Revisiting Assumptions for Website Fingerprinting Attacks

Weiqi Cui
Oklahoma State University

Tao Chen
Oklahoma State University

Christian Fields
Oklahoma State University

Julianna Chen
Oklahoma State University

Anthony Sierra
Oklahoma State University

Eric Chan-Tin
Loyola University Chicago

ABSTRACT

Most privacy-conscious users utilize HTTPS and an anonymity network such as Tor to mask source and destination IP addresses. It has been shown that encrypted and anonymized network traffic traces can still leak information through a type of attack called a website fingerprinting (WF) attack. The adversary records the network traffic and is only able to observe the number of incoming and outgoing messages, the size of each message, and the time difference between messages. In previous work, the effectiveness of website fingerprinting has been shown to have an accuracy of over 90% when using Tor as the anonymity network. Thus, an Internet Service Provider can successfully identify the websites its users are visiting. One main concern about website fingerprinting is its practicality.

The common assumption in most previous work is that a victim is visiting one website at a time and has access to the complete network trace of that website. However, this is not realistic. We propose two new algorithms to deal with situations when the victim visits one website after another (continuous visits) and visits another website in the middle of visiting one website (overlapping visits). We show that our algorithm gives an accuracy of 80% (compared to 63% in a previous work [24]) in finding the split point which is the start point for the second website in a trace. Using our proposed "splitting" algorithm, websites can be predicted with an accuracy of 70%. When two website visits are overlapping, the website fingerprinting accuracy falls dramatically. Using our proposed "sectioning" algorithm, the accuracy for predicting the website in overlapping visits improves from 22.80% to 70%. When part of the network trace is missing (either the beginning or the end), the accuracy when using our sectioning algorithm increases from 20% to over 60%.

CCS CONCEPTS

• **Security and privacy → Pseudonymity, anonymity and untraceability**; • **Networks → Network privacy and anonymity**.

KEYWORDS

Privacy, Website Fingerprinting, Anonymity

ASIACCS'19 , July, Auckland, New Zealand
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329802

ACM Reference Format:
Weiqi Cui, Tao Chen, Christian Fields, Julianna Chen, Anthony Sierra, and Eric Chan-Tin. 2019. Revisiting Assumptions for Website Fingerprinting Attacks. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3321705.3329802

1 INTRODUCTION

Anonymous communication's goal is to hide the relationship and communication contents among different parties. Once two parties establish an anonymous communication between them, the contents are encrypted and routing information is hidden, thus masking the source and destination IP addresses from third parties. Tor [6, 21] is one of the most popular low-latency anonymity-providing network. It is used by millions of people daily [18]. Tor protects users' privacy through a telescoping three-hop circuit and encrypts the network traffic using onion routing. Although Tor and many other privacy-enhancing technologies such as HTTPS proxy hide the communication contents and network layer contents, the network traffic itself may leak information such as packet size, inter-packet timing information, and direction of the packets (from server to client or other way around).

A website fingerprinting (WF) attack is one where an attacker identifies a user's web browsing information by merely observing that user's network traffic. The attacker is not attempting to break the encryption algorithm or the anonymity protocol. The only information available to the attacker is the metadata information such as packet size, the timing information between packets, and the direction of the packet. The success of this attack is measured by the number of websites correctly identified. The accuracy has been shown to be around 90%[16], thus violating any privacy offered by HTTPS and anonymity services like Tor.

It has been more than 15 years since the first website fingerprinting attack was proposed [11]. A number of studies on this topic have been released since then [3, 16, 24], showing high accuracy in predicting websites in both the open and closed world models. Most previous work rely on certain assumptions. The **goal** of this research is to revisit some of these assumptions, namely: 1) the adversary can record the whole network traffic trace for a website [1], 2) the victim visits one website at a time; here, we consider two cases where i) the victim visits two pages one after the other (continuous visits) and ii) the victim visits a second page before the first one finishes loading (overlapping visits). We propose two new algorithms to deal with these cases. The contributions of this paper are summarized as follows.

- **A "splitting" algorithm to identify two continuous network traces**. We propose a new algorithm based on Hidden

[1] Note that we used trace, network trace, website, and webpage interchangeably

Markov Model to split and detect traces with two continuous pages. We show that our algorithm gives a higher accuracy in finding the split point in two continuous websites (80% compared to 63% in previous work). This is also the first time that the accuracy in directly predicting websites in continuous traffic traces is tested and shown.

- **A "sectioning" algorithm to identify overlapping network traces**. We propose a new algorithm to section the trace into multiple sections and treat each section independently to perform the website prediction. The hypothesis is that if two traces overlap, the beginning of the first trace and the end of the second trace would be unaffected. Sectioning then still allows for correct identification of the two websites. When considering overlapping traces, the accuracy of current techniques for website fingerprinting decreases to 20% – 30%. Our sectioning algorithm improves the accuracy to around 70%.
- **Applying "sectioning" algorithm on partial traces**. By applying "sectioning" on partial traces, the accuracy (62.66%) is higher compared to previous methods (20.76%) on predicting websites with the beginning 5% of the trace missing. When predicting websites with the last parts of the trace missing, the accuracy is comparable. Hence, with sectioning algorithm, we can reduce the impact of missing packets in a network trace.

This paper is structured as follows: in Section 2, we give the related background and terminology of this paper. In Section 3, we propose a new "splitting" algorithm to find the split point in two continuous page traces and present the results. We propose a new "sectioning" algorithm to improve the accuracy in overlapping traces in Section 4 and in partial traces in Section 5. We conclude and provide avenues for future work in Section 6.

2 BACKGROUND

- **Definitions**. We first define some terms used.
  - Trace. A trace is a time series of recorded network packets for a visit to a webpage. Usually, *tcpdump* is used to record the network traffic. A trace contains no background noise, only the network traffic to/from that webpage.
  - Continuous Trace. When a trace consists of two pages, and the second page starts when the first page ends, we call it a continuous trace. It has the same meaning as when the two pages are separated with zero-time.
  - Split Point. When a trace is composed of two pages, the first step is to separate them before further detecting. The point where the second page starts and the first page ends is the split point.
  - Overlapping Trace. When a trace consists of two pages, and the second page starts before the first page ends, we call it an overlapping trace. It has the same meaning as when the two pages are separated with negative-time.
- **Threat Model**. In website fingerprinting attacks, the adversary records network traffic data of his own visits to a list of websites first through the Tor network. Then the adversary can eavesdrop on the link between the victim and the entry node. Figure 1 depicts where the adversary is. We assume the attacker to be a passive observer which means it does not modify transmissions and is not able to decrypt packets. An example of the adversary is Internet Service Providers (ISP), and state-level agencies.

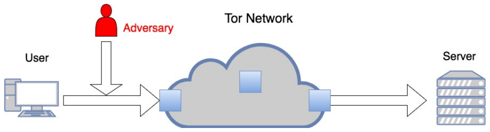

Figure 1: Threat Model.

