Hindawi
Security and Communication Networks
Volume 2022, Article ID 2962318, 11 pages
https://doi.org/10.1155/2022/2962318

Research Article

Defending against Deep-Learning-Based Flow Correlation Attacks with Adversarial Examples

Ziwei Zhang and Dengpan Ye

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China

Correspondence should be addressed to Dengpan Ye; yedp@whu.edu.cn

Received 23 September 2021; Revised 8 December 2021; Accepted 23 February 2022; Published 27 March 2022

Academic Editor: Mamoun Alazab

Tor is vulnerable to flow correlation attacks, adversaries who can observe the traffic metadata (e.g., packet timing, size, etc.) between client to entry relay and exit relay to the server will deanonymize users by calculating the degree of association. A recent study has shown that deep-learning-based approach called DeepCorr provides a high flow correlation accuracy of over 96%. The escalating threat of this attack requires timely and effective countermeasures. In this paper, we propose a novel defense mechanism that injects dummy packets into flow traces by precomputing adversarial examples, successfully breaks the flow pattern that CNNs model has learned, and achieves a high protection success rate of over 97%. Moreover, our defense only requires 20% bandwidth overhead, which outperforms the state-of-the-art defense. We further consider implementing our defense in the real world. We find that, unlike traditional scenarios, the traffic flows are "fixed" only when they are coming, which means we must know the next packet's feature. In addition, the websites are not immutable, and the characteristics of the transmitted packets will change irregularly and lead to the inefficiency of adversarial samples. To solve these problems, we design a system to adapt our defense in the real world and further reduce bandwidth overhead.

1. Introduction

Tor is the most popular and low-latency anonymity network that provides anonymous communication services for more than two million people [1]. It includes over 3000 relays that transmit massive encrypt packets and conceal client's information. Every relay only knows its previous and latter relay's address.

But flow correlation attacks break this security model. Network-level adversaries, i.e., autonomous systems (ASes) have the power to observe traffic characteristics between client to entry relay and exit relay to the destination server. They can link these data (in particular packet timings and packet sizes) to deanonymize users, as shown in Figure 1. The correlation algorithm used in the beginning studies is usually a traditional method like Pearson correlation or Cosine similarity. Recent research leverages a deep learning model to correlate traffic characteristics with significantly higher accuracies than existing algorithms.

Existing defense methods to detect or mitigate traffic analysis attacks mainly focus on obfuscating encrypt packets, traffic morphing, changing network-level characteristics that does not affect the deep-learning-based attack. And to our best knowledge, existing defenses are all designed to mitigate traffic analysis attacks like website fingerprint attacks or BGP hijack attacks. There is no effective defense faced to flow correlation attack.

Against this strong deep-learning-based attack, the adversarial example is a natural choice for us to confuse CNNs model. So, we explore how effective the adversarial examples defend flow correlation attacks and how to implement defense in the real world.

First, we reconstruct the targeted model that represents state-of-the-art attack and gets the similar accuracy that Milad Nasr et al. [10] mentioned. Second, we evaluate various adversarial example methods' effects including FGSM, C&W, Deepfool, and BIM. The experimental results show that the success rate of applying adversarial examples

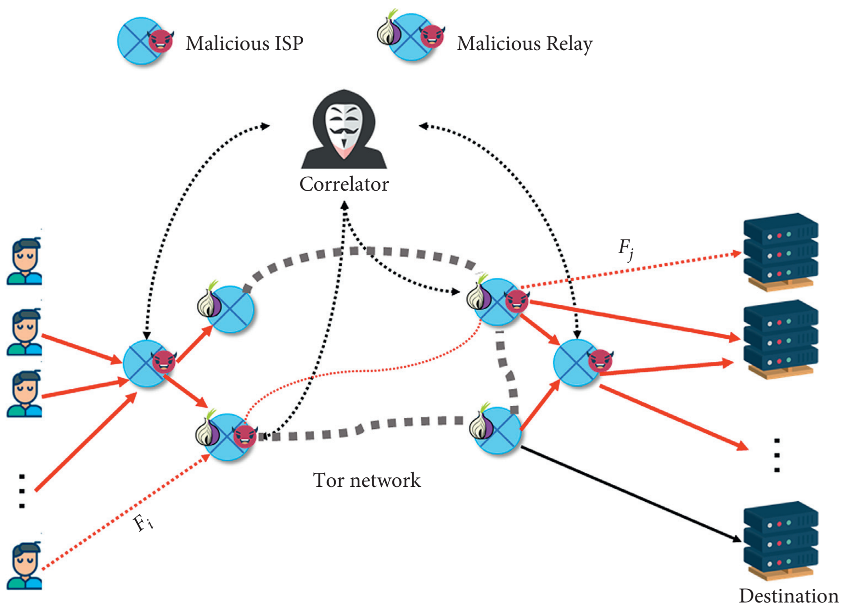

Figure 1: The main process of flow correlation attack on Tor. The adversary intercepts Tor flows either by running malicious Tor relays or eavesdropping on Internet ASes and IXPs.

to defeat the flow correlation model is more than 97% with only 20% bandwidth overhead.

Third, we try to implement our defense in the real world, but we find that there are some challenges we have to face. (1) The websites are not immutable, so the characteristics of the transmitted packets are not immutable. (2) The traffic flows are "fixed" only when they are coming, which means we must know the next packet's feature. (3) The dummy packets we add will go through entire circuit (client- > entry relay- > middle relay- > exit relay- > server). This has increased bandwidth overhead. How can we reduce these extra dummy packets after they have done their job?

To solve the first and second problem, we design a center server that termly collects traffic characteristics of websites and generates corresponding adversarial examples. To solve third problem, we design a mechanism to drop redundant dummy packets at the entry relay, which further reduces bandwidth overhead.

The key contributions of this work are as follows:

(1) We propose a novel defense mechanism against deep-learning-based flow correlation attacks that inject dummy packets into flow traces by pre-computing adversarial examples.

