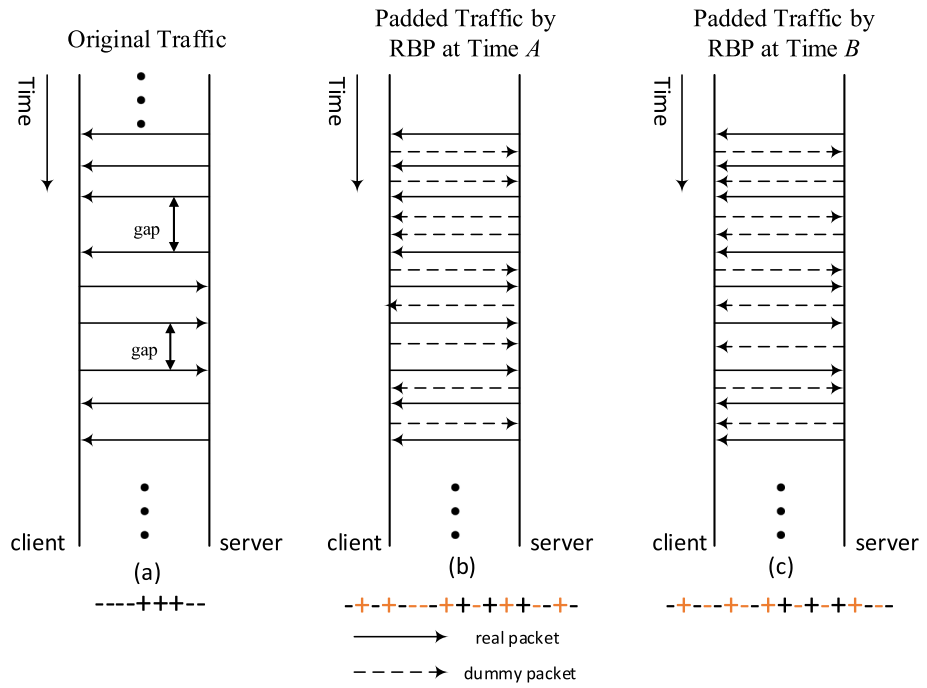


Fig. 5 RBP Example. Figure(a) present the original traffic generated by user access website, Figure(b) shows the process of RBP of the original traffic in time A while Figure(c) present the process in time B. Time A and B mean a user accesses a website at different time

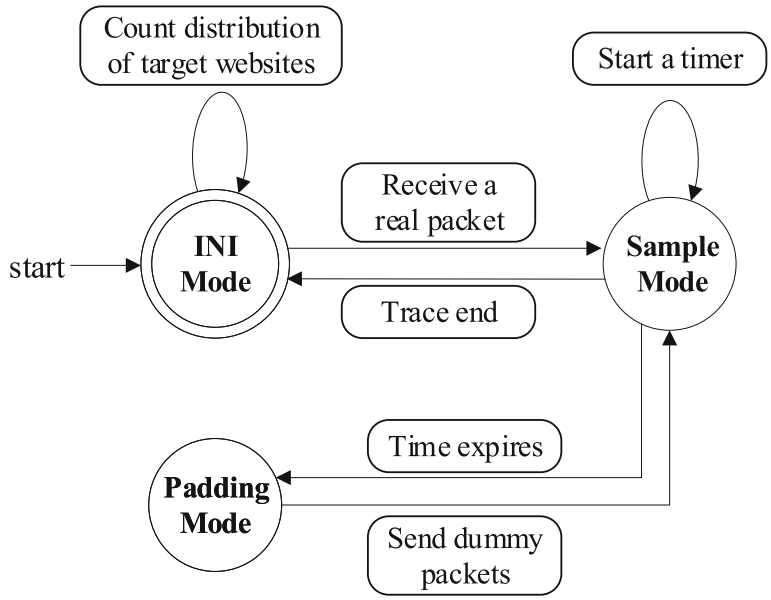


Fig. 6 Finite state machine to illustrate the RBP algorithm

will be used in the *padding* phase. The architecture will be explained in detail in the following sections.

