High-Confidence Computing 3 (2023) 100099

Contents lists available at ScienceDirect

High-Confidence Computing

journal homepage: www.sciencedirect.com/journal/high-confidence-computing

Research

Discovering onion services through circuit fingerprinting attacks

Bin Huang *, Yanhui Du

College of Information Technology & Cyber Security, People's Public Security University of China, Beijing 100091, China

ARTICLE INFO

Article history:
Received 7 November 2022
Revised 2 December 2022
Accepted 19 December 2022

Keywords:
Tor
Anonymity
Circuit fingerprinting
Onion service

ABSTRACT

Tor onion services provide anonymous service to clients using the Tor browser without disclosing the real address of the server. But an adversary could use a circuit fingerprinting attack to classify circuit types and discovers the network address of the onion service. Recently, Tor has used padding defenses to inject dummy cells to protect against circuit fingerprinting attacks. But we found that circuits still expose much information to the adversary. In this paper, we present a novel circuit fingerprinting attack, which divides the circuit into the circuit generated by the client and the circuit generated by the onion service. To get a more effective attack, we tried three state-of-the-art classification models called SVM, Random Forest and XGBoost, respectively. As the best performance, we attain 99.99% precision and 99.99% recall when using Random Forest and XGBoost classification models, respectively. And we also tried to classify circuit types using our features and the classification model mentioned above, which was first proposed by Kwon. The best performance was achieved with 99.99% precision and 99.99% recall when using the random forest classifier in circuit type classification. The experimental results show that we achieved highly accurate circuit fingerprinting attacks even when application-layer traffic is identical and some type of circuits using the defenses provided by Tor.

1. Introduction

Tor provides anonymity to users and onion services through multi-hop and source routing technologies. So before transmitting data, each host selects three or more relays (or called onion router) to establish a virtual connection, which is called a circuit in the Tor protocol. The circuit can be divided into general circuit, client-side Introduction Point circuit (Client-IP circuit), client-side Rendezvous Point circuit (Client-RP circuit), onion service-side Introduction Point circuit (OS-IP circuit), and onion service-side Rendezvous Point circuit (OS-RP circuit) according to the purpose of use.

Circuit fingerprinting attack is a traffic analysis attack against Tor which break or reduce the anonymity that Tor aims to provide. Kwon et al. [1] discovered the fingerprint features of circuits and used the features to first propose a circuit fingerprinting attack. To measure the popularity of onion services, Jansen et al. [2] proposes a circuit fingerprinting attack that can be implemented in the middle relay.

Against circuit fingerprinting attacks, Tor has deployed a defense framework that extends the Website Traffic Fingerprinting Protection with Adaptive Defense (WTF-PAD) algorithm in 0.4.1.1-alpha. In addition, two pairs of padding machines are established on this framework: Client-side introduction circuit hiding machine and Client-side rendezvous circuit hiding machine. They send dummy cells during the construction of the Client-IP and the Client-RP circuits respectively, so that the first 10 cell sequences of these two types of circuits are the same as those of the general circuit.

But the circuit still leaked much information, despite two padding machines tried to hide features of circuits. Therefore, we believe that Tor is still vulnerable to circuit fingerprint attacks. This would cause malicious entry relays to use circuit fingerprinting attacks to threaten the anonymity provided by Tor.

In this paper, we propose a novel circuit fingerprint attack to discover onion services against the Tor network using two pairs of padding machines, which divides circuits into client circuit and onion service circuit. Using our attack, an attacker can identify onion services hidden among millions of Tor users with high accuracy. This detection reduces the anonymity of onion services.

Compared with Kwon's circuit fingerprinting attack, we discard features of the circuit construction sequence and duration of activity used by Kwon, and use counting-related and direction-related features. Our goal is to discover onion services, so we did not need to classify circuit types like Kwon did, but divide them into two categories: client and onion service.

To evaluate our circuit fingerprinting attack, we collected a large amount of circuit data using modified Tor in Shadow. We conduct experiments with SVM, Random Forest and XGBoost classification models respectively in order to select the best performance model. Experimental results show that we attain 99.99%

* Corresponding author.
E-mail address: beenhuang@126.com (B. Huang).

https://doi.org/10.1016/j.hcc.2022.100099

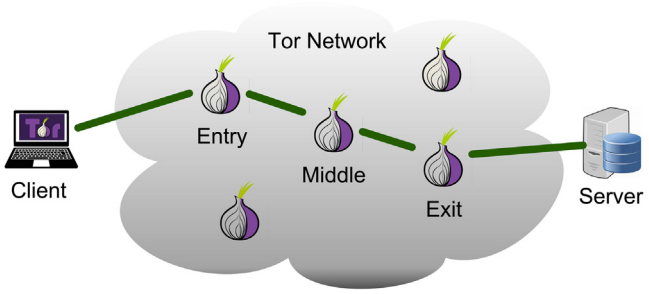


Fig. 1. General circuit for Internet services.

precision and 99.99% recall using Random Forest and XGBoost models.