- **WF Attack Procedures**. Website fingerprinting has been shown to be a serious threat against privacy mechanisms for anonymous web browsing. Researchers have proposed different scenarios for website fingerprinting. The attack and resulting experiment vary from each other; however, they all follow similar steps. A website fingerprinting attack and analysis can be divided into six steps: 1) collect data, 2) extract features from data, 3) select algorithm, 4) build model based on 1) to 3), 5) evaluate real network traffic trace, and 6) evaluate results. Figure 2 shows an illustration of all the steps of a website fingerprinting attack. The last right-most block contains the measurements to evaluate the effectiveness of an attack.
When setting up an experiment for a website fingerprinting attack, the first step is to perform data collection. A network traffic recording tool such as wireshark or tcpdump is used. Before running any scripts to automatically collect data, the configuration of the browser should be set to match the assumptions, such as disabling all plug-ins to avoid background noise and clearing the browser cache. The automated script will then visit websites in a certain order. The time taken to collect data depends on the number of instances recorded for each website and the size of the website list. Features extracted from the recorded network traffic traces will be used for training. Each network trace is composed of a list of features. The features can be treated as attributes in a machine learning context. A classification algorithm is applied to these features to build the attack model. Different websites correspond to different classes. Different network traffic traces are then collected to evaluate the performance of the model. A 10-fold cross validation is often employed to reduce the bias in the evaluation process.
- **Closed world and Open World**. The WF attack experiments can be built based on two different scenarios: closed world and open world. The closed world model is used when complete information is available. The assumption in a closed world model is that an attacker knows the metadata information for a list of websites. The website visited by a victim is in the list known by the attacker. It is a strong assumption which is used to simplify the threat model, implementation of the experiment, and evaluation of the success of the attack. Since the closed world scenario is the more basic model, most research work [1, 3–5, 10–13, 17, 19, 20, 23] include an analysis of results in this closed world model.

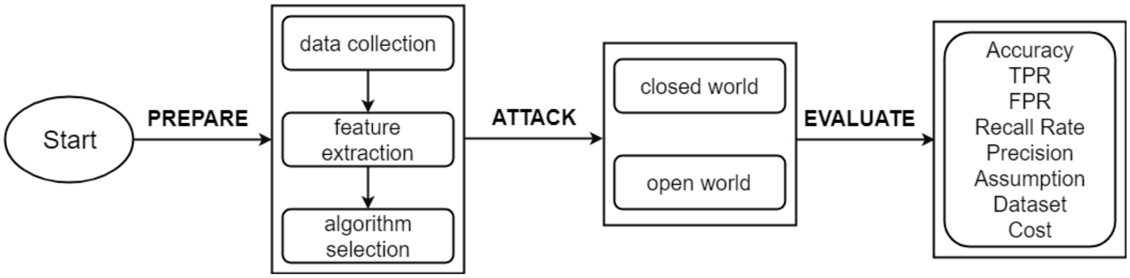

Figure 2: Steps of launching and evaluating a website fingerprinting attack.

In an open world model, a website being fingerprinted can be either from the list or not in the list. The attacker keeps track of a small list of monitored websites. Once a website fingerprint is obtained, the attacker attempts to determine if that website is part of the list of monitored websites or not. More recent research work [4, 5, 8, 12, 14–17, 22–24] deployed their website fingerprinting experiments under the open world model and identified whether a website is from the list of monitored sites.

- **Dataset**. Based on the foreground dataset of RND-WWW from [16], our experiments in Section 3 , Section 4, and Section 5 randomly pick 100 website records which contain 40 instances for each website from the original dataset. Each instance is a trace containing the timestamped incoming and outgoing packets’ size in chronological sequence. Incoming packets are marked with a positive sign, while outgoing packets are marked with a negative sign.
- **Hidden Markov Model**. The Hidden Markov Model (HMM) is a Markov process with unobserved states. It is a statistical tool to model sequences that can be characterized by a process from a generated observable sequence [2]. Based on some training data, the HMM generates the probabilities of the states in the dataset. The parameters of a HMM are of two types: transition probabilities and emission probabilities. The transition probability indicates the probability that a state changes to another state and the emission probability is the probability of an observation within a state. The transition matrix and emission matrix store the transition probability and emission probability of each state respectively.
- **Classification of single-page and two-page traces**. An approach was developed to distinguish traces between one-page trace and two-page traces in [24]. The authors employed k-NN binary classification and trained on two classes: a class of two-page traces (a network trace consisting of two webpages), and a class of single-page traces (a network trace consisting of only one webpage). The classification accuracy is 97%. Based on their results, we assume it is capable to identify a trace with single page or two pages.
- **Related work**. The practicality of the WF attack has been discussed previously. A critical evaluation of WF attacks [12] pointed out the common assumptions of previous work and limitation of datasets and single page visits. The CUMUL algorithm [16] was developed with an Internet scale dataset and achieved an accuracy of more than 90%. [7] investigated the assumption of the victim and attacker visiting the webpages under the same conditions such as browsers and devices. The assumption of single page visits has been explored in [24, 25]. The algorithm in [24] achieves more than 90% accuracy in distinguishing between one-page trace and two-pages traces that are positive-time separated. However, in the following steps of finding the split point, the algorithm has many limitations. First of all, the accuracy is low in the two cases we consider. When two pages are zero-time separated (continuous visits), the accuracy to find the split point, that is, where the end of one website trace ends and the second website trace begins, is around 63%. When two pages are negative time separated (overlapping visits), the accuracy to find the split point falls to 32%. [25] improves the accuracy to find the split point, however, they can only predict the first webpage in a two-page visit. Another limitation of [24] and [25] is that they could not directly predict the websites in the network traffic trace recorded. They attempted to find the split point first and used previous WF approaches for predicting the websites, which leads to a higher cost in time. In our work, we develop two new algorithms to eliminate these limitations.

3 ANALYSIS OF CONTINUOUS TRACES

Notation	Definition
n_{wb}	the number of websites
n_{unique}	the number of packets with unique sizes in all website traces
$n_{p_{As}}$	the number of a packet size p in website A {*start*} state
$n_{total_{As}}$	the total number of packets in website A {*start*} *state*
s_{block}	the size of each block
p_{final}	a matrix of the probabilities of each packet belonging to each class/website
l_{trace}	the length of a trace

Table 1: Notations used in our algorithm.

In this section, we introduce our algorithm based on Hidden Markov Model to detect two continuous websites with zero-time

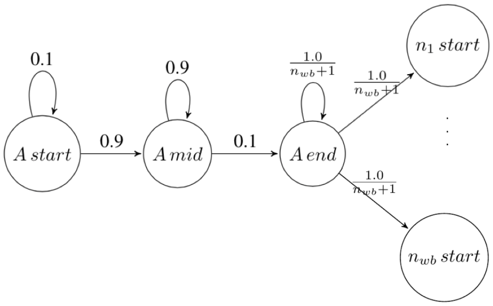

Figure 3: State transition of website A.