Initialization

RBP consists of three main components, time sampling, Inter-Arrival Time (IAT) sampling, and random bidirectional padding. We now explain the three components in detail.

Time Sampling. First we define some notations that we use in our figures and throughout the paper. Let T be the traffic sequence generated by the target websites, and given a sample space of n possible IAT. We describe the target traffic sequence as a column vector representing a probability mass function over the n sizes $X^T = [x_1, x_2, ..., x_n]$, where x_i is the probability of the i^{th} largest IAT at the target traffic sequence. To sample the time t will be used in the padding phase, the algorithm first sums the probabilities into a cumulative distribution function

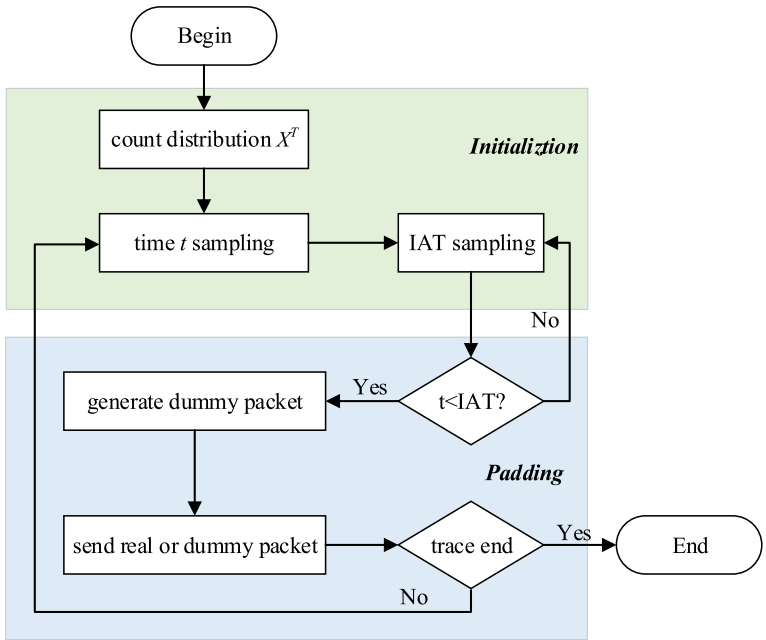


Fig. 7 RBP workflow

such that the cumulative probability of target IAT s_i is equal to the sum of the probability for all IAT sizes $\leq s_i$. Then, the algorithm runs a pseudorandom number generator to get a random number $r \in [0, 1]$, and selects first target IAT with a cumulative probability $\geq r$. For example, if $cdf(j) \geq r$, the algorithm selects the j^{th} largest IAT as the time t.

Inter-Arrival Time (IAT) Sampling. IAT is a distinctive feature, and differences in IAT will directly affect the distribution of packets. Although only the feature of the packet direction is used in DF attacks, the sampling of IAT is crucial for our *padding* phase. To reduce the delay and bandwidth overhead caused by the defense mechanism as much as possible, we need to pad the gaps of the traffic sequence with dummy packets, which can minimize the delay and avoid affecting the transmission of real packets. When the first real packet is received, we calculate the arrival time of the next packet in the same direction as the packet. The interval time is the IAT obtained by padding in this phase. Compare the time t obtained by direct time sampling and IAT, if $t < IAT$, it means that new real packet will arrive before the time t expires, the dummy packet is not padded, or padding the dummy packet at the end of the time t if $t > IAT$, and restart the procedure of time sampling and IAT sampling.

Random Bidirectional Padding

In our defense, the *padding* phase apply randomization to the packet padding without any delay and moderate bandwidth overhead. The algorithm of padding as follows.