(2) We further evaluate various adversarial example methods' effects, and the experimental results show that even the worst method (FGSM) we used also gain a protection success rate of over 90% with an acceptable bandwidth overhead (30%).

(3) We analyze the challenges of applying our defense in the real world and design a system to solve these challenges, including center server, full-duplex mode, and drop dummy packets mechanism.

The rest of the paper is organized as follows: Section 2 introduces related work, including the development of traffic analysis and adversarial examples. Section 3 describes our proposed method in detail. Section 4 shows the details and results of our experiment. In Section 5, we point out our limitations and give future directions. In Section 6, we conclude our work.

2. Related Work

2.1. Flow Correlation Attack and Defense. Flow correlation attack was a type of traffic analysis attack, and the traffic analysis attack was a type of side-channel attack. Side-channel attacks always leveraged non-normal ways to infer sensitive information from well-protected systems, such as by observing traces (e.g., timing, power, or resource usage). Diao et al. [2] launched inference attacks without any permission in Android by interrupting timing analysis and applying it to interrupt logs. Liu et al. [3] presented a side-channel attack to infer user inputs on keyboards by exploiting sensors in the smartwatch. Schuster et al. [4] aimed to identify video information by using the deep-learning model to classify encrypted video streams.

Flow correlation attacks as a significant side-channel attack was applied in many fields. Shmatikov et al. [5] investigated an active attack called watermark attacks. They modified the packet flows to "fingerprint" them and analyze the tradeoffs between the amount of cover traffic, extra latency, etc. In addition, they also proposed a defense method by using adaptive padding. The work of Paxson and Zhang [6] made the traffic packets as a series of ON and OFF patterns and used these data to correlate network flows. Murdoch and Zieliński [8] developed and evaluated Bayesian traffic analysis techniques to process sampled data. Blum et al. [7] correlated the aggregate sizes of network packets over time. Sun et al. [9] further combined the asymmetric traffic analysis and BGP hijacking to deanonymize users.

All the above papers used the static metric standard statistical correlation metrics to correlate the vectors of flow timings and sizes. And to gain a higher accuracy, they need to observe the associated flow for five minutes or more. The time it take was too long to correlate lots of short-lived connections. Nasr et al. [10] were the first one to use CNNs models to learn a flow correlation function and achieve drastically higher accuracies.

There was still a big gap in the defense of flow correlation attacks, and Sun et al. [11] proposed a defense method that mainly solved the BGP hijacking and reduced the chance of adversary observed network traffic. The obfs4 [12] as a Tor official defense could randomly obfuscate packets time and size but get a poor protection success rate with an unacceptable bandwidth overhead. The ScrambleSuit [43] was a thin protocol layer above TCP whose obfuscated the transported application data by using morphing techniques and a secret exchanged out-of-band. It also had impact on defending the flow correlation attacks but has the same problem as obfs4.

There were some ways to improve classification model's ability of defending noisy labels. Liu et al. [47] proved that any surrogate loss function could be used for classification with noisy labels by using importance reweighting. Yu et al. [45] considered the influence of noisy labels in transfer learning and proposed a novel denoising conditional invariant component (DCIC) framework. Xia et al. [46] presented granular-ball sampling that reduced the data size, improved the data quality in noisy label, and get the same classification accuracy on the original data sets. Noise filtering is an effective method of dealing with label noise, but most of them aimed at binary classification. Xia et al. [44] presented a novel label noise filtering learning method for multiclass classification. These methods mainly focus on the scenario of noisy labels that could help adversary improve their correlation model's robustness and our methods aimed at defending against flow correlation attacks by using adversarial examples. Numan et al. [48] carried out a systematic review of clone detection techniques in static WSNs and provided a comprehensive survey of the existing centralized and distributed schemes with their drawbacks and challenges. Guo et al. [49] proposed a deep graph neural network-based Spammer detection (DeG-Spam) model to gain a better effect than baselines that could be a superior choice to correlate with flow data.

2.2. Website Fingerprint Attack and Defense. The scenario and challenge for website fingerprint attacks are very similar to our work. Adversaries get sensitive information about websites such as domain or page content by analyzing network characteristics. It used to be realized by the traditional machine learning method, but now the deep learning method is gradually emerging.

Nowadays more and more studies have been proposed to defeat website fingerprint attacks. Some research focused on the application layer [13–15], defenders changed the routing algorithm or confused HTTP requests to make adversary touch real traffic as little as possible. Application-layer defense strategies were often difficult to implement because the premise of their implementation was very harsh, such as target websites only had HTTP protocols. And these methods could not defend deep-learning-based attacks (less than 60% protection success rate). Other researches focused on the network layer. They aimed to fool the classification model by inserting dummy packets. In the earlier studies [16–18], they used constant rate padding to reduce information leakage caused by time intervals and traffic volume. However, these methods always require high bandwidth overhead of 150%. A recent study [19] found that inserting packages between two packets with a large time gap would reduce the bandwidth overhead. They were also useless when applied in defending deep-learning-based attacks (only achieve 9% and 28% protection success rate). Finally, there was a super sequence defense method called Walkie-talkie [20], which committed to finding a longer package trace that contains subsequences of different website traces. But it only gets a 50% protection success rate against DNN attacks. In general, no method can maintain a high success rate with a small amount of bandwidth overhead. All related works are present in Table 1.

2.3. Adversarial Examples. Adversarial examples are a series of methods to fool machine learning models, such as deep neural networks. They add perturbations to the clean input, forward it to the classifie,r and get an unexpected result. How to generate adversarial perturbations becomes a hot topic in computer vision, natural language processing, etc. There were many prior works that had shown first-order gradient-based attacks to be fairly effective in fooling DNN-based models in both image [21–27], audio [28–30], and text [31–33] domains. The idea of such adversarial attacks was to find a good trajectory that maximally changed the value of the model's output and pushed the sample towards a low-density region. However, to our best knowledge, there is no paper to apply adversarial examples in defending deep-learning-based flow correlation attacks.