The contributions of our work are as follows:

- We propose a novel circuit fingerprint attack. It does not distinguish the types of circuits proposed before, but the identity of the host (client or onion service) that generated circuits.
- The dataset used in the paper was obtained by running the modified Tor software in the Tor virtual network environment provided by Shadow.
- We then show the effectiveness of our circuit fingerprinting attack using the dataset that we collected in Shadow. The attack reaches 99.99% accuracy even with a small number of features and traditional machine learning classification models.
- Based on the study of the attack and the analysis of the experimental results, we found that there were obvious differences in the traffic model between client and onion service in Tor. And we propose a new direction in defense against circuit fingerprinting attacks.

The rest of the paper is structured as follows. We survey background and related work in Section 2. We next discuss, in Section 3, our threat model. In Section 4, the methodology is introduced. We illustrate experimental results in Section 5 and the discussion is given in Section 6. Finally, we conclude in Section 7.

2. Background and related work

2.1. Tor network

Tor [3], the second-generation onion router, is one of the most popular anonymous networks and a censorship resistance system. As of May 2022, the Tor network consists of about 7,000 volunteer-run relays, more than 2,700 bridges (non-public relays), about 5,000 onion services, and more than 2.5 million clients use Tor to access the Internet every day [4].

When a client wants to access the Internet, the client selects multiple relays to establish a virtual multi-hop circuit (3 by default), as shown in Fig. 1.

A client firstly chooses exit relay as the last relay, allowing the relay to communicate directly with Internet service external to the Tor network, chooses one of his own entry guards set as his first relay and chooses a new random middle relay. The client then negotiates a temporary shared encryption key with the entry relay, shares the key with the middle relay through the entry relay, and in this way shares the key with the last exit relay. Once the client shares the encryption key with each relay, the client can access the Internet service through this circuit. This circuit that accesses the external is usually called a general circuit.

2.2. Onion services

The Tor network not only provides anonymity to clients, but also to servers called onion services [5]. As shown in Fig. 2, the connection between the client and the server is established anonymously as follows:

In order to be accessible by clients, the onion service maintains several long-term circuits, which we call the onion service-side Introduction Point circuit (OS-IP circuit). The onion service randomly selects some relays with the middle flag and establishes three hops or direct connection to relays. The network address of the Introduction Point (IP) relay and other onion service information are encrypted with its public key to form a descriptor of an onion service, and then the descriptor is sent to a distributed database composed of relays that have HSDir flag. the last relay of this circuit is called the Introduction Point. It acts as a proxy for the onion service, mainly forwarding connection requests of clients to the onion service.

When a client wants to access an onion service, he first selects a middle relay as a Rendezvous Point (RP) and builds a three hops or direct circuit at that relay (we called this circuit Client-RP circuit). In order to get the address of the Introduction Point relay of the current onion service, the client sets up a circuit connected to the HSDir relay and fetches the onion service descriptor to access. The client then establishes a circuit (we called this circuit Client-IP circuit) to one of the onion service's IP relays, and sends a cell to this relay containing the address of client's RP relay and a one-time secret key. The IP relay that receives the connection request from the client forwards the cell to the onion service via the OS-IP circuit.

The onion service decrypts the cell sent by the client to obtain RP address. It then builds a circuit to the RP relay (we called this circuit OS-RP circuit) and sends a response cell containing the one-time secret key. The RP relay forwards this cell to the client via the Client-RP circuit. At this point, a RP circuit between the client and the onion service is established, and can be used to send application data to each other.

2.3. Circuit fingerprinting attacks

Circuit fingerprinting attack is a approach to classify and discover specific Tor circuits through Tor circuit fingerprints on the onion router.

Kwon et al. [1] were the first to find that Tor circuits presents different characteristics in construction sequence, duration of activity, and proposed a circuit fingerprinting attack. It is classified by these features using C4.5 decision tree classification model to achieve more than 98% true positive rate (TPR) and less than 0.1% false positive rate (FPR). Additionally, he used the traditional website fingerprinting attack on the OS-RP circuit to deanonymize onion services.

Jansen et al. [2] proposed a circuit fingerprint attack in the middle relay to measure the popularity of onion services. He extracted the cell type and relay command, counts of the number of cells, the total number of cells from circuits and the guard and exit relays address from the network status consensus file, and used Random Forest machine learning algorithm to distinguish circuits. This attack divides traffic between Client-RP circuit and other circuits with 91.77% TPR and 8.23% FPR.

Circuit fingerprinting attacks are inspired by the related field of website fingerprinting attacks. A website fingerprinting attack is a technique that detects a user's access to a target website without decrypting encrypted traffic through the traffic fingerprint feature of the website. So far, many attacks using different features and classification models have been proposed on the Tor network [6–13].