Algorithm 1: Random Bidirectional Padding

Input: Original traffic $T_{original}$, target traffic T_{target}
Output: The traffic sequence padded by RBP

```
// Define the size of padding packet ;
length ← Definition()
while trace(T_original) do
    t ← Sampleing(T_target) ;
    if t expires then
        flow ← flow_direction(T_original) ;
        // Generate dummy packet ;
        dummy ← generate_dummy(flow, t, length) ;
        // Inject the dummy packet ;
        insert(dummy) ;
    end
end
```

Padding. At the beginning of *padding* phase, the algorithm sets the length of packets is MTU (Maximum Transmission Unit) that means the length of all dummy packets padded by RBP is MTU. Then, we use the time t obtained in the *initialization* phase to determine the position of traffic trace for padding dummy packets. If client does not receive real packet when the time t expire, RBP algorithm will inject the dummy packet to trace. The generated dummy packets are determined by timestamp, the real packet direction and length. For example, the timestamp of the dummy packet equal to the current traffic timestamp add the time t. The direction of dummy packet is determined by the two previous packet directions. If the first two packets are in the same direction, the direction of the dummy packet is opposite, if the first two packet directions are opposite, the direction of the dummy packet is consistent with the previous packet.

Experimental evaluation and discussion

In this section, we demonstrate the effectiveness of the proposed WF defense. We evaluate our defense against a Tor dataset. We examine the closed-world and open-world scenarios when no defense is applied and when there is a defense mechanism.

Dataset

The dataset we use contains the traffic data in the closed-world and the open-world described in §2.1. We use the Tor dataset to evaluate the effectiveness and overhead of our defense in a practical scenario. The dataset is composed of encrypted packets generated by the browser accesses the Tor network. The dataset is described in detail in [8]. In addition, we also use the dataset in [22] in closed-world scenario as an experimental comparison to further test the performance of our defense under different datasets.

As described in Table 1, the dataset consists of two groups of collections. The first one is a group of the monitored website, there are 100 websites with 90 traces each. We use this group of data for the closed-world experiments. The second collection consists of 9000 websites where each website has on trace. These websites were selected from the Amazon Alexa's top websites [23]. In the open-world setting, we consider the first group of 100 websites as the monitored set and the second group of 9000 websites as the unmonitored set. The dataset in literature [22] is also selected from the Amazon Alexa's top 100 websites which consists of 100 websites with 40 traces each.

Experimental setup

We use the dataset of the top 100 monitored websites in the Tor dataset [8] and the dataset in the literature [22]

Table 1 The Tor dataset

Dataset		# of websites	# of traces per websites
Tor [5]	Monitor	100	90
	Unmonitor	9000	1

to evaluate the performance of RBP in the closed-world scenario. We use these datasets to train and test the classifier and take the average accuracy for assessment. For evaluating RBP in the open-world scenario, we use the whole Tor dataset. The monitored set consists of the 9000 instances of the 100 blocked websites in the first collection while the unmonitored set consists of the second collection websites (i.e. 9000 websites with one instance each). The classification becomes a binary classification problem with each monitored website as the class 1, and each unmonitored website as the class 0.

We choose two defenses, WTF-PAD [3] and TAMARAW [16], as competitors to our defense representing two extremes in design philosophy: WTF-PAD is a lightweight obfuscation defense, while TAMARAW is a heavyweight defense with high latency and bandwidth overhead. Other defenses have been broken by known attacks or more expensive than TAMARAW, or impractical to implement.

We use three state-of-the-art WF attacks: CUMUL [9], TF [5] and DF [4] explained in §2 to evaluate our defense. CUMUL train a support vector machine (SVM) classifier by using the cumulative feature set, achieve a 92.9% accuracy against no defense dataset. TF uses a triplet network as feature extractor and use a k-NN classifier to train the feature vector generated by feature extractor, TF solve the challenge of the difference between datasets in WF attack filed and achieve over 90% accuracy. DF uses an improved neural network classifier based on CNN and attains over 98% accuracy on Tor traffic without defenses which is the highest accuracy compare with all prior attacks.

Evaluation metrics

We use several metrics to evaluate our defense, such as Accuracy in multi-classification problem, F1 score, TPR and FPR in the binary classification problem and overhead etc.