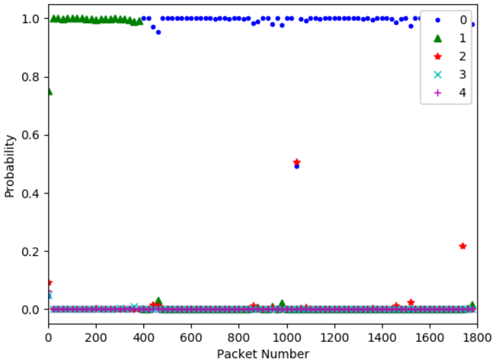

Figure 4: Probability of each packet belonging to each website (the sum of three states for each website) obtained from the HMM model (for clarity, every 20 data point is plotted. Best viewed in color).

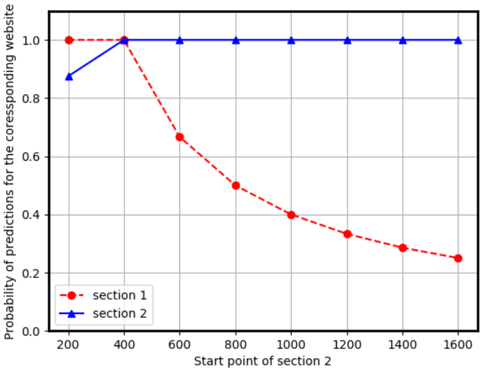

Figure 5: Probability of predictions for the corresponding website (probability of website1 belonging to section1 and probability of website0 belonging to section2) when moving the split point between section 1 and section 2 from left to right. See Figure 4 for the actual predictions.

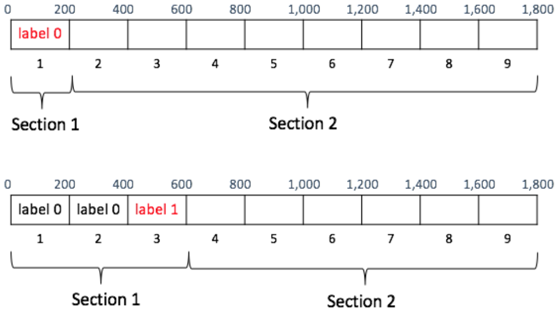

Figure 6: Example when labeling block 1 and block 3.

separated. We describe the details of the algorithm first, followed by the experiments and evaluations of this algorithm. The notations used in the algorithm are introduced in Table 1.

3.1 Algorithm description

Our proposed algorithm can be divided into two steps: 1) Apply Hidden Markov Model to get the probability matrix, 2) Label each block based on the probability matrix and pick split point based on labels.

Step 1: Apply Hidden Markov Model to network traffic trace and obtain the probability of each class (website) that each packet belongs to (probability matrix).

To form states, we split each packet trace into three parts: 1) *start* which is first 20 packets in the network traffic trace, 2) *middle* which is the collection of packets between *start* and *end*, and 3) *end* which is last 20 packets in the network traffic trace. We then build our transition and emission matrices. The dimension of the transition matrix would be $(3 \times n_{wb})^2$.

We use website A as an example. Assume the length of a trace is l_{trace}; this is the number of packets in website A. Figure 3 shows the state transition within website A. For a packet from website A, it has $\frac{20}{l_{trace}}$ probability to belong to A's start state, $1 - \frac{40}{l_{trace}}$ probability to belong to A's middle state and $\frac{20}{l_{trace}}$ probability to belong to A's end state. If a packet in A is in the *start* state, then for the next packet it has a $\frac{20/l_{trace}}{1-(40/l_{trace})+(20/l_{trace})}$ probability to stay in its current state and $\frac{1-(40/l_{trace})}{1-(40/l_{trace})+(20/l_{trace})}$ probability to change to the *middle* state. From analysis of the dataset, we find that $\frac{20}{l_{trace}} = 0.9$ or 9%, thus we set $\frac{20/l_{trace}}{1-(40/l_{trace})+(20/l_{trace})}$ as 10% and $\frac{1-(40/l_{trace})}{1-(40/l_{trace})+(20/l_{trace})}$ as 90%. In a similar way, if A is currently in the *middle* state, then the next packet could stay in the *middle* state with 90% probability or change to the *end* state with a 10% probability. When the packet is in the *end* state, we set that A has an equal probability of $\frac{1.0}{n_{wb}+1}$ to stay in the *end* state or change to any other website's *start* state. For the emission matrix, the dimension is $(3 \times n_{wb}) \times n_{unique}$ where n_{unique} is the number of unique length of packets in all website traces, where 3 indicates the three states (*start*, *middle*, and *end*) for each website. For a packet size P in website A in the *start* state, $n_{p_{As}}$, the total number of packets in A *start* state is $n_{total_{As}}$. The emission probability for P in A's start state would then be $\frac{n_{p_{As}}}{n_{total_{As}}}$. After applying forward and backward propagation, we obtain the probabilities of each packet belonging to each class/website as p_{final}. For simplicity, we show the prediction for five websites in Figure 4. The split point happens at packet number 389 and the network traffic trace is composed of website 1 followed by website 0. The different colors of the lines indicate different websites. A website's different states are shown in the same color. From the figure, we can see that the algorithm predicts website 1 with a probability higher than any other website until about the split point where the algorithm predicts website 0 with the highest probability.

Step 2: Find the split point.

The split point needs to be identified automatically. The main idea of this step is to traverse each packet in a trace from left to right as a split point and measure the probability of the website in two sections, section 1 and section 2, split by the split point. Based on the example in Figure 4, Figure 5 shows the trend of the ratio in section 1 and section 2. By moving the split point from left to right, the probability of predicted website (website 1) in section 1 drops while the probability of predicted website (website 0) increases in section 2. The split point 400 occurs at the intersection of the two lines in Figure 5.

Instead of analyzing each packet, we decide to extract features from blocks to reduce the processing time. We divide the whole trace into several blocks with multiple packets; the size of each block is s_{block}. Assume the length of a trace is l_{trace}, then a trace is split into $n_{block} = l_{trace}/s_{block}$ blocks. We name the block from left to right as block 1, block 2, ..., block n. And we assume the split point is at the end of one of the blocks.

First, we label each block to indicate whether the block is before or after the split point. When labeling block m, we consider block 1 to block m to be *section 1* and block $m + 1$ to block n as *section 2.* The block m is labeled based on the ratio r_{s1} of the number of packets belonging to website X in *section 1* and the ratio r_{s2} of the number of packets belonging to website Y in *section 2*, where X and Y indicate the website with the highest ratio in *section 1* and *section 2* respectively. We set a ratio bottom line t_{block} for r_{s1} and r_{s2}. When labeling the block, we use 0 to represent the block is before the split point and 1 to represent the block is after the split point. If $r_{s1} > t_{block}$, which indicates more than t_{block} section1 is composed of website X , and label the block as 0. If $r_{s1} < t_{block}$ and $r_{s2} > t_{block}$, label the block as 1. If $r_{s1} < t_{block}$ and $r_{s2} < t_{block}$, it indicates that this block does not provide valid information, then this block won't be labeled and recorded. When labeling every block, record the end point of each block as a block index into *point_list*.