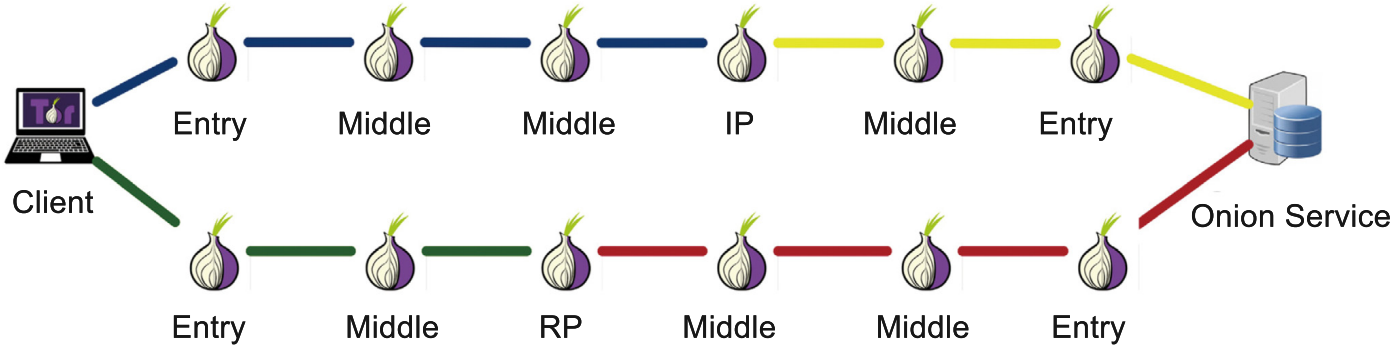


Fig. 2. Four types of circuits in an onion service connection. The blue circuit is the Client-IP circuit, the green circuit is the Client-RP circuit, the yellow circuit is the OS-IP circuit, and the red circuit is the OS-RP circuit.

2.4. Circuit fingerprinting defense

Tor circuit-level padding subsystem [14] is a defense against circuit fingerprinting attacks, which sends dummy cells to obfuscate original traffic features to defend traffic analysis attacks by external and internal adversaries. The subsystem is an extension of the WTF-PAD event-driven state machine design. Website Traffic Fingerprinting Protection with Adaptive Defense (WTF-PAD) proposed by Juarez et al. [15] is a defense framework against website fingerprinting and circuit fingerprinting attacks on Tor. It is an extension of Adaptive Padding (AP) algorithm proposed by Shmatikov and Wang [16]. The AP algorithm generates dummy message in a targeted manner in accordance with the outgoing traffic pattern examined by the defender to disrupt the inter-arrival time distribution of the traffic. The WTF-PAD algorithm adds a bidirectional padding mechanism, a negotiation mechanism between communication parties, and a soft stopping condition mechanism to enable more complex padding strategies.

At present, Tor deploys two pairs of circuit padding machines: Client-side introduction circuit hiding machine and Client-side rendezvous circuit hiding machine. They are mainly used to hide features of Client-IP and Client-RP circuit construction sequences. In this approach, a client and a middle relay send dummy cells to each other up to first 10 cells after negotiation, so that the Client-RP and the Client-IP construction sequences were exactly the same as the general circuit construction sequence. This prevents an entry relay from easily discovering Client-RP and Client-IP circuits by circuit construction sequence.

In order to protect clients accessing onion service against circuit fingerprinting attacks, Kadianakis et al. [17]. proposed two types of padding machines: fractional-delay strategy and preemptive circuit padding strategy. The fractional-delay strategy inserts a dummy onion handshake in the general circuit after receiving the client's request to access Internet service, then sends the application data after the dummy onion handshake is complete; Preemptive Circuit Padding strategy (PCP) injects a dummy onion handshake in a preemptive circuit. A preemptive circuit is a circuit that a Tor client continuously builds, and stays dormant until it is used. Therefore, this strategy does not delay client traffic compared with the previous strategy.

3. Threat model

The onion routers of the Tor network are built and maintained by volunteers. Among these relays, some that meet the specific requirements of the Tor protocol are given Guard flag, and from then on, connect directly to hosts as a first node of circuit. Therefore, these relays get the network addresses of hosts with which they communicate.

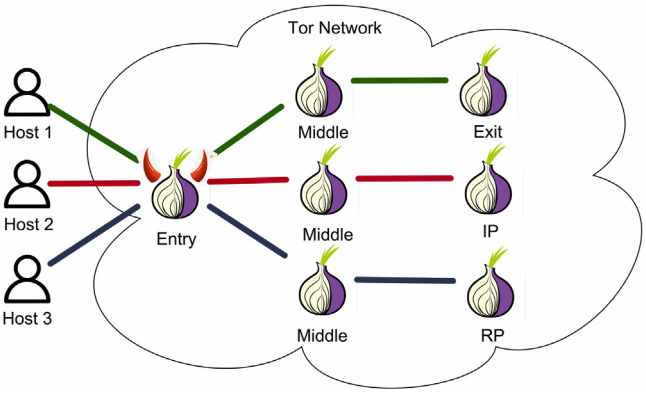


Fig. 3. The circuit fingerprinting threat model.