3. Method

In this section, we introduce the target model and the specific details of defending against deep-learning-based flow correlation attacks with adversarial examples. Next, we will show our system that were designed to implement defense in the real world.

3.1. Target Model. We reconstruct the idea of Milad Nasr et al. to perform traffic correlation attacks. They use a convolutional neural network (CNN) model to learn a correlation function for Tor's noisy network. It is composed of two convolutional layers and three fully connected layers. The input is a flow pair called $F_{i,j}$, which represents two bidirectional network flows i and j. The specific of $F_{i,j}$ can be described as follows:

$$F_{i,j} = \left[T_i^u; T_j^u; T_i^d; T_j^d; S_i^u; S_j^u; S_i^d; S_j^d\right], \tag{1}$$

Table 1: The related work of flow correlation attack, flow correlation defense, and website fingerprint defense.

Scheme	Method	Innovation points	Authors	Drawbacks
Flow correlation attack	Watermark attacks	Modified the packet flows to "fingerprint" them.	Shmatikov et al. [5]	Require high privileges and break the original communication easily.
	Timing based	Use the traffic patterns to correlate flows.	Paxson and Zhang [6]	Low accuracy.
	Bayesian traffic analysis	Developed Bayesian traffic analysis techniques to process sampled data.	Murdoch and Zieliński [8]	Cannot correlate lots of short-lived connections.
	Fine-grained level detection	Correlated the aggregate sizes of network packets over time.	Blum et al. [7]	Low accuracy.
	Asymmetric traffic analysis	Further combined the asymmetric traffic analysis and BGP hijacking to deanonymize users	Sun et al. [9]	Only useful for BGP hijacking.
	Deep learning based	Use CNNs models to learn a flow correlation function and achieve drastically higher accuracies.	Nasr et al. [10]	Require hardware support.
Flow correlation defense	Counter-RAPTOR	Reduced the chance of adversary observed network traffic.	Sun et al. [11]	Only useful for defending BGP hijacking.
	Obfs4	Randomly obfuscate packets time and size.	Tor project. [12]	Unacceptable bandwidth overhead.
	ScrambleSuit	Use morphing techniques.	Winter et al. [43]	Unacceptable bandwidth overhead.
Website fingerprint defense	Application layer defense	Changed the routing algorithm or confused HTTP requests.	Wladimir et al. [13] Giovanni et al. [14] Henri et al. [15]	Hard to implement in real world.
	Network layer defense	Fool the classification model by inserting dummy packets.	Juarez et al. [19] Wang et al. [20]	Cannot defend the deep-learning-based attack.

where T is the vector of interpacket delays, S is the vector of packet size, and u and d stand for "upstream" and "downstream," respectively (e.g., S_i^u represents the upstream packet size of i).

The model hyperparameters we choose are consistent with Milad Nasr, which are presented in Table 2. To take a first look at the performance, we train our model using data set that publishes with the paper [10]. It includes 50,000 pairs of associated flow pairs and $50,000 \times 24,999 \approx 1.24 \times 10^9$ pairs of nonassociated flow pairs. And we gain a similar performance as described in the paper.

DeepCorr is able to achieve such high accuracy using only 300 packets of each flow. It tells us that we must take action to prevent AS/ISP level adversaries from compromising the anonymity and privacy of Tor users. In the next chapter, we will introduce the defense effect of the adversarial sample against flow correlation attack model and the system we designed to make the defense method applicable to the real world.

Table 2: The model hyperparameters of target model.

Layer	Details
Convolution layer 1	Kernel num: 2000 Kernel size: (2, 30) Stride: (2, 1) Activation: Relu
Max pool 1	Window size: (1, 5) Stride: (1, 1)
Convolution layer 2	Kernel num: 1000 Kernel size: (2; 10) Stride: (4, 1) Activation: Relu
Max pool 2	Window size: (1, 5) Stride: (1, 1)
Fully connected 1	Size: 3000, activation: Relu
Fully connected 2	Size: 800, activation: Relu
Fully connected 3	Size: 100, activation: Relu

3.2. Adversarial Samples against Flow Correlation Attack. Due to the popularity of artificial intelligence and deep learning, adversarial samples have appeared in various scenarios and practical applications. But in many cases, adversarial samples are usually used as a means of attack to escape detection models. In our experiments, adversarial samples are used as a means of defense to fight adversaries who eavesdropping or analyzing users' traffic. Therefore, our defense strategies focus on improving the protection success rate, every small increase will have a huge impact on the adversaries. Because traffic flows are very large, the adversary who eavesdropping traffic will take a lot of manual analysis time if the attack success rate cannot reach 95%, which almost means that this method of attack is no longer meaningful. This is the first difference between applying adversarial samples in defending flow correlation attacks and other traditional fields. In addition, every clean image or text is "fixed" before adding perturbation. However, traffic flows will be "fixed" only when it's coming. That means we must know the next packet's feature and add corresponding adversarial perturbation. This is the second difference between applying adversarial sample in defending flow correlation attacks and other traditional fields.

To generate adversarial example, we use different methods: FGSM [34], C&W [36], Deepfool [37], and BIM [35]. The reason for choosing these four methods is to get a more comprehensive evaluation including gradient-based methods and optimization-based methods.

The fast gradient sign method (FGSM) was proposed by Goodfellow et al. in 2015. This algorithm performs a single gradient ascent step as the following formula:

$$\mathbf{x}^* = \mathbf{x} + \eta\,\mathrm{sign}\left(\nabla_x L\left(g\left(\mathbf{x};\theta\right), y\right)\right), \tag{2}$$

$\mathbf{x}$ is the original input sample, $g(\mathbf{x};\theta)$ presents the model parameterized by θ, y is the label corresponding to the x, and the $L(g(\mathbf{x};\theta), y)$ is the loss function of the classifier. ∇_x is the gradient of the given loss L, which means the direction where the loss increases the most.