Accuracy. In the closed-world scenario, the accuracy is defined as the proportion of target (i.e. the target class of attack) classifications that are correct. If the WF attacker classifies a trace as belonging to a target webpage, it is a accurate. In an experiment, let N_{ALL} and N_{ACC} denote the number of all class and correct classifications. Then the accuracy is:

$$Accuracy = \frac{N_{ACC}}{N_{ALL}}$$

F1 score and Recall. Furthermore, as the open-world scenario is a binary classification problem, we measure the true positive rate (TPR) and false positive rate (FPR). These are defined as follows: $TPR = \frac{TP}{TP+FN}$) and $FPR = \frac{FP}{FP+TN}$. Here, TP (True Positive) is the number of traces which are monitored, and predicted as monitored by the classifier. FP (False Positive) is the number of traces which are unmonitored, but predicted as monitored. TN (True Negative) is the number of traces which are unmonitored and predicted as unmonitored. FN (False Negative) is the number of traces which are monitored, but predicted as unmonitored. In addition, we measure the F1 score and Recall. These are defined as:

$$F1 = \frac{2TP}{2TP + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

Trace. A trace is a sequence of packets collected during a page loading process, denoted as $Trace = (t_1, Packet_1), (t_2, Packet_2), .., (t_{|T|}, Packet_{|T|})$ where $|T|$ is the total number of cells in the trace. t_i is the timestamp of the i^{th} packet. $Packet_i$ show the direction and length of the i^{th} packet.

Bandwidth Overhead. Let $Trace$ denote the original trace and $Trace'$ denote the trace after implementing some defenses. The bandwidth overhead BO is the total amount of dummy data divided by the total amount of real data:

$$BO = \frac{|Trace'| - |Trace|}{|Trace|}$$

Delay Overhead. The delay overhead DO of Defense on $Trace$ is the extra time taken to transmit real packets, divided by the original transmission time. Denote the last real packet in $Trace'$ as t_k, then we have:

$$DO = \frac{t_k - t_{|T|}}{t_{|T|}}$$

Generally, delay overhead affects users' browsing experience while bandwidth overhead shows the extra burden laid on the network. They should be considered together when evaluating a defense. We define these two metrics to be independent of each other, to simplify the analysis and to more easily highlight how defenses change each overhead. When bandwidth is a concern, for example, increasing the bandwidth overhead will likely delay page loading but will not change the time overhead. Our defense have zero delay overhead and moderate bandwidth overhead.

Results

Using the Tor dataset, we evaluate the RBP approach in the closed-world and open-world settings. We show the results when no morphing is applied (normal traffic) and compare them to the morphed data (when packets are morphed).

RBP in closed-world. Table 2 presents the closed-world results using the original and defended data. As shown in the table, after classifying the 100 websites the accuracy of the data when no defense is applied is pretty high, TF attack attains 92.2% accuracy, while DF model achieve the highest 98.3% accuracy. When defenses are applied to traffic, the accuracy drops.

Table 2 Accuracy (%) of known attacks in the closed-world setting against normal and morphed data

Defense	Attack Accuracy			Average Accuracy
	CUMUL	TF	DF	
No Defense	92.9	92.2	98.3	94.5
TAMARAW	4.36	5.82	8.8	6.33
WTF-PAD	87.4	86.6	91.9	88.6
RBP	**21.4**	**28.8**	**38.6**	**29.6**