In this work, we assume a relay adversary that is: the adversary builds one or more onion routers in the Tor network that have been given Guard flag. And the adversary can only passively record traffic information but cannot insert, modify, delay, and drop them.

As shown in Fig. 3, the adversary who controls the first relay can obtain network addresses of hosts. To distinguish the identities of these hosts, the adversary records all cell information exchanged with hosts, extracts features that best represent characteristics of circuits and classifies these circuits through a trained supervised learning classifier. Based on the classification results, the adversary can find onion services in all hosts and record network addresses of these onion services.

4. Methodology

4.1. Features

Tor specifies that fixed-size packets (called cell) are transmitted over circuits, so traffic on a circuit is represented as a sequence of cells. In this work, we only extract the cell command, the relay command and direction information in each cell. A cell sequence is represented as

$$[10:0:+1],\ [11:0:-1],\ [9:14:+1],\ [3:15:-1],$$
$$[9:0:+1],\ [3:0:-1],\ [9:0:+1],\ [3:0:-1],$$
$$[9:0:+1],\ [9:0:+1],\ [9:0:+1],\ [9:0:+1],$$
$$[9:0:+1],\ [3:0:+1],\ [3:0,+1],\ \ldots$$

where the first number in square brackets is the cell command, the second number is the relay command (0 means the entry relay cannot recognize the relay command), and the last one is

the direction of the cell (+1 indicates incoming cell to the relay, −1 indicates outgoing cell from the relay).

We extracted features from our a large amount of circuits based on the observation. First the construction cell sequence and duration of activity features used by Kwon are discarded. These features are no longer valid due to the effect of Tor padding machines. And all time-related features that are most affected by network jitter are not used, although these features have been used in some previous traffic classification studies.

Based on incorporate the previous observations, we decided to use counting-related and direction-related features. Finally, the following five features are selected.

- the number of outgoing cells
- the number of incoming cells
- the total number of cells
- the number of outgoing cells as fraction of total cells
- the number of incoming cells as fraction of total cells

4.2. Classification

We propose a novel circuit fingerprinting attack that identifies circuits created by onion services from all circuits in order to discover onion services.

In this attack, the adversary controls one or more entry nodes that connect directly to hosts. Therefore, the entry relay can obtain network addresses of hosts, but cannot distinguish identity of hosts (client or onion service). The relay records information of cells sent and received to the host and arranges these cells to form the cell sequence of the circuit. Then, it extracts features from the cell sequence and uses a trained classification model to classify the circuit as either client circuit or onion service circuit.

In this work, machine learning algorithms are used to predict identity of hosts. Previous work by Kwon et al. used a decision-tree driven classifier to predict types of circuits. we choose SVM, Random Forest and XGBoost classification models and implemented these classifiers using sklearn [18] and xgboost [19] python APIs on the features we previously described.

- sklearn.svm.SVC: it is an open source Phython implementation of support vector machine algorithm based on libsvm [20] library.
- sklearn.ensemble.RandomForestClassifier: it is a random forest classifier provided by sklearn. Random Forest is a classification algorithm consisting of a number of decision trees, and uses averaging to improve the predictive accuracy and control over-fitting.
- XGBoost: it stands for Extreme Gradient Boosting and is an implementation of gradient boosted decision trees designed to be highly efficient and flexible.

In order to verify the accuracy of our classification model for circuit type classification under the interference of padding machine, we also classified circuits by type using our features and classification models. Specifically, circuits are classified into five categories: general circuit, Client-IP circuit, Client-RP circuit, OS-IP circuit and OS-RP circuit.

4.3. Differences between our circuit fingerprinting and Kwon's circuit fingerprinting

We explain the significant differences of our circuit fingerprinting compared to Kwon's circuit fingerprinting.

Features. Instead of using features of circuit construction sequence and duration of activity, we use counting-related and direction-related features. This is because Client-side introduction circuit hiding machine and Client-side rendezvous circuit hiding machine confuse circuit construction sequence. In addition, Client-side introduction circuit hiding machine extend the lifetime of the introduce circuit, hiding the short-term lifetime feature of the introduce circuit.

Classification. We used a binary classification approach. According to the identity of hosts, it can be divided into client and onion service. Since we only focus on finding onion service, we do not need to classify types of circuits.

Table 1
Information on hardware and software.