We use the example in Figure 4 to illustrate the algorithm behind the labeling process. Figure 5 and Figure 6 are based on this example. The trace contains 1,800 packets and is divided into 9 blocks; the size of each block is 200 packets. The split point between section 1 and section 2 moved from 200 to 1600.

We only list the process when labeling block 1 and block 3 as an example. For a packet, we use the website with the highest probability as the prediction for the packet. In this example we set t_{block} as 95%. When labeling block 1, section 1 contains block 1 and section 2 is from block 2 to block 9; split point is 200. From Figure 4 we can see that the predicted website in section 1 is website 1 and from Figure 5, the confidence of this prediction is close to 1, which means, r_{s1} is close to 100% thus $r_{s1} > 95\%$, then we label block 1 as 0, meaning block 1 is before the split point. For labeling block 3, section 1 is composed of block 1, block 2 and block 3, and section 2 is from block 4 to block 9. For the first 600 packets in section 1, the probability of website 1 is 67% which is less than 95%, then we label block 3 as 1 representing block 3 is after the split point. Under the perfect condition, the split point should be at the point when r_{s1} and r_{s2} meet which is in between 0 and 1 in the label list.

After labeling each block, we pick the split point based on the sequence of labels. The expected list of labels is $\{0, 0, ..., 0, 1, 1, ..., 1\}$. However, when labeling block n, if none of the highest ratio in section 1 and section 2 achieve the threshold t_{block}, the label of this block will be missed. We propose an algorithm to find the split point and is able to handle all these situations. The two main cases are classified by whether 1 is in the *label_list*.

- **label_list contains 1s.** If $label\_list = (0, 0...0, 1, 1...1)$, that is the format we expect. 0 represents the block is before the split point and 1 is the opposite. Then the split point is after the last block labeled with 0.
  If $label\_list = (1, 1, 1...1, 0, 0.., 0, 1, 1, ..., 1)$, it shows that there is some noise at the beginning of the trace as well as some at the end of the trace. However it does not affect the process to find the split point since we only focus on the changes in the trace. We will still assume the split point is after the block with last 0.
- **label_list contains 0s only.** The algorithm will check if enough information is obtained first before making the decision. If blocks are continuously labeled from the first to last block, then we assume that pattern for the probabilities of the first website is clear and return the point after the last labeled block as the split point. This means that the last block is section 2. Otherwise, the backup algorithm will be applied.

The pseudo code of the algorithm is outlined in Algorithm 1.

The main idea of the backup algorithm is to find the split point when the average of the highest ratio of predicted websites in section 1 and section 2 is higher than any other point. Assume that $point\_list = (1\ s_{block}, 2\ s_{block}, ..., n\ s_{block})$, for split point $i - s_{block}$, where $i = 1, ..., n$. The two sections split by this point i are called section 1 and section 2. We denote the percentage of packets belonging in section 1 as $r1_i$ (that is, these packets are correctly marked in the correct section) and the percentage of packets belonging in section 2 as $r2_i$. The average ratio at point $i - s_{block}$ is $avg(r1_i, r2_i)$ – the point with the highest average ratio among all points in the *point_list* is considered as the split point. The backup algorithm is rarely called in our simulations.

3.2 Results for Finding Split Point

The values of s_{block} and t_{block} are selected as 200 and 95%. We used the dataset foreground RND_WWW and CUMUL features from [16] and randomly picked 100 distinct websites with 40 instances each from the dataset. For each website, 20 instances are applied in training and the other 20 are used for testing. Training dataset is then composed of 100 websites with 20 instances each. In order

Algorithm 1 Main Algorithm to Find Split Point

```
procedure GET_CHANGEPOINT($point\_list$, $label\_list$)
    if $point\_list$ is not $empty$ then
        if 1 is in $label\_list$ then
            if $label\_list[0]$ is 0 then                              ▷ $label\_list = (0, 0...0, 1, 1...1)$
                Return $point\ after\ last$ 0
            else                                                       ▷ $label\_list = (1, 1...1, 0, 0...0, 1, 1...1)$
                if $label\_list[0]$ is 1 then
                    Return $point\ after\ last$ 0
                else
                    Return $backup\ algorithm$
                end if
            end if
        else                                                           ▷ $label\_list$ contains 0 only
            if $length(point\_list)$ is 1 then
                Return $backup\ algorithm$
            end if
            if $point\_list[0]$ is $s_{block}$ and $point\_list = (1\ s_{block}, 2\ s_{block}, ..., n\ s_{block})$ then
                Return $point\_list[n-1]$                              ▷ return last point in the $point\_list$
            else
                Return $point\_list[0]$
            end if
        end if
    else                                                               ▷ $point\_list$ is empty
        Return $backup\ algorithm$
    end if
end procedure
```

to simulate the process of visiting one website after another, we picked two websites randomly from the testing set 200 times and concatenated their network traffic trace to form the test set. Since each website for the testing set has 20 instances, there are 4, 000 traces in total in the testing dataset.

We removed the packets with size of Maximum Transmission Unit (MTU) to improve the accuracy. We also consider another threshold in addition to p_{final}, that is, if the highest probability of a packet belonging to every class is lower than the threshold $t_{original}$, where $t_{original}$ is set to 0.8 in the experiment, then that packet will be ignored. A few other thresholds were chosen and it was found that 0.8 gave the best result. Thus we only consider the predictions with high probability. We found that by removing MTU and adding this new threshold value, the accuracy is increased.

We analyze the accuracy of the split point from two metrics: 1) the related deviance of the predicted split point from the real split point and 2) the accuracy of the split point. The related deviance is calculated by $abs(predicted\_point - real\_point)/l_{trace}$. The lowest average related deviance we obtained when testing on 10 and 100 websites are 0.154 and 0.16 respectively, which means the performance of the algorithm is stable when increasing the number of websites. If the predicted split point is before the real split point, the first website loses partial data at the end, and the second website receives extra data at the beginning. Figure 7 shows the decrease in prediction accuracy under a closed world setting. The test dataset is composed of 12 parts; each part contains 100 websites and 20 instances of each website. The first 6 parts consist of cutting 30%/20%/10% traces at the beginning or end of each trace, The second 6 parts are composed of adding 30%/20%/10% traces at the beginning or end of each trace. It shows the effect on prediction accuracy for the first and second website in a continuous trace when the predicted split point is before or after the the real split point. For two continuous websites, the error on the split point has a bigger effect on the second website. A 0.16 related deviation means the average split point is between 0.84 to 1.16 on the x-axis in Fig 7. In this area, it can detect the first website with a decreased accuracy of 15%. Since the original accuracy in detecting one website with website fingerprinting using the k-NN algorithm is about 90% (from figure 5), the accuracy to predict the first website is thus around 90% − 15% = 75%. However, for the second website, the accuracy is lower. To calculate the accuracy of the split point, we consider that the prediction is correct if the block/point prediction is closest to the real split point. The number of points/blocks is decided by l_{trace}/s_{block}, where s_{block} is selected as 200. For example, if the length of the continuous traces is 3, 000 with the split point at packet number 425, and $point\_list = \{200, 400, 600, 800, ..., 2800\}$, we consider the prediction is correct if the predicted split point is 400. Among 4, 000 test traces, 3, 200 of them are predicted with the correct split point. The split point accuracy is thus 80%.