We can control bandwidth overhead from small to large by adjusting param η.

Optimization-based attack C&W was proposed by Carlini & Wagner in 2017. This algorithm generates adversarial perturbation based on certain constraints as the following formula:

$$\min\|\delta\|_p^2\ s.t.\, g(\mathbf{x}+\boldsymbol{\delta}) \neq y \text{ and } \mathbf{x}+\boldsymbol{\delta} \in X, \tag{3}$$

$\mathbf{x}$ is also the original input sample and the added perturbation is constrained by $\mathscr{L}_p$ to keep small. The $g(\mathbf{x}+\delta)$ is the obtained result under constraint conditions.

The basic iterative method (BIM) was proposed by A. Kurakin in 2016, it increases the loss of the classifier by adjusting the direction after each step. It iteratively computes as following:

$$\mathbf{I}_\rho^{i+1} = \mathrm{Clip}_\epsilon\left\{\mathbf{I}_\rho^i + \alpha\,\mathrm{sign}\left(\nabla J\left(\theta, \mathbf{I}_\rho^i, \ell\right)\right)\right\}, \tag{4}$$

$\mathbf{I}_\rho^i$ presents the perturbed input at the i^{th} iteration and $\mathrm{Clip}_\epsilon\{.\}$ clips the input in its argument at ϵ and α determines the step size. The BIM algorithm starts with $\mathbf{I}_\rho^0 = \mathbf{I}_c$ and runs for the number of iterations determined by the formula $\lfloor\min(\epsilon+4, 1.25\epsilon)\rfloor$.

Deepfool was proposed by Moosavi-Dezfooli, it perturbs the input by a small vector, which is computed to take the resulting image to the boundary of the polyhedron at each iteration. The final perturbation is accumulated by perturbations added in each iteration when the original decision boundaries of the network change their label.

All these adversarial sample methods are designed to add perturbations to the area of the entire image. But in our scene, we can only change the traffic characteristics between client and entry relay. So, we can only change the part of matrix data. In addition, the ways we add perturbations are by padding packet to change packet size and inserting dummy packets to change interpacket delays. Thus, the value of our adversarial perturbation will always be positive. In order to achieve these requirements, we add extra constraints as follows:

$$St.\begin{cases} P > 0, \\ x \in S, S = \left\{T_i^u; T_i^d; S_i^u; S_i^d\right\}, \end{cases} \tag{5}$$

where P presents the perturbations value we add, x presents the input, and S presents the area we can change.

3.3. Implement Defense in the Real World.

When we think about the actual implementation of our defense in the real world, we must face other challenges. First, the websites are not immutable, and the manager could deploy new functionality, update index pages, launch new activities, etc. So, the characteristics of the transmitted packets will change irregularly and lead to the inefficiency of adversarial samples. Second, we have talked about the limit of adding perturbation in Section 3.2, and we know that only traffic between client and entry relay can be changed. Under this circumstance, we will consider two modes naturally: full-duplex and simplex, who is better? Third, due to network fluctuations, packets might be delayed or received quicker, which will cause the precomputing adversarial examples loss its effect. To meet these requirements, we design a system consisting of some components, as shown in Figure 2.

To solve the first problem, we create "traffic consensus" concept that derives from Tor consensus [38] and stores in a center server. Users can fetch this traffic consensus before connecting to the destination server and add perturbation into live traffic according to the content of traffic consensus. In this traffic consensus, we build the mapping relationship that has the key of website domain and value of corresponding traffic characteristics.

There is an automatic crawler system that collects traffic characteristics termly behind this traffic consensus. Our center server has a Tor client that will request W_n websites that users mostly access like Alexa top 50,00 every T time and check the live status by status code. In addition, we made our exit Tor traffic tunnel through our own SOCKS proxy server. Thus, we can capture ingress Tor flows and the egress Tor flows. If the monitored website is live, dump the traffic file p by tcpdump. Next, we will process p and extract traffic characteristics including the first 300 packets' size and delays. Finally, we will use these data to generate adversarial samples and write them to the traffic consensus with the website domain. The specific details are shown as Algorithm 1.

To solve the second problem, we need to consider differences between full-duplex and simplex. The simplex means that only inserting dummy packets into flows from client to entry relay, it could be done more easily because we can add perturbation at Tor client directly. But it brings other problems: the area where we can add perturbation is further limited and bandwidth overhead is too large to bear.

The full-duplex means we can add perturbation form client to entry relay and entry relay to client. It has more area to add perturbation than simplex. However, the dummy packets we add will go through entire circuit (client- > entry relay- > middle relay- > exit relay- > server). This has definitely increased bandwidth overhead. Thus, we design a drop dummy packets mechanism to further reduce bandwidth overhead. The goal of our approach is to letting adversaries to eavesdrop on dummy packets, and the circuit does not pass through dummy packets. Therefore, we should have a