Hardware and software	Specific information
CPU	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30 GHz
Memory	32 GB
GPU	NVIDIA Tesla V100 SXM2 32GB
Operating System	Fedora 34
Shadow	2.0.0
Tor	0.4.6.9
TGen	1.0.0
TorNetTools	1.1.0
Python	3.8.3
scikit-learn	1.0.2
XGBoost	1.6.1
imbalanced-learn	0.9.0

5. Experiment

5.1. Experiment setup

In the past four months we have done all the experiments on the workstation of our college. we set up Tor virtual network environment using Shadow and collected the raw circuit data in the simulated environment. A tool written in Python was then used to extract the cell information of circuits from Tor log files. An oversampling algorithm provided by the imblearn [21] API was used to increase the number of samples of some types of circuits, and classification models implemented by the sklearn and XGBoost were used for classification. All software used in the experiment were the latest stable version at that time. Table 1 shows hardware configurations and software versions used in the experiment.

5.2. Data collection

We ran the modified Tor software in the Shadow [22] simulation environment to obtain a large amount of circuits for analysis. Shadow is a discrete-event network simulator developed specifically for Tor network simulation experiments and can run Tor software directly. Therefore, Shadow follows all logic related to Tor circuits. In the simulation environment provided by Shadow, we can build servers, clients, directory authorities, onion services and relays, and can control all nodes. Therefore, we can get circuit data in Shadow without the real Tor network.

We aim to collect a large amount of circuits from entry relays. And we plan not to use time-relating features of circuits, so we do not need to consider network congestion and the size of the Tor network. Therefore, a scaled-down Tor network was built that required lower hardware requirements and took less time to complete. We downloaded the network status consensuses, server-descriptors, bandwidth, onionperf, and userstats-relate-country files for June 2021 from Tor Collector [23] and used TorNetTools [24] to construct a private Tor network with 66 relays (including 26 entry guard relays, 28 middle relays, 3 exit relays, and 9 exitguard relays).

Two different scenarios are set up: one consists of 100 clients and 10 external servers, and clients randomly access a server

through multiple relays in the Tor network; the other consists of 100 clients and 1 onion service, all clients accesses the unique onion service. During the experiment, clients generate simulated traffic using the TGEN [25] tool. In order not to reflect the traffic characteristics of the website in the circuit data collected, all clients use same traffic model to generate traffic.

We repeated the experiment in Shadow 276 times using different initial random seeds. After each experiment, the cell sequence of the circuit, whose circuit ID is the same as the particular type of circuit ID of the client (or onion service), is extracted from the Tor log file of the corresponding entry relay. In order to extract circuit data more easily, we write an extraction tool in Python. The tool records cell commands, relay commands and directions for each cell. After all the experiments, we finally collected a total of 48,088 circuits, including 9,774 onion service circuits.

5.3. Datasets

We used our own Python program to extract features from the cell sequence of the circuit in order to obtain a feature vector for each circuit. And features were extracted from the first 10, 20, 30 and all cells of each cell sequence, respectively.

In each dataset, there are 38,314 client circuit data and 9,774 onion service circuit data. The client circuit data includes 13,798 general circuits, 15,139 Client-IP circuits, 9,377 Client-RP circuits; the onion service circuit data includes 397 OS-IP circuits, 9,377 OS-RP circuits. In the dataset, the number of onion service circuits is relatively less than the number of client circuits. Therefore, the RandomOverSampler algorithm provided by the imblearn API is used to increase the amount of onion service-side circuit data.

In the circuit type classification, the oversampling algorithm mentioned above is also used, since the number of OS-IP circuits is much less than the number of other types of circuits.

5.4. Hyperparameter tuning

To find the optimal values for the hyperparameters of the classifier, we used grid search with k-fold cross validation. The approach is to iterate through various combination of hyperparameters, fit the model on the training set and give the hyperparameter combination with the best k-fold cross validation accuracy. In the experiment, the GridSearchCV class provided by sklearn is used, and the k value is set to 5. We started from the hyperparameter with a large data gap and gradually narrowed the data gap, and finally obtained the optimal hyperparameter of the classifier on our dataset.

Through the above method, we selected the SVM classifier with RBF (Radial Basis Function) kernel, and its C and gamma values were 128 and 128 respectively; 30 decision trees and Gini impurity were used in the Random Forest classifier; 20 boosting rounds were used in the XGBoost classifier, and its objective value was binary:logistic (logistic regression for binary classification). Then, we retrain these classifiers with their optimal hyperparameters on the training set and obtain the final classification model.

In circuit type classification, the SVM classifier used the RBF kernel, and the C and gamma values are 512 and 128 respectively; the Random Forest classifier used 30 decision trees and Gini impurity; the XGBoost classifier used 20 boosting rounds and the objective value was binary:logistic.

5.5. Evaluation metrics

In order to evaluate the proposed method, we use three metrics to evaluate the classification results: precision, recall and F-score.