3.3 Results for Website Prediction

The ultimate goal of WF attack is to predict visited websites. The advantage of this algorithm is that it can detect the website directly after finding the split point. We still use $t_{original}$ to filter packets first and assume the packet belongs to website *A* if A has the highest probability among all websites (known from p_{final}). Then we

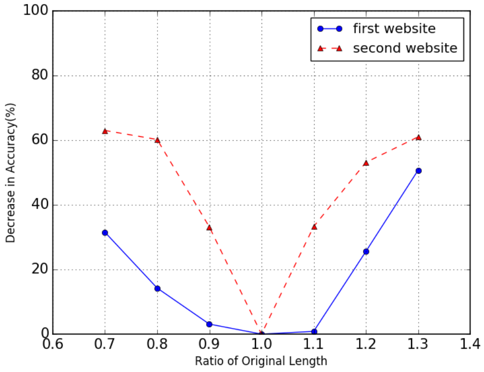

Figure 7: Decrease on prediction accuracy of the first and second website when the predicted split point is not accurate.

calculate the percentage of each website before and after the split point. The website with the highest percentage is the predicted website. We trained on 100 websites and tested on 4, 000 instances. When considering websites with the top three highest probabilities, the accuracy for the first website is 70.2% and the accuracy for the second website is 69.2%.

3.4 Summary

In summary, our "splitting" algorithm has three distinct advantages over [24]. First, it doesn't require new training data, only based on original website traces. Second, it has a higher accuracy of 80% in detecting split point compared to 63%. Third, [24] didn't predict websites for real after finding the split point. From their description, they will reuse previous WF attack approach on two split sections. However, in our algorithm, websites can be predicted directly after finding the split point from the probability matrix and predicted split point.

4 ANALYSIS OF OVERLAPPING TRACES

4.1 Motivation

This section provides an overview of the design of our experiments and a description of our website fingerprinting attack when considering the situations of two overlapping traces (webpages that are negative-time separated). This means that a victim visits a second webpage while the first webpage is still loading. It's not realistic to assume that a user visits only one webpage at a time. However, only one previous paper [24] has looked at overlapping website visits. Figure 8 illustrates two overlapping traces. Trace A belongs to website A and Trace B is from website B. The size of the overlap can vary. We focus on predicting both website A and website B. In previous work, the prediction accuracy of classifying websites based on features like packet sizes and number of packets is high at around 90%. Figure 9 shows the accuracy of the k-NN algorithm when predicting traces with overlapped packets. It can be seen that the accuracy decreases significantly from 89.89% to 22.80% with 5% overlapped packets and to 19.29% with 10% overlapped packets. Thus, overlapping traces have a big impact on prediction accuracy.

In fact, visiting a webpage at the same time as another webpage can be used as a defense to mitigate website fingerprinting attacks because it generates "noise". We, thus, propose a new "sectioning" algorithm that can still accurately perform website fingerprinting attack on overlapped website visits.

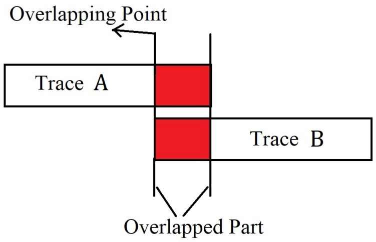

Figure 8: Two website traces A and B overlap.

4.2 Sectioning Algorithm

We now present the design of our proposed "sectioning" algorithm. Instead of treating a traffic trace as a whole, we split the trace into a certain number of sections and perform website prediction on each section. The intuition behind why sectioning will help improve accuracy is that the overlapped parts will only appear in some sections of the trace and other sections will not be disturbed. We also hypothesize that most sections of the trace will not be disturbed. This allows us to perform a majority voting on all the sections to decide which website is being visited.

Figure 10 shows the key parts of our sectioning algorithm: partitioning and majority voting.

1) Partitioning an instance into *n* sections: Partitioning each instance into sections is the most important part of our algorithm. Each trace, whether for training set or testing set, will be partitioned into *n* sections. If $n = 1$ section, this means there is one section and this is what previous work has looked at; this is the base case. Each

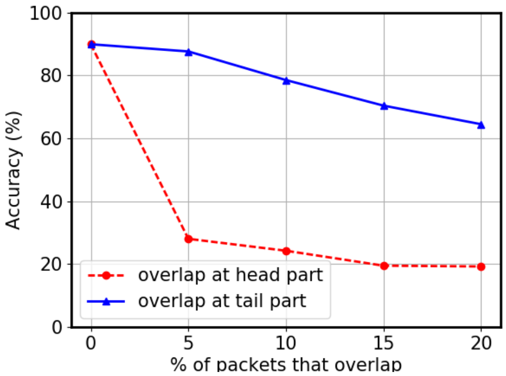

Figure 9: Prediction accuracy as more packets overlap in the two traces.

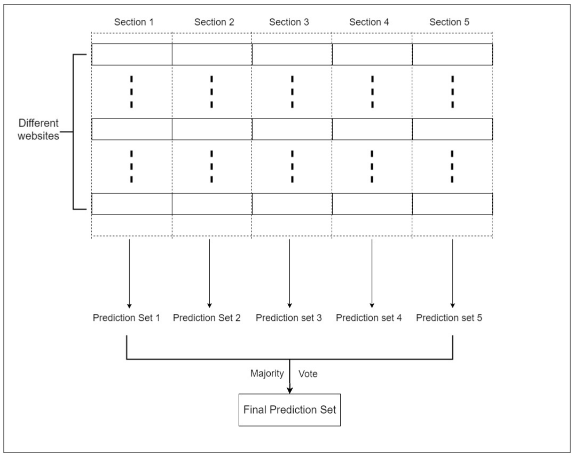

Figure 10: Outline of sectioning algorithm.

section will be evenly split by two methods: a) number of packets; b) time duration of a trace.

1a) sectioning by number of packets: If a trace has 1,000 packets and will be partitioned into 10 sections, then each section will contain 100 packets.

1b) sectioning by time duration: If the duration of a trace is 10 seconds, when partitioning it into 10 sections, then interval of each section will be 1 second. The sections with overlapping traces will clearly have more packets, but the number of sections stays the same with regards to the training set.