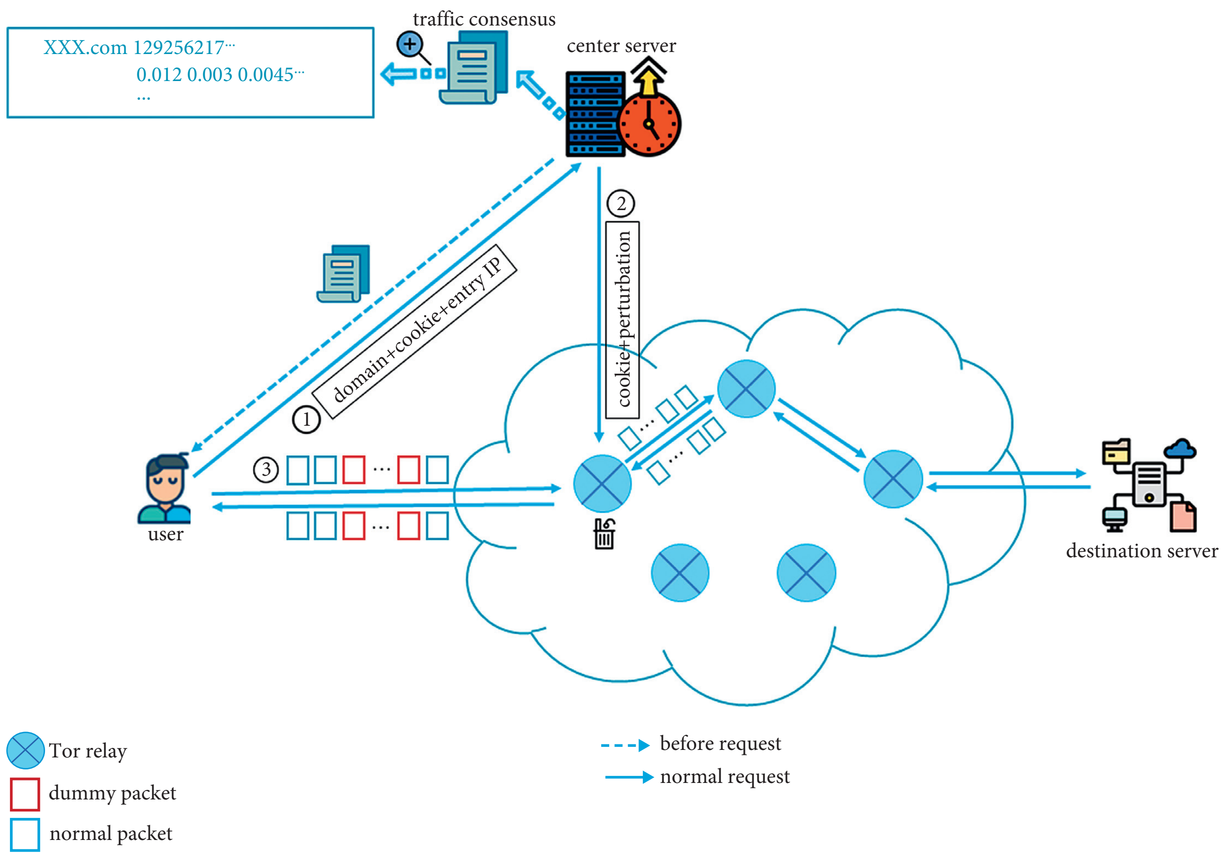

Figure 2: The overview of our system and the entire process of protecting users to request websites from flow correlation attacks.

```
Input: Cycle $T$, Time $t$, Website Groups $W_n$, Traffic File $p$,
    Adversarial Samples Method $A$
Output: Traffic consensus
(1)  While $t$ % $T$ == 0
(2)  For website $w$ in $W_n$ do
(3)  If $w$ is live then
(4)      Dump $p$ from ingress and egress Tor flows
(5)      Extract traffic characteristics $M$ from $p$
(6)      Adversarial samples $S = A(M)$
(7)      Write $S$ to the traffic consensus $\{w: S\}$
(8)  Else
(9)      Continue
(10) End if
(11) End while
```

Algorithm 1: Traffic consensus generate algorithm.

reasonable method to drop dummy packets at the entry relay. We introduce a new control cell INFO, which is referred to in this paper [13]. This cell will record the order of transmitted packets and be send to entry relay before communication begins. Once the entry relay receives INFO cell, it will drop the extra dummy packets that we add at Tor client according to the cell INFO information and send the packets that truly participate in communication to the middle relay.

In addition, the circuit from entry to the client is controlled by entry relay, which means adding perturbation is finished by entry relay. We must provide anonymity that entry relay should not know about users' information of visiting websites. To achieve this goal, our idea is inspired by the rendezvous cookie applied in Tor onion services [39] to establish a connection between the user and an onion service. The users will send a cell that consists of the website domain, which will be visited, a cookie that is a 20-byte cryptographic nonce chosen randomly by the users, and the entry relay's IP to the center server. Once center server receives this cell, it will send the perturbation according to the traffic consensus and the cookie generated by users to the entry relay. The entry will store this cookie and perturbation. When users begin to connect to the entry relay with the cookie, the entry relay will compare the cookie that it stores with the cookie that users send. If they are the same, the entry relay will add perturbation into flows to the client. For the third problem, because we already have the drop dummy packets mechanism and full-duplex mode, the only thing what we must do is buffering subsequent cells until the missing cell arrives at the entry relay.

Implementation: we did not implement all components, because it is a large project that needs the entire Tor community's help to modify Tor source code. But we have designed a set of plans as mentioned above and done a lot of experiments to prove the feasibility of our defense including various adversarial samples methods' effect, traffic consensus used time, the advantage of full-duplex brings less bandwidth overhead, etc. We will show the results in detail in the next section.

4. Results

In this section, we perform a systematic evaluation of our work. Specifically, we compare various adversarial example methods' effects and efficiency against the flow correlation attack model. In our system, we have talked about the advantage and challenges that full-duplex brings, we will further show that our methods' high performance. In addition, we will compare our defense with the state-of-the-art method, and we will test our defense against the traditional flow correlation attack method.

4.1. Data Set. Tor Flow Correlation Data set: in our experiments, we choose to use the public data set of DeepCorr [40]. This data set contains a large number of Tor flows that are captured by visiting Top Alexa 's websites. The storage form in the data set is pickle file, which contains the packet size and interpacket delays. Meanwhile, flow pair that belongs to the same Tor connection(associated flow) is labeled with 1, and the flow pair that belongs to arbitrary Tor connections(nonassociated flow) is labeled with 0. We evaluate our system's performance with 9000 flows.

Sirinam and Rimmer Data set: to our best knowledge, the public flow correlation attack data set has only one that is released by Nasr et al. [10]. But we have pointed our scenarios and challenges are very similar to website fingerprint attacks. Thus, we use two well-known WF data sets including Sirinam et al. [41] and Rimmer et al. [42] to evaluate our system's performance. They both contain Tor users' flow pairs and their corresponding websites. The specific details of these two data sets are presented in Table 3.