Precision is calculated as the ratio between the number of positive samples classified correctly to the total number of classified positive samples. TP indicates the number of true positive and FP indicates the number of false positive.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Recall is the ratio of the number of positive samples classified correctly to the total number of positive samples. TP indicates the number of true positive and FN indicates the number of false negative.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

F-score, also called F1-score, is the harmonic mean of precision and recall, that is used to express the performance of the machine learning model.

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

5.6. Experiment results of our circuit fingerprinting attack

Our classification model is trained in two ways. The first method, called n-fold cross-validation, splits data into n disjoint subsets. Then this method iterates n times, with one of the subsets as test set and the others as training set in each iteration. Finally, the results from all n folds are averaged. We set n to 10 in our experiment. The second is 70–30 percentage split in which 70 percent of the sample is used for training and the remaining 30 for testing.

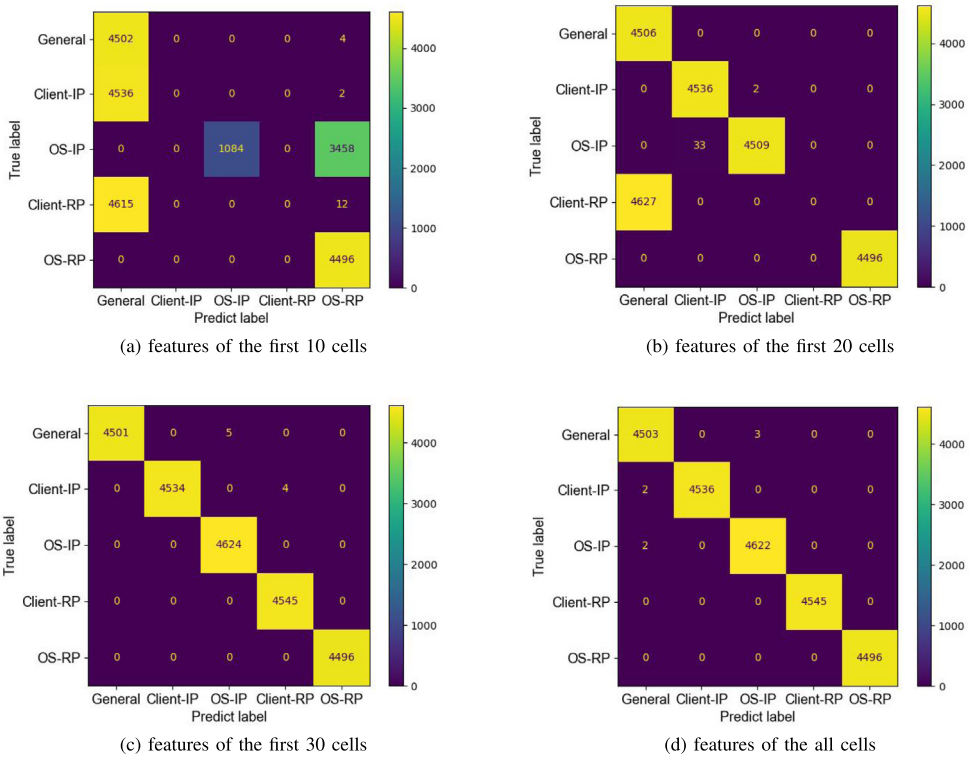
Table 2 gives a summary of the precision, recall and F1 achieved by the considered classifiers for the first 10, 20, 30 and all cells in each circuits using 10-fold cross validation. From the table, we found that the considered classifiers performed well, whether we use the first 10, 20, 30, or all cells. Especially with the first 10 cells, the SVM, RF and XGBoost classifiers have achieved 99.95%, 99.99% and 99.99% precision respectively. The main reason is that the 10th cell of the cell sequence sent by the client is the incoming cell, while the cell sent by the onion service is the outgoing cell (+1 indicates incoming cell to the entry relay, −1 indicates outgoing cell from it).

$$Client : +1, -1, +1, -1, +1, -1, +1, -1, +1, +1$$
$$Onion\ Service : +1, -1, +1, -1, +1, -1, +1, -1, +1, -1$$

In classification using the features of the first 20 cells, all three classifiers performed well with 99.97% accuracy and 99.97% recall. Because from the 11th cell to the 20th cell in the cell sequence, the client is mainly used to send the request message, so most of the cells are incoming cells to the entry relay; while the onion service is mainly used to receive the client's request messages, so most of the cells are outgoing cells from the entry relay. And all classifiers also achieved over 99.97% precision and 99.97% recall using the features of the first 30 and all cells. Mainly because the client receives significantly more cells than it sends, but the onion service does the opposite.

Table 2
Precision (Pr), Recall (Re) and F1 of our circuit fingerprinting attack.