Table 3 Bandwidth and delay overhead of various WF defenses

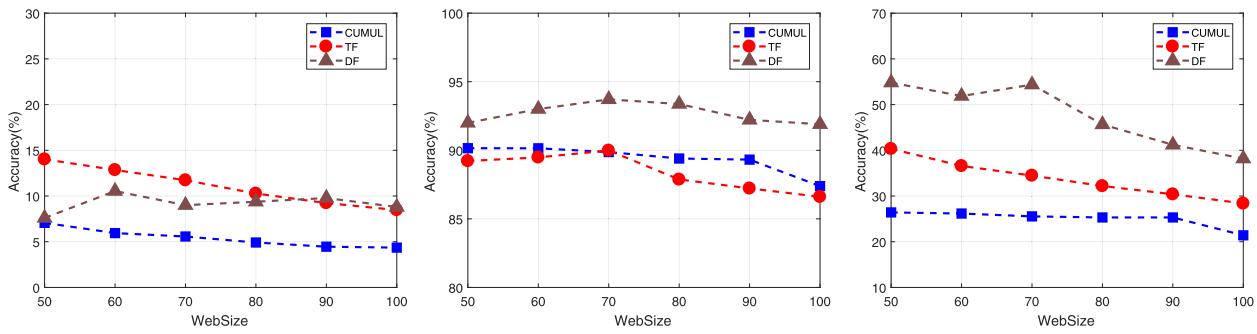
Defense	BW Overhead (%)	Delay Overhead
No Defense	100	No
TAMARAW	543	Yes
WTF-PAD	177	No
RBP	**270**	**No**

It can be seen that for all three CUMUL, TF and DF attacks, RBP achieves less accuracy than WTF-PAD [3]. The lower the accuracy, the more effective the defense is. Especially against DF, our defense achieves remarkable results that directly reduced its accuracy from 98.3% to 38.6%. This shows the effectiveness of the proposed RBP defense which considers a zero delay optimized random padding technique. Not only does RBP disguise the bidirectional packet patterns via the bidirectional time sampling and random padding, but it also protects against the inter-packet arrival time leak through the IAT sampling technique. From Table 3, it can be seen that TAMARAW [16] performs better than RBP regarding accuracy. However, we later explain why TAMARAW is not a practical defense strategy to apply in website fingerprinting.

We also analyze the impact of the number of monitored websites in the Tor dataset on accuracy. As shown in Fig. 8, the performance of the three defenses technologies improves as the number of websites increases, that is, the accuracy of WF attacks gradually decreases. Figure 8(a) shows that TAMARAW has the best defense performance. Neither machine learning nor deep learning classifiers can well identify the traffic protected by TAMARAW. The highest accuracy rate is about 14%, which is only containing 50 websites. Figure 8(b) and (c) show the performance of WTF-PAD and RBP defenses against three WF attacks in the closed-world setting. The overall performance is in accordance with Table 2. The defense performance of RBP is between TAMARAW and WTF-PAD, but far better than WTF-PAD, and RBP achieves a moderate bandwidth overhead. In general, our defense is an effective WF defense in the closed-world scenario.

RBP in open-world. The results of the open-world scenario are illustrated in Table 4, we show the results when no defense is considered as well as when applying the defenses techniques. In an open-world, the effectiveness of a defense must decrease the TPR of WF attacks while increasing its FPR as much as possible. It can be seen from Table 4, although our defense cannot significantly improve the FPR of WF attacks, the TPR reduces from 94.7% (no defensed) to 25.8% (padded) against DF, while WTF-PAD only reduces TPR to 70.4%, which shows that most of the traffic defended by WTF-PAD are identified. For the TF attack which also uses deep learning technology, our defense achieves a lower TPR than DF, and the accuracy and F1 score are further reduced. The reason is that TF uses the triplet network in deep learning to extract features, but TF also use the k-NN classifier in traffic classification. Although the training and testing time of TF is significantly reduced compared with DF, the accuracy of traffic identify is lower than DF.

CUMUL is an effective WF attack which calculates the cumulative features based on the traffic sequence. Thus, the machine learning such as k-NN or SVM classifier use the cumulative features of CUMUL can still achieve a good classification result. It can be seen from Table 3 that the TPR value decreases from 96.6% (no defense) to 57.2%



(a) TAMARAW (b) WTF-PAD (c) RBP

Fig. 8 Closed-world: The impact of the amount of monitored websites on accuracy

Table 4 Defense performances in an open-world scenario on Tor dataset

Defense	CUMUL			TF			DF		
	TPR	FPR	F1	TPR	FPR	F1	TPR	FPR	F1
No Defense	96.6	6.48	96.5	85.7	0.43	92.1	94.7	0.55	96.7
WTF-PAD	78.1	12.6	81.7	82.9	0.51	90.4	70.4	1.44	81.2
TAMARAW	30.6	18.4	31.3	6.19	0.88	11.6	0.6	0.2	1.19
RBP	**57.2**	**11.3**	**67.9**	**22.1**	**0.78**	**36.1**	**25.8**	**0.79**	**40.9**

for RBP and to 78.1% for the WTF-PAD defense. Along with the TPR and FPR ratios, the tables also show the F1 score as well as accuracy value in open-world setting.