4.2. Experiment Results. We test FGSM, C&W, Deepfool, and BIM on the same test data set that contains 9000 flows and compares their protection success rate with the same bandwidth overhead (all use the L_∞ perturbation norm). Except for DeepCorr flow correlation attack, we also test our defense against traditional flow correlation attacks including RAPTOR, Pearson, and Cosine. Table 4 shows the result, and we can see even the worst method FGSM could get the 71.2% protection success rate with only 25% bandwidth overhead, and it is also effective against traditional flow correlation attacks. We must point that because the Pearson and Cosine methods use the static metric to measure the correlation, any slight perturbation will have a big impact on the result. Even our method is oriented to the deep-learning-based attack, and the perturbation we added will also break the pattern that the Pearson and Cosine catch.

We also evaluate FGSM, C&W, Deepfool, BIM against website fingerprint attacks including deep-learning-based attack Var-CNN and non-DNN attacks k-NN, k-FP on the Sirinam, Rimmer data set. Table 5 shows the protection success rate of our method, and we can find adversarial examples is effective for defending WF attacks that get sensitive information by classification model.

Table 3: Two WF data sets used by our experiments.

Data set name	Labels	Training flows (K)	Testing flows (K)
Sirinam	95	7	1
Rimmer	900	5	0.8

In Chapter 3.3, we have talked about the difference between full-duplex and simplex. Full-duplex has more area to add perturbation and less bandwidth overhead because of dropping dummy packets mechanism. Figure 3 shows that how effective of two modes are with FGSM. We find full-duplex mode has a higher protection success rate than simplex mode with the same bandwidth overhead and the same adversarial example generation method. In addition, our system will update the traffic consensus termly, which means that this process must be within a tolerable time frame. We evaluate our system's efficiency on a PC computer that has an i7 11700k CPU and four GTX 2080Ti GPU. We evaluate the total time of generating 500 websites' traffic consensus and adversarial examples on our test data set. Table 6 shows the result, and we can see that our system is very portable. The FGSM method can update the adversarial perturbation in 1575 s seconds. And we should be aware that our hardware is limited, and anyone can extend the hardware environment to further reduce time consumption.

In Table 4, we can see the FGSM gets the worst protection success rate compared to other methods. But because it is a one-step method, it has the highest efficiency. In our system, time consumption is an important indicator because when the website we focus on become more and more, small-time consumption will be magnified a zillion time over. As for the protection success rate, FGSM get 71.2% with 20% bandwidth overhead. It looks a little low, but as we all know traffic flows are very large, adversaries who eavesdropping traffic will take a lot of manual analysis time if the attack success rate cannot reach 95%, and it almost means that this attack is no longer meaningful. Figure 4 shows the protection success rate of FGSM as bandwidth overhead changes, and we can see it also can get a 95% protection success rate with 35% bandwidth overhead, which is lower than state-of-art defense.

4.3. Comparison

4.3.1. Obfs4. To our best knowledge, obfs4 is the state-of-art and official defense. It is a Tor's pluggable transports to defeat censorship by nation-states who block all Tor traffic. obfs4 modified packet timings and packet sizes to defeat flow correlation, by padding or splitting packets, or by delaying packets to perturb their timing characteristics. Table 7 shows that our defense protection success rate compares with obfs4. Table 8 shows that our defense bandwidth overhead compares with obfs4. Our defense has advantages both in protection success rate and bandwidth overhead.

4.3.2. ScrambleSuit. ScrambleSuit [43] is a thin protocol layer above TCP whose obfuscates the transported application data by using morphing techniques and a secret

Table 4: The protection success rate of various adversarial examples against flow correlation attacks with the same bandwidth overhead. PSR presents the protection success rate.

Method	Bandwidth overhead (L_∞)	DeepCorr (PSR) (%)	RAPTOR (PSR) (%)	Pearson (PSR) (%)	Cosine (PSR) (%)
FGSM	0.20	71.2	93.8	97.5	97.2
C&W	0.20	97.4	93.5	97.2	96.4
Deepfool	0.20	93.6	92.7	96.9	96.2
BIM	0.20	87.5	95.6	97.2	95.8

Table 5: The protection success rate of various adversarial examples against website fingerprint attacks with the same bandwidth overhead. PSR presents the protection success rate.

Data set	Method	Bandwidth overhead (L_∞)	K-NN (PSR) (%)	K-FP (PSR) (%)	Var-CNN (PSR) (%)
Sirinam	FGSM	0.20	97.2	97.4	80.3
	C&W	0.20	99.4	99.5	88.8
	Deepfool	0.20	99.7	99.3	94.5
	BIM	0.20	98.5	98.8	86.7
Rimmer	FGSM	0.20	97.2	96.7	86.7
	C&W	0.20	99.9	99.3	93.2
	Deepfool	0.20	98.7	98.1	97.5
	BIM	0.20	98.2	97.5	89.4

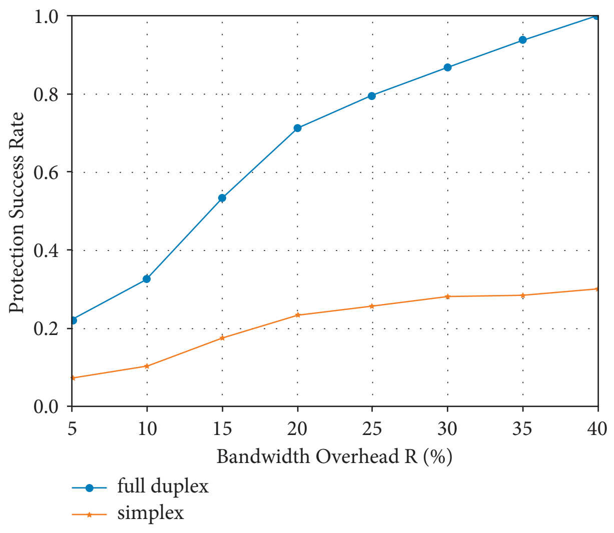

Figure 3: Compare full-duplex with simplex mode under the same conditions.