Classifier	Number of cells per circuit											
	10			20			30			All		
	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
SVM	0.9995	0.9995	0.9995	0.9997	0.9997	0.9997	0.9998	0.9998	0.9998	0.9997	0.9997	0.9997
RF	0.9995	0.9995	0.9995	0.9997	0.9997	0.9997	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
XGBoost	0.9995	0.9995	0.9995	0.9997	0.9997	0.9997	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999



(a) features of the first 10 cells

(b) features of the first 20 cells

(c) features of the first 30 cells

(d) features of the all cells

Fig. 4. Confusion matrices for SVM and 70–30 split in the circuit type classification.

Table 3
Precision (Pr), Recall (Re) and F1 of the circuit type classification.

Classifier	Number of cells per circuit											
	10			20			30			All		
	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
SVM	0.3805	0.4490	0.3238	0.8581	0.7987	0.7328	0.9997	0.9997	0.9997	0.9994	0.9994	0.9994
RF	0.4005	0.4489	0.3238	0.8081	0.7986	0.7329	0.9998	0.9998	0.9998	0.9999	0.9999	0.9999
XGBoost	0.3805	0.4490	0.3238	0.8582	0.7987	0.7328	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998

5.7. Experiment results of the circuit type classification

Table 3 shows the precision, recall and F1 achieved in circuit type classification. All classifiers get the worst precision and recall when using the features of the first 10 cells for classification. When using features from the first 20 cells, classifiers achieved over 80% accuracy and about 79% recall. In contrast, using the first 30 or all cells in each circuit, classifiers performed best with over 99% precision and recall. To help explain how come we see such varying results, we reclassified the dataset using the 70–30 split method and displayed the results in the confusion matrix figure.

From Fig. 4a, the SVM classifier (including the other two classifiers) cannot distinguish general, Client-IP and Client-RP circuits only by the count-related features of first 10 cells due to the effect of the padding machine. The classifier does not classify OS-IP circuits and OS-RP circuits well, although there is no interference from the padding machine. Because the number of first 10 cells of the OS-IP circuit and OS-RP circuit created by the onion service is very similar.

As shown Fig. 4b, when using the features of the first 20 cells, the classifier can recognize Client-IP circuits in the three types of circuits created by the client. Because in the first 20 cells, only the IP-Client circuit has more outgoing cells than incoming cells. And classifier is still unable to distinguish the general circuit and the Client-RP circuit, since the first 20 cells of the general circuit and the Client-RP circuit are similar. Among the two types of circuits created by the onion service, the classifier can accurately distinguish between OS-IP and OS-RP circuits. Finally, when using the features of the first 30 cells and all cells, the classifier can distinguish all circuit types with high accuracy in Fig. 4c and Fig. 4d.

Table 4
Results for circuit fingerprintings.

	Kwon's circuit fingerprinting	Our circuit fingerprinting
Accuracy	51.29%	99.96%
Precision	67.29%	99.97%
Recall	45.87%	99.97%
F1	54.55%	99.97%

5.8. Comparison with Kwon's circuit fingerprinting

We compare our circuit fingerprinting attack to Kwon's. According to Kwon's paper, we re-implemented his circuit fingerprinting using Python. To ensure fair comparison, we run Kwon's circuit fingerprinting on our dataset. And we discard the duration of activity feature of Kwon's circuit fingerprinting, since our dataset does not contain timestamps.

Table 4 shows the accuracy, precision, recall and F1 results. Our circuit fingerprinting attains 99.96% accuracy, 99.97% precision and 99.97% recall, which is better than Kwon's circuit fingerprinting attack. Since Tor uses Client-side introduction and rendezvous circuit hiding machines on all client-side circuits, the performance of Kwon's circuit fingerprinting attack with circuit construction sequence feature becomes worse.

6. Discussion

At present, the circuit-level padding machine inject dummy cells during the construction of Client-IP and Client-RP circuits, so that the circuit construction cell sequence of these circuits is the same as that of general circuits, which is used to hide the behavior of clients accessing onion services. However, it is found through experiments that the three types of circuits created by the client (general circuit, Client-IP circuit and Client-RP circuit) are significantly different from the two types of circuits (OS-IP circuit and OS-RP circuit) created by the onion service. This would cause the network addresses of onion services to be easily exposed to malicious entry relays using circuit fingerprinting attacks.

Compared to the IP addresses of clients, the IP addresses of onion services does not change frequently. Therefore, a adversary can further deanonymize these onion services that obtain real IP addresses for a long time. Such as discovering onion service owner information by IP address, or implementing a website fingerprinting attack on the OS-RP circuit to discover the onion address of the onion service.

Therefore, onion service urgently needs protection measures to prevent these circuit fingerprinting attacks. For example, developing new padding machines for onion services or using traffic splitting or merging mechanisms to hide circuit features of onion services.

7. Conclusion

In this work, we propose a novel circuit fingerprinting attack, which allows an entry relay to identify onion services from all hosts. We applied three types of state-of-the-art machine learning algorithms to classify the first 10, 20, 30 and all cells of circuits separately. The experimental results show that the Random Forest and the XGBoost classifiers perform best with 99.99% precision and 99.99% recall when using the first 30 and all cells of circuits.