2) Perform majority voting: As Figure 10 shows, the last step of our algorithm is to perform majority voting. The purpose of sectioning is to reduce the interference in prediction caused by the overlapped packets, that is, any incorrect predictions made due to overlapped packets will be ignored if the majority of the trace (or sections) is not affected (overlapped). We already have the predictions for each section of each trace. To predict the website for a trace, majority voting is performed on the n sections of that trace to determine the predicted website. If there is no clear majority, any of the highest number of predictions is chosen. For example, like the overlapped trace B in Figure 8, a trace of website B is partitioned into 5 sections. Suppose first 2 out of these 5 sections contain overlapped packets from another trace of website A. The prediction for the first section is website A while the prediction for the second section is website B. Since the remaining 3 sections are unaffected, the predictions are website B. In this case, website B received 4 predictions while website A received 1 prediction. Using majority voting, this trace will be classified as website B.

4.3 Experiment Setup

Figure 11 shows our sectioning algorithm. The steps are as follows: 1) split dataset into training and testing sets (Figure 11(a)); 2) Insert certain amount of packets randomly from another website into the trace of each instance of testing sets – this forms the overlapped traces (Figure 11(a)); 3) Partition into n sections for both training and testing sets accordingly (Figure 11(b)); 4) Apply machine learning classifier (for example, k-NN) to each section ((Figure 11(c)); 5) A majority vote will be performed for the predictions from the different sections (Figure 11(d)); 6) Repeat to do 10-fold cross validation.

We detail each step next.

1) Dataset: As mentioned before, we randomly chose $n = 100$ websites and $k = 40$ instances per website from the RND-WWW dataset and CUMUL features from [16]. Our first step is to split instances of each website into training and testing set under a 10-fold cross validation. 10% of instances are in testing set, the rest are in the training set. This means that 36 of 40 instances will be treated as training set data for each website. We repeat each experiment 10 times, each time choosing a random 36 instances for training.

2) Overlapped traces simulation: An overlapping visit means visiting one website while visiting another website, so that it is hard to tell which website the packet trace belongs to. As Figure 8 shows, website B has an overlap at the beginning with website A and website A has an overlap at the end with website B. We attempt to predict both websites using the sectioning algorithm. Wang's work [24] showed that it's possible to find the split point which is the end of website A and the start of website B in overlapped traces. We will outline our improved algorithm in Section 4.5. Figure 12 shows that for our simulation, we insert-merge packets to the beginning of a website trace when predicting website B, and insert-merge packets to the end of a website trace when predicting website A. To simulate overlapped traffic traces, we add packets from one traffic trace (instance) of another website A to the beginning of website B or vice versa. This is not a prepend method, but instead a merging is performed. Each instance contains packets' sizes along with the time stamp for each packet. We take the last few packets of website A and reset the timestamp of that first packet to be zero so that the last few packets of website A are merged into the beginning of website B. We also simulated different overlapping fractions from 5% to 20%; this means we obtained the last 5% of packets from website A's network trace and merged with the beginning of the trace for website B. Also, we do the same procedure to the end of the trace for website A.

As an example of inserting A to the beginning of B, all packets are of the format $< time >:< packetsize >$. Let's say the last two packets of website A are $2045 : 1040$ and $2100 : 500$ and the first two packets of website B are $50 : 412$ and $70 : 250$. Resetting the timestamp of the first packet from website A to zero, the packets are then $0 : 1040$ and $55 : 500$. Merging both set of packets together produces a new network trace with packets $0 : 1040$, $50 : 412$, $55 : 500$, and $70 : 250$.

3) Sectioning: We emphasize that the training sets are the original traces. Only the testing datasets are "overlapped". We cross-validated the training set to obtain a reasonable model. Every trace, in both training and testing sets, will be partitioned into n sections, where $n = 1, 4, 5, 8, 10$. Each section is then parsed using the CUMUL features, similar to [16].

4) Run training/testing: After we have each trace split into n sections, 90% of instances with same section number will be used as the training set. We test the trained classifier on the remaining 10% of instances with the same section number. For classifier algorithm, we use the k-nearest neighbor (k-NN) algorithm. Since each section is trained and tested independently of other sections, the result is n predictions for the n sections. The n predictions can be the same

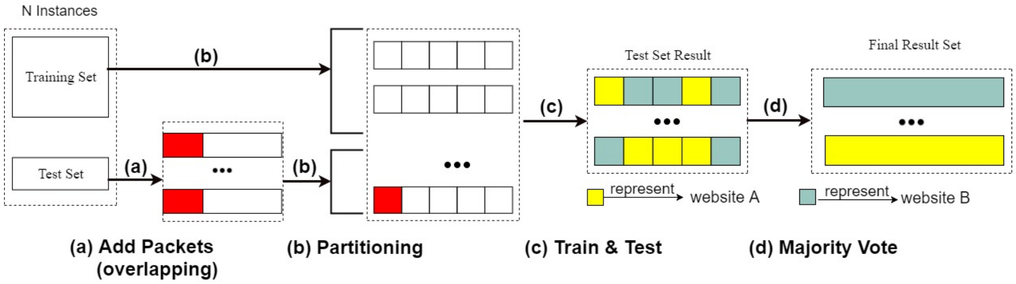

Figure 11: Overview of the sectioning algorithm.

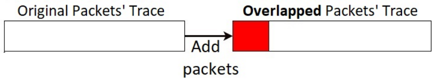

Figure 12: Simulate overlapping: add packets to the beginning of trace.

website or different websites. Figure 10 shows this procedure; in the figure, $n = 5$ sections, thus there are 5 prediction sets accordingly.

5) Perform majority voting: Finally, we perform a majority voting on predictions obtained from different sections, to get a final prediction of which website the trace belongs to.

4.4 Results

a) Sectioning by number of packets: Figure 13 and Figure 14 show the accuracy result in correctly predicting websites A and B, when using sectioning by number of packets. The % of overlapping packets and the number of sections are also varied in the figures. Figure 13 shows the prediction accuracy for website A. With the base case (1 section), the accuracy is comparable with the no overlap case (89%). Sectioning by number of packets has a slightly decrease from 87.61% to 77.13% when the number of sections is 4 and 5% overlap. From Figure 14, it can be seen that even with 5% overlapping packets, the prediction accuracy for website B with 1 section is 22.80%. When the number of sections increases to 4, the accuracy also increases to 64.95%. This indicates that sectioning helps in mitigating the impact of the overlap. Increasing the number of sections further from 4 to 10 slightly increases the prediction accuracy and peaks at 67.92% with 8 sections. As the % of overlap increases from 5% to 20%, the accuracy decreases as expected. When there is 20% overlapping packets, the accuracy for 1 section decreases further to 15.85%. As the number of sections is increased to 4, the accuracy is 39.06%. With 10 sections, the accuracy is 48.47%. This is expected as the overlapping part becomes bigger, it affects more sections, which makes prediction of the whole website harder. As shown in [9] and later in Section 5, the difference in prediction accuracy in predicting websites A and B is because the beginning of a trace is more important than the end when predicting a website.