Table 6: Our system's time consumption under the limited hardware environment.

	Traffic consensus	FGSM	C&W	Deepfool	BIM
Time (s)	1205	1575	47382	28377	4858

exchanged out-of-band. It also has impact on defending the flow correlation attacks. Table 7 shows that our defense protection success rate compares with ScrambleSuit. Table 8 shows that our defense bandwidth overhead compares with ScrambleSuit.

4.3.3. Blind Adversary. Blind Adversary [50] create universal adversarial perturbations by GANs (generative adversarial networks). This approach protects against both flow correlation attack and website fingerprint attack but require significant additional resources and bandwidth overhead. Table 7 shows that our defense protection success rate compares with Blind Adversary. Table 8 shows that our defense bandwidth overhead compares with Blind Adversary.

5. Limitations and Future Directions

As mentioned earlier, this work is focused on defeating CNN-based flow correlation attacks with adversarial

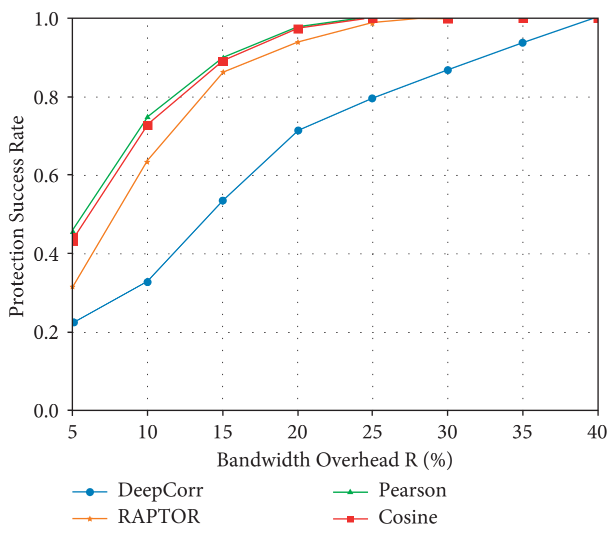

Figure 4: The protection success rate of FGSM as bandwidth overhead changes.

Table 7: The protection success rate of our defense, obfs4 and ScrambleSuit with the same bandwidth overhead.

Method	Bandwidth overhead (L_∞)	DeepCorr (PSR) (%)
FGSM	0.20	71.2
C&W	0.20	97.4
Deepfool	0.20	93.6
BIM	0.20	87.5
Obfs4	0.20	44.5
ScrambleSuit	0.20	22.8
Blind adversary	0.20	88.9

Table 8: The bandwidth overhead of our defense, obfs4 and ScrambleSuit with the same protection success rate.

Method	Bandwidth overhead (L_∞)	DeepCorr (PSR) (%)
FGSM	0.41	100
C&W	0.22	100
Deepfool	0.24	100
BIM	0.27	100
Obfs4	0.78	100
ScrambleSuit	0.82	100
Blind adversary	0.33	100

examples. At present, there are a lot of research about defending the adversarial examples, and the adversarial training is one of the most effective approaches. Adversary can compute our adversarial perturbations and retrain their models against them to improve robustness. Future work can extend our system to defeat adversarial training and other methods that aim to reduce the effect of adversarial examples.

6. Conclusion

In this paper, we evaluate the effect of using adversarial samples to defend flow correlation attacks, and the experimental results show that we achieved a good performance. We further consider implementing our defense in the real world. And we find some challenges we must face. To solve these problems, we design a system including traffic consensus, full-duplex mode, and drop dummy packets mechanism. Our system not only makes adding adversarial perturbations become reality but also further reduce bandwidth overhead.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China NSFC (grant nos. 62072343 and U1736211) and the National Key Research Development Program of China (grant no. 2019QY(Y)0206).

References

[1] T. Metrics, “Tor Metrics,” 2021, https://metrics.torproject.org/userstats-relay-country.html.

[2] W. Diao, X. Liu, Li Zhou, and K. Zhang, “No pardon for the interruption: new inference attacks on android through interrupt timing analysis,” in *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, May, 2016.

[3] X. Liu, Z. Zhou, and W. Diao, “When good becomes evil: keystroke inference with smartwatch,” in *Proceedings of the CCS'15: 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver, Colorado, US, October, 2015.

[4] R. Schuster, V. Shmatikov, and E. Tromer, “Beauty and the burst: remote identification of encrypted video streams,” *26th USENIX Security Symposium (USENIX Security*, vol. 17, 2017.

[5] V. Shmatikov and M.-H. Wang, “Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses,” in *Proceedings of the European Symposium on Research in Computer Security*, pp. 18–33, Hamburg, Germany, September, 2006.

[6] Y. Zhang and V. Paxson, “Detecting stepping stones,” *USENIX Security Symposium*, vol. 171, p. 184, 2000.

[7] A. Blum, D. Song, and S. Venkataraman, “Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds,” in *Proceedings of the Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, RAID*, Sophia Antipolis, France, September, 2004.

[8] S. J. Murdoch and P. Zieliński, “Sampled Traffic Analysis by Internet-Exchange-Level Adversaries,” in *Proceedings of the Privacy Enhancing Technologies*, pp. 167–183, Ottawa, Canada, June, 2007.

[9] Y. Sun, A. Edmundson, and L. Vanbever, “RAPTOR: Routing Attacks on Privacy in Tor,” in *Proceedings of the USENIX Security 2015*, Washington, D.C.USA, August, 2015.

[10] M. Nasr, A. Bahramali, and A. Houmansadr, “DeepCorr: Strong flow correlation attacks on tor using deep learning,” in *Proceedings of the CCS '18: 2018 ACM SIGSAC Conference on*

Computer and Communications Security, Toronto Canada, October, 2018.

[11] Y. Sun, A. Edmundson, and N. Feamster, "Counter-RAPTOR: safeguarding tor against active routing attacks," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, California, USA, May, 2017.