From the Table 4 shows that CUMUL achieve a higher TPR than TF and DF which is inconsistent with the literature [3]. We believe that deep learning classifier requires a lot of data for the traffic of no defense, so the TPR of DF is lower than CUMUL. Regarding the defended traffic, the Tor dataset [5] has not reached the scale of websites in the literature [3], DF cannot train the CNN model well. On the other hand, due to the cumulative feature of CUMUL, it may be offset by the accumulation operation after padding dummy packets of the same size. For example, two dummy packets are the same size and in opposite direction are padded, the result of adding these two packets is zero. After the features processing of CUMUL, the final result is that these two dummy packets have not been padded, thus this operation decreases the performance of our defense.

Non-practical Defense. TAMARAW based padding model comes with substantial bandwidth overhead and a reduction in protocol obfuscation, although results in lower accuracy for most of the attacks. The cause of this is the greater amount of padding after the transmission has finished in TAMARAW compared to other defense techniques. For instance, in the closed-world setting, the experiments in Table 2 show that under the TAMARAW defense, CUMUL, TF and DF attacks achieve only 4.36%, 5.82% and 8.8% accuracies respectively. However, these experiments reveal that TAMARAW comes with an enormous bandwidth overhead cost, which is roughly more than 500% as shown in Table 3. On the other hand, RBP and WTF-PAD achieve 270% and 177% bandwidth overhead respectively, which is insignificant compared to the bandwidth overhead of TAMARAW. This leads to the conclusion that defenses like TAMARAW are not practical approaches to deploy as WF defenses in Tor compared to other defenses that achieve much lower bandwidth overheads.

Defense Overhead. When we apply WF defense technology to pad or change packets and confuse the adversary, it creates some inevitable overhead, namely bandwidth overhead and time overhead. In the literature [19], the bandwidth overhead of a defense is defined as the number of extra packets added in the morphed data, divided by the number of packets in the original packet sequence. The time overhead of a defense is defined as the extra time needed to load the packet sequence in the morphed data, divided by the original time required in the original packet sequence. We follow Al-Naami et al's consideration. As an effective WF defense algorithm must minimize these overheads while ensuring the defense performance. In reality, unlike bandwidth overhead, any delay overhead becomes a concern in low-latency networks like Tor. As discussed in §3, RBP achieves a zero delay algorithm that sends the extra sampled dummy packets in gaps of real packets in a way that ensures real packets arrive on time.

In this section, we show the bandwidth (BW) and delay overhead. We see from Table 3 that RBP achieves a lower bandwidth overhead than TAMARAW while WTF-PAD achieve the lowest bandwidth overhead. For the delay overhead, as shown in Table 3, RBP and WTF-PAD scores a zero delay overhead to the actual traffic whereas TAMARAW cannot avoid it. From Table 3, we can see WTF-APD achieve the minimal overhead. However, due to the application of deep learning in the field of WF attack, WTF-PAD cannot resist the deep fingerprinting attacks based on neural networks, such as DF [4]. Although our defense increases bandwidth overhead, it does not impose heavy burden on users. At the same time, RBP protects users' privacy in network activities without any delay.

Evaluation of Bandwidth Overhead. In this part, we want to measure how an increase in the overhead budget affects the attacker's effectiveness. We show TPR of three WF attacks in Fig. 9. Without RBP, DF, CUMUL and TF can achieve 96.6%, 94.7% and 85.7% TPR respectively. Its TPR decreases quickly as we initially increase the size overhead. With about 260% bandwidth overhead, its TPR is already lower than 40%.