b) Sectioning by time duration: Figure 15 and Figure 16 show the accuracy result in correctly predicting websites A and B when using sectioning by time duration. Figure 15 shows that the accuracy decreases from 83.35% with 1 section to 75.70% with 5 sections with 5% overlap. However, as the % of overlap increases to over 10%, the accuracy with 5 sections is higher than with 1 section. For example, when the % of overlap is 20%, the accuracy for 1 section

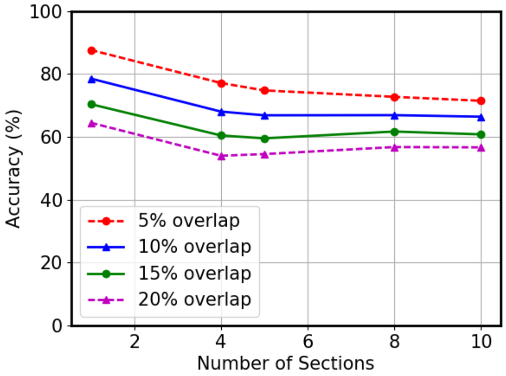

Figure 13: Prediction accuracy of website A with varying number of sections and overlap %, using a) sectioning by number of packets.

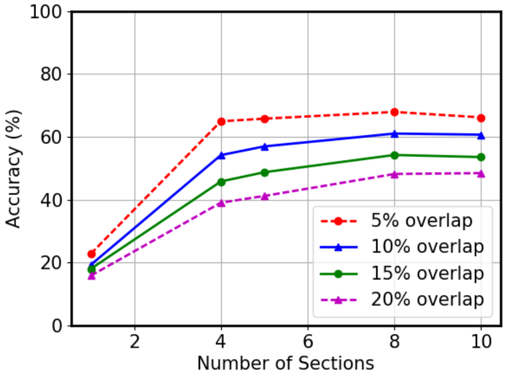

Figure 14: Prediction accuracy of website B with varying number of sections and overlap %, using b) sectioning by number of packets.

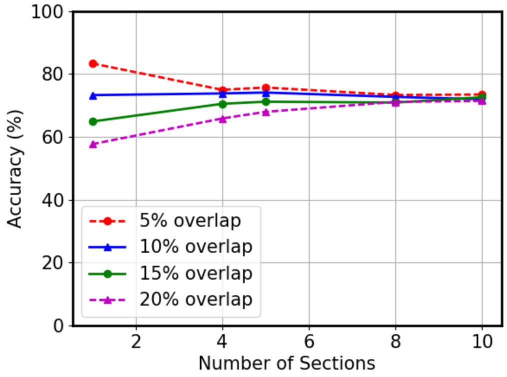

Figure 15: Prediction accuracy of website A with varying number of sections and overlap %, using b) sectioning by time duration.

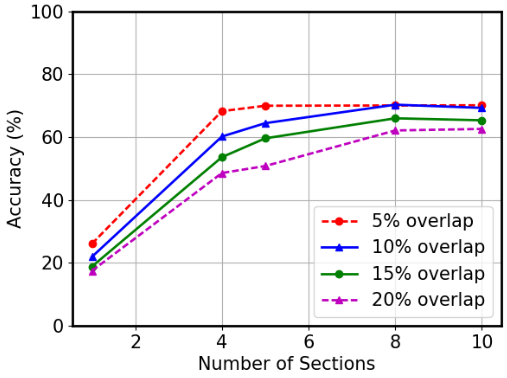

Figure 16: Prediction accuracy of website B with varying number of sections and overlap %, using b) sectioning by time duration.

decreases to 57.67%, and the accuracy for 10 sections is 71.44%. This shows that unlike sectioning by number of packets, the sectioning algorithm improves the accuracy when predicting website A. From Figure 16, it can be seen that with 5% overlapping packets, the prediction accuracy with 1 section is 26.09%. When the number of sections increases to 4, the accuracy also increases to 68.25%. This indicates that sectioning helps in mitigating the impact of the overlap. Increasing the number of sections further from 4 to 10 slightly increases the prediction accuracy and peaks at 70.11% with 10 sections. As the % of overlap increases from 5% to 20%, the accuracy decreases as expected. When there are 20% overlapping packets, the accuracy for 1 section decreases further to 17.47%. As the number of sections is increased to 4, the accuracy is 48.58%. With 10 sections, the accuracy is 62.59%. This result shows that sectioning by time duration is slightly better than sectioning by number of packets, but the shape of the graphs is similar.

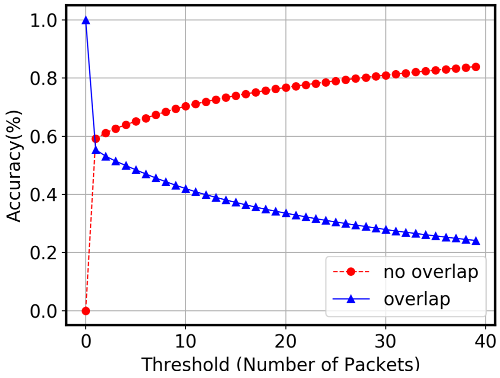

Figure 17: Prediction accuracy of the overlapping parts and non-overlapping parts.

Sectioning by number of packets means the number of packets is the same for each section while sectioning by time duration means the time interval is the same but number of packets could be different for each section. The results show that sectioning by time duration is better than sectioning by number of packets for predicting both websites A and B (first and second websites).

4.5 Predicting Overlapping Point

Previous work [24] showed that the accuracy to find the split point in overlapped trace is 32%. In Section 3, we showed that the split point of two continuous traces can be accurately found. In this section, we attempt to improve the prediction accuracy on the start and end of where the two webpages overlap.

Our method works as follows. To determine if there is an overlap, we hypothesize that the number of packets during an overlap will be higher than when there is no overlap, since there will be the network traffic from two webpages instead of one. We divided the time into bins, so that we have discrete bins. For each bin, we then counted the number of packets. If the number of packets in a bin is higher than a threshold, we consider this as an overlap part. In all our overlapped traces, we know the ground truth, so we can calculate the accuracy of our prediction.

We vary the size of the bin from 1 millisecond to 10 seconds. Figure 17 shows the prediction accuracy for the overlap and non-overlap part when the bin size was 500 milliseconds. The accuracy is around 60% when predicting either the overlap or no-overlap part. Increasing the bin size shifts the graph to the right. We also considered the size of all the packets in each bin as a predictor and we obtained a similar result.

4.6 Summary

We proposed a "sectioning" algorithm that can achieve better accuracy (around 70% when predicting either the first or second website) than previous methods (57% when predicting first website and 26% when predicting second website) when there is some overlap of two websites. We also showed that the exact point where the overlap starts and stops can be reasonably predicted. The overlap part can thus be effectively ignored and an effective website fingerprinting attack performed.