Next, we might study defense methods against circuit fingerprinting attacks for onion services, such as designing new padding machines for onion services based on the existing Tor padding framework. Alternatively, we can develop some mechanisms such as traffic splitting to obfuscate the features of onion service circuits.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful feedback.

Appendix A. Supplementary data

All code and dataset used in this paper are available via https://github.com/beenhuang/cf-attack-for-discovering-os/.

References

[1] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, S. Devadas, U. Assoc, Circuit fingerprinting attacks: Passive deanonymization of TOR hidden services, in: 24th USENIX Security Symposium, Washington, DC, 2015, pp. 287–302.
[2] R. Jansen, M. Juarez, R. Galvez, T. Elahi, C. Diaz, S. Internet, Inside job: Applying traffic analysis to measure TOR from within, in: 25th Annual Network and Distributed System Security Symposium, NDSS, San Diego, CA, 2018.
[3] R. Dingledine, N. Mathewson, P. Syverson, TOR: The second-generation onion router, in: 13th USENIX Security Symposium, San Diego, CA, 2004, pp. 303–319.
[4] Tor Metrics, https://metrics.torproject.org/.
[5] Onion service, https://github.com/torproject/torspec/blob/main/rend-spec-v3.txt/.
[6] T. Wang, X. Cai, R. Nithyanand, R. Johnson, I. Goldberg, U. Assoc, Effective attacks and provable defenses for website fingerprinting, in: 23rd USENIX Security Symposium, San Diego, CA, Aug 20-22 2014, 2014, pp. 143–157.
[7] A. Panchenko, et al., Website fingerprinting at internet scale, in: 23rd Annual Network and Distributed System Security Symposium, NDSS, San Diego, CA, 2016.
[8] J. Hayes, G. Danezis, U. Assoc, K-fingerprinting: A robust scalable website fingerprinting technique, in: 25th USENIX Security Symposium, Austin, TX, 2016, pp. 1187–1203.
[9] P. Sirinam, M. Imani, M. Juarez, M. Wright, Acm, Deep fingerprinting: Undermining website fingerprinting defenses with deep learning, in: ACM SIGSAC Conference on Computer and Communications Security, CCS, Toronto, CANADA, 2018, pp. 1928–1943.
[10] X. Cai, X.C. Zhang, B. Joshi, R. Johnson, Touching from a distance: Website fingerprinting attacks and defenses, in: 2012 ACM Conference on Computer and Communications Security, NC, United States, 2012, pp. 605–616.
[11] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, W. Joosen, Automated website fingerprinting through deep learning, in: 25th Annual Network and Distributed System Security Symposium, NDSS, San Diego, CA, 2018.
[12] S. Bhat, D. Lu, A. Kwon, S. Devadas, Var-CNN: A data-efficient website fingerprinting attack based on deep learning, Proc. Priv. Enhanc. Technol. (4) (2019) 292–310, 10/ 2019.
[13] Z. Zhuo, Y. Zhang, Z. l. Zhang, X. Zhang, J. Zhang, Website fingerprinting attack on anonymity networks based on profile hidden Markov model, Ieee T. Inf. Foren. Sec. 13 (5) (2018) 1081–1095.
[14] Tor padding subsystem, https://github.com/torproject/torspec/blob/main/padding-spec.txt/.
[15] M. Juarez, M. Imani, M. Perry, C. Diaz, M. Wright, Toward an efficient website fingerprinting defense, in: 21st European Symposium on Research in Computer Security, ESORICS, Heraklion, GREECE, 2016, pp. 27–46.
[16] V. Shmatikov, M.H. Wang, Timing analysis in low-latency mix networks: Attacks and defenses, in: 11th European Symposium on Research in Computer Security, Hamburg, GERMANY, 2006, pp. 18–33.
[17] G. Kadianakis, T. Polyzos, M. Perry, K. Chatzikokolakis, Tor circuit fingerprinting defenses using adaptive padding, 2021, arXiv.
[18] scikit-learn, https://scikit-learn.org/stable/index.html/.
[19] XGBoost, https://xgboost.ai/.
[20] libsvm, https://www.csie.ntu.edu.tw/cjlin/libsvm/.
[21] imbalanced-learn, https://imbalanced-learn.org/stable/index.html/.
[22] R. Jansen, N. Hopper, Shadow: Running TOR in a box for accurate and efficient experimentation, in: NDSS, 2012.
[23] TOR Collector, https://collector.torproject.org/.
[24] R. Jansen, J. Tracey, I. Goldberg, U. Assoc, Once is never enough: Foundations for sound statistical inference in TOR network experimentation, in: 30th USENIX Security Symposium, Electr Network, 2021, pp. 3415–3432.
[25] TGen, https://github.com/shadow/tgen/.