Similarity Measurement. We extensively analyze the difference of the two datasets by using a similarity measurement. The similarity measurement is used to evaluate how similar of two network traffic's vectors in the latent space. We apply the Cosine distance to calculate the distance between a pair of network traffic; the smaller distance represents the lower similarity of the given pair of network traffic. The range of Cosine distance is [0,1]. We calculated the Cosine distance between the two datasets

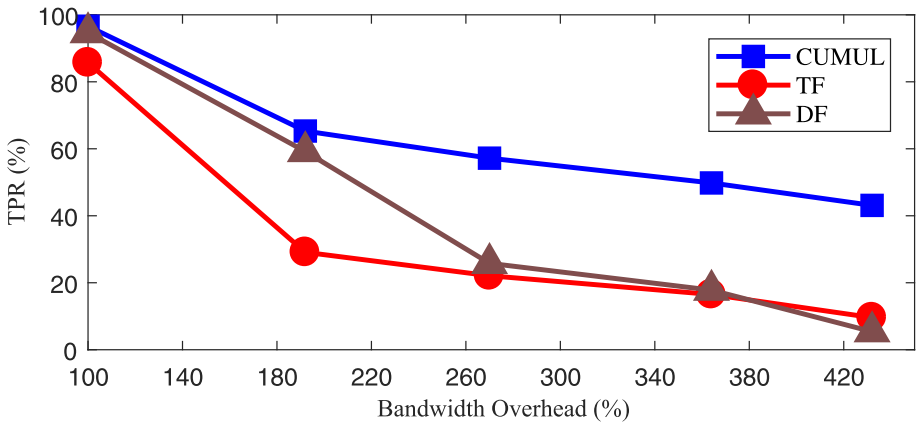


Fig. 9 Three WF attacks' TPR on protected traces given different bandwidth overhead budgets

protected by WTF-PAD and RBP in difference time. We also consider the closed-world and open-world settings.

As shown in Table 5, the Cosine distance is calculated for all traffic examples from the WTF-PAD dataset and the original dataset is 0.961 (closed-world) and 1.0 (open-world). As we discussed in §3, due to the fixity of the packets padding algorithm, WTF-PAD cannot completely disguise the distinctive features in traffic sequence. For deep learning classifiers, the WTF-PAD dataset is almost the same as the original dataset. This is one of the reasons why DF can achieve such a high accuracy against WTF-PAD. On the contrary, the Cosine distance of RBP datasets is only 0.014 in closed-world setting and 0.053 in open-world setting, while the distance of original dataset and RBP dataset is 0.069 and 0.107. The results show that our defense makes the traffic sequence generated at different times completely different and obfuscates the most of features.

Conclusions

In this paper, we proposed a novel WF defense that can defeat WF attacks based on deep learning models which combines direct time sampling and random bidirectional padding, which ensures moderate bandwidth overhead and incurs zero delay for real packets exchanged between client and server. Our defense can conceal the inter-arrival time feature in the traffic sequence and extend the difference of distribution of traffic generated when the user is accessing the same website in different time. Thus, an adversary uses the deep learning model which trained on the previous dataset cannot identify the traffic well, making the attack cost of the adversary increase. We proved the effectiveness of the proposed approach empirically by examining the defense against passive attacks and comparing it with state-of-the-art methods. The promising results, moderate bandwidth overhead, and real packets zero latency give a new perspective for a more practical WF defense.

Table 5 The Cosine distance between the datasets

Cosine distance	WTF-PAD		RBP	
	Original	WTF-PAD	Oirginal	RBP
ClosedWorld	0.961	0.999	0.069	0.014
OpenWorld	1.0	1.0	0.107	0.053

Acknowledgments
The research work was supported by the National Natural Science Foundation of China under Grant U1736216. In addition, the authors would like to acknowledge the National Natural Science Foundation of China under 61702230.