[12] Obfs4, "Obfs4," https://bridges.torproject.org/bridges?transport=obfs4.

[13] W. D. L. Cadena and A. Mitseva, "Traffic-sliver: fighting website fingerprinting attacks with traffic splitting," in *Proceedings of the . CCS (2020*, pp. 1971–1985, Virtual Event, USA, October, 2020.

[14] G. Cherubin, J. Hayes, and M. Juarez, "Website fingerprinting defenses at the application layer," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, pp. 186–203, 2017.

[15] S. . Henri, G. Garcia-Aviles, and P. Serrano, "Protecting against website fingerprinting with multihoming," *PoPETS*, vol. 2, no. 2020, pp. 89–110, 2020.

[16] X. Cai and N. Rishab, "Cs-buflo: A congestion sensitive-website fingerprinting defense," in *Proceedings of the WPES (2014)*, pp. 121–130, Scottsdale, AZ, USA, November, 2014.

[17] X. Cai, "Asystematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the CCS (2014)*, pp. 227–238, Scottsdale, AZ, USA, November, 2014.

[18] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "peek-a-boo, I still see you: why efficient traffic analysis countermeasures fail," in *Proceedings of the IEEE S&P (2012*, pp. 332–346, IEEE, San Francisco, CA, USA, May. 2012.

[19] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *Proceedings of the Computer Security - ESORICS 2016*, pp. 27–46, Springer, Heraklion, Greece, September, 2016.

[20] T. Wang and I. G. Walkie-talkie, "An efficient defense against passive website fingerprinting attacks," in *Proceedings of the of USENIX Security*, pp. 1375–1390, San Diego, CA, USA, August, 2017.

[21] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, March, 2016.

[22] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical blackbox attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, April, 2017.

[23] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," 2015, https://arxiv.org/abs/1412.6572.

[24] S. M. M. Dezfooli, A. Fawzi, F. Omar, and F. Pascal, "Universal adversarial perturbations," in *Proceedings of the The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Juan, PR, USA, July 2017.

[25] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (sp)*, May, 2017.

[26] D. Song, K. Eykholt, I. Evtimov et al., "Physical adversarial examples for object detectors," in *Proceedings of the 12th USENIX Workshop on Offensive Technologies (WOOT 18)*, USENIX Association, Vancouver, BC, Canada, August, 2018.

[27] Y. Shi, S. Wang, and Y. Han, "Curls & whey: boosting black-box adversarial attacks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, June, 2019.

[28] N. Carlini and D. Wagner, "Audio adversarial examples: targeted attacks on speech-to-text," in *Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW)*, May, 2018.

[29] Q. Yao, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *Proceedings of the 36th International Conference on Machine Learning, Volume 97 of Proceedings of Machine Learning Research*, PMLR, California, USA, June, 2019.

[30] P. Neekhara, S. Hussain, P. Pandey, S. Dubnov, J. McAuley, and F. Koushanfar, "Universal adversarial perturbations for speech recognition systems," in *Proceedings of the Interspeech*, Seattle, WA, USA, September 2019.

[31] J. Ebrahimi, A. Rao, L. Daniel, and D. Dou, "Hotflip: white-box adversarial examples for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia, July, 2018.

[32] Y. Belinkov and Y. Bisk, "Synthetic and natural noise both break neural machine translation," 2018, https://arxiv.org/abs/1711.02173.

[33] P. Neekhara, S. Hussain, S. Dubnov, and F. Koushanfar, "Adversarial reprogramming of text classification neural networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, June, 2019.

[34] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015, https://arxiv.org/abs/1412.6572.

[35] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, https://arxiv.org/abs/1607.02533.

[36] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2016, https://arxiv.org/abs/1608.04644.

[37] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, Seattle, WA, USA, June, 2016.

[38] T. consensus, "Tor consensus," 2006, https://gitweb.torproject.org/torspec.git/tree/path-spec.txt.

[39] T. Rendezvous, "Tor rendezvous specification version 3," 2017, https://gitweb.torproject.org/torspec.git/tree/rend-spec-v3.txt.

[40] DeepCorr Dataset, "DeepCorr Dataset," 2018, http://traces.cs.umass.edu/index.php/Network/Network.

[41] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: undermining website fingerprinting defenses with deep learning," in *Proceedings of the CCS (2018*, pp. 1928–1943, Toronto, Canada, October, 2018.

[42] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Proceedings of the NDSS*, San Diego, California, USA, February, 2018.

[43] P. Winter, T. Pulls, and J. F. ScrambleSuit, "A polymorphic network protocol to circumvent censorship," in *Proceedings of the WPES '13 Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, New York; NY, USA, August 2013.

[44] S. Xia, B. Chen, and G. Wang, "Two classification methods based on a novel multiclass label noise filtering learning framework," *IEEE Transactions on Neural Networks and Learning Systems*, no. 1–15, 2021.

[45] X. Yu, T. Liu, and M. Gong, "Transfer learning with label noise," 2017, https://arxiv.org/pdf/1707.09724.pdf.

[46] S. Xia, S. Zheng, G. Wang, X. Gao, and B. Wang, "Granular ball sampling for noisy label classification or imbalanced classification," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2021.

[47] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 447–461, 2016.

[48] M. Numan, F. Subhan, W. Z. Khan et al., "A systematic review on clone node detection in static wireless sensor networks," *IEEE Access*, vol. 8, Article ID 65450, 2020.

[49] Z. Guo, L. Tang, T. Guo, K. Yu, M. Alazab, and A. Shalaginov, "Deep Graph neural network-based spammer detection under the perspective of heterogeneous cyberspace," *Future Generation Computer Systems*, vol. 117, pp. 205–218, 2021.

[50] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating DNN-based traffic analysis systems in real-time with Blind adversarial perturbations," in *Proceedings of the of USENIX Security*, San Diego, CA, USA, August, 2021.