Authors' contributions
All authors have participated in conception and design, or analysis and interpretation of the data, drafting the article or revising it critically for important intellectual content, approval of the final version. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
The data and materials are uploaded to the github, this contain the source code and datasets which the conclusions of the manuscript rely. The data and materials can be found at https://github.com/lt1342258812/RBP.git

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 17 August 2020 Accepted: 6 May 2021
Published online: 20 May 2021

References

1. Shi W, Sun H, Cao J, Zhang Q, Liu W (2017) Edge computing-an emerging computing model for the internet of everything era. J Comput Res Dev 54(5):907–924
2. Dingledine R, Mathewson N, Syverson P (2004) Tor: The second-generation onion router. In: 13th Usenix Security Symposium. Usenix

3. Juarez M, Imani M, Perry M, Diaz C, Wright M (2016) Toward an efficient website fingerprinting defense. In: European Symposium on Research in Computer Security. Springer, Switzerland. pp 27–46
4. Sirinam P, Imani M, Juarez M, Wright M (2018) Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp 1928–1943
5. Sirinam P, Mathews N, Rahman MS, Wright M (2019) 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. pp 1131–1148
6. Rescorla E, Modadugu N, et al. (2006) Datagram transport layer security. RFC 4347, April
7. Rescorla E, Schiffman A (1999) The secure hypertext transfer protocol. IETF Request for Comments, RFC 2660
8. Wang T, Cai X, Nithyanand R, Johnson R, Goldberg I (2014) Effective attacks and provable defenses for website fingerprinting. In: 23rd {USENIX} Security Symposium ({USENIX} Security 14). pp 143–157
9. Panchenko A, Lanze F, Pennekamp J, Engel T, Zinnen A, Henze M, Wehrle K (2016) Website fingerprinting at internet scale. In: NDSS. pp 1–15
10. Rimmer V, Preuveneers D, Juarez M, Van Goethem T, Joosen W (2018) Automated website fingerprinting through deep learning. In: Proceedings of the 25nd Network and Distributed System Security Symposium
11. Abe K, Goto S (2016) Fingerprinting attack on Tor anonymity using deep learning. Proceedings of the Asia-Pacific Advanced Network 42:15–20
12. Hayes J, Danezis G (2016) k-fingerprinting: A robust scalable website fingerprinting technique. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). pp 1187–1203
13. Wang L, Sheng VS (2020) Multilevel identification and classification analysis of Tor on mobile and PC platforms. IEEE Trans Ind Inform 17:1079–1088
14. Dyer KP, Coull SE, Ristenpart T, Shrimpton T (2012) Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In: 2012 IEEE Symposium on Security and Privacy. IEEE, San Francisco. pp 332–346
15. Cai X, Nithyanand R, Johnson R (2014) Cs-buflo: A congestion sensitive website fingerprinting defense. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society. pp 121–130
16. Cai X, Nithyanand R, Wang T, Johnson R, Goldberg I (2014) A systematic approach to developing and evaluating website fingerprinting defenses. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp 227–238
17. Wright CV, Coull SE, Monrose F (2009) Traffic morphing: An efficient defense against statistical traffic analysis. In: NDSS. Citeseer, San Diego Vol. 9. pp 35–49
18. Wang T, Goldberg I (2017) Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In: 26th {USENIX} Security Symposium ({USENIX} Security 17). pp 1375–1390
19. Al-Naami K, El Ghamry A, Islam MS, Khan L, Thuraisingham BM, Hamlen KW, Alrahmawy M, Rashad M (2019) Bimorphing: A bi-directional bursting defense against website fingerprinting attacks. IEEE Trans Dependable Secure Comput 18:505–517
20. Luo X, Zhou P, Chan EW, Lee W, Chang RK, Perdisci R (2011) Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows. In: NDSS Vol. 11
21. Shmatikov V, Wang M-H (2006) Timing analysis in low-latency mix networks: Attacks and defenses. In: European Symposium on Research in Computer Security. Springer, Berlin Heidelberg. pp 18–33
22. Juarez M, Afroz S, Acar G, Diaz C, Greenstadt R (2014) A critical evaluation of website fingerprinting attacks. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp 263–274
23. The top visited sites on the web. https://www.alexa.com/. Accessed 07 July 2020

Publisher's Note

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com