5 ANALYSIS OF PARTIAL TRACES

5.1 Motivation

This section shows the impact of the possibility of partial traces (only part of the website traffic have been captured) on website fingerprinting attacks. This could happen when a victim visits one website and close the browser before the download is complete or the adversary was only able to record part of the trace (either the beginning or the end).

We assume there is only one website in the traffic trace. However, the adversary is only able to record a fraction n of the traffic trace. When $n = 100\%$, then this is the assumption taken from previous work that an attacker is able to capture entire traces for all websites. We vary n from 80% to 100% of the traffic trace from either the beginning or the end. The adversary can observe the first $n\%$ of a website's traffic trace before some interference occurs, or the last $n\%$ of a website's traffic trace. Figure 18 shows the result of our experiments. When the whole trace is recorded, the accuracy is at 89.9%. When 10% of the packets are missed at the end of the trace, then the accuracy goes down to 64.1%. However, when 10% of the packets are missed at the beginning of the trace, then the accuracy goes down to 15.05%. It can be seen that capturing the first $n\%$ of a website's trace is more important than the last $n\%$. This could be due to more outgoing requests from the client to the server which makes fingerprinting easier and more identifiable. This result confirms that of [9]. The figure also shows that as the percentage of the trace available decreases, the accuracy decreases significantly.

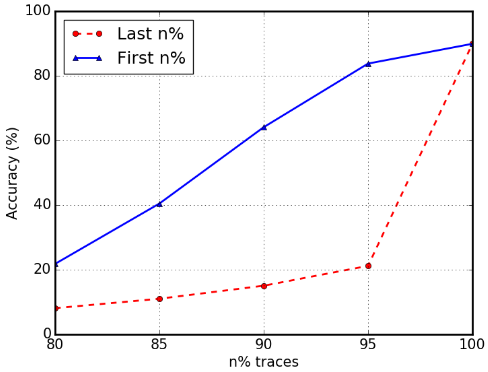

Figure 18: Accuracy of website fingerprinting when observing different percentages of network traffic traces.

5.2 Sectioning Algorithm on Partial Traces

Since we have shown that our sectioning algorithm can still provide a high prediction accuracy for overlapped traces, we now apply the same algorithm to partial traces. The hypothesis is the same: some sections will be missing, but this should not affect the other sections. We used the sectioning algorithm by time duration as this has been shown to provide a better prediction accuracy. We also used the same dataset as before. The training datasets consist of the whole network traces. The testing datasets consist of the remaining instances with missing packets either at the beginning or at the end. For each testing dataset, we remove the first $n\%$ of packets either from the beginning or from the end.

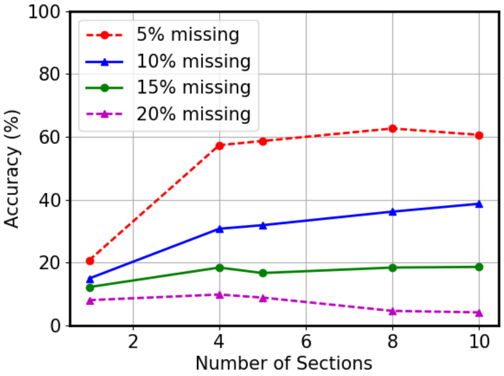

Figure 19: Prediction accuracy when varying the number of sections and the % of missing packets from the beginning.

5.3 Results

Figure 19 and Figure 20 show the accuracy in correctly predicting websites based on partial traces, when varying the % of missing packets and the number of sections. The base case is with 1 section, which means no sectioning algorithm applied. From Figure 19, it can be seen that with 5% missing packets from the beginning of a trace, the prediction accuracy with 1 section is 20.76%. When the number of sections increases to 4, the accuracy increases to 57.34%. This indicates that sectioning helps in mitigating the impact of the missing packets. Increasing the number of sections further from 4 to 10 slightly increases the prediction accuracy and peaks at 62.66% with 8 sections. As the % of missing packets increases from 5% to 20%, the accuracy decreases. This is expected since with more missing packets, it affects more sections, which makes prediction of the whole website harder. By using our sectioning algorithm, the accuracy improves significantly from the base case.

Figure 20 shows the accuracy of correctly predicting websites based on partial traces with packets missing from the end. When missing 5% and 10% packets from the end of a trace, the prediction accuracy with 1 section is 79.02% and 58.80% respectively. With 10 section, the accuracy is 64.78% and 53.92% respectively. It is slightly lower than the base case. However, when the % of missing increases to 15% and 20%, the accuracy with 10 sections is 42.35% and 30.61% compared to the base case 35.92% and 19.49%.

5.4 Summary

We show that our "sectioning" algorithm can also be used for partial traces. It has a better accuracy (62.66%) comparing to previous methods (20.76%) on predicting websites with missing packets at the beginning. Our algorithm achieves similar accuracy with packets

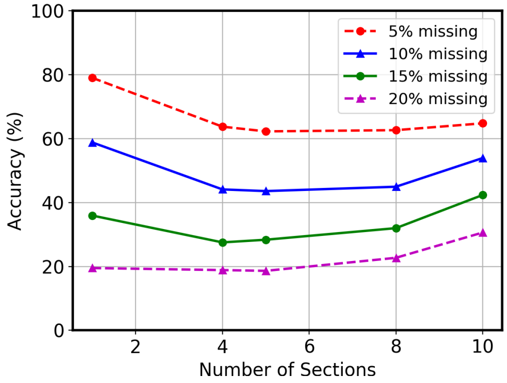

Figure 20: Prediction accuracy when varying the number of sections and the % of missing packets from the end.

missing at the end. In general, this shows that our proposed sectioning algorithm provides a higher or similar prediction accuracy as current algorithms.

6 CONCLUSION AND FUTURE WORK

In this paper, our goal is to address the impracticalities of website fingerprinting attacks and propose solutions to several limitations:

1. We propose a "splitting" algorithm to identify two continuous network traces with an accuracy of 80% in finding the split point of the two traces.
2. We propose a "sectioning" algorithm to improve the accuracy in website prediction of two overlapping traces from 22.80% to 67.9% and partial traces from 20.76% to 62.66%.

For the future work, we will test our algorithm in the open world setting and will consider the scenario when more than two pages are continuous or overlap. Moreover, we have shown some promising results in predicting exactly where two webpages overlap; we plan to investigate this further. We will also run more experiments with a more diverse dataset.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1659645. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. 2006. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Proceedings of the 5th International Conference on Privacy Enhancing Technologies (PET'05)*. Springer-Verlag, Berlin, Heidelberg, 1–11. https://doi.org/10.1007/11767831_1

[2] P. Blunsom. 2004. Hidden Markov Models. http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf

[3] Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society (WPES '14)*. ACM, New York, NY, USA, 121–130. https://doi.org/10.1145/2665943.2665949

[4] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, New York, NY, USA, 227–238. https://doi.org/10.1145/2660267.2660362

[5] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*. ACM, New York, NY, USA, 605–616. https://doi.org/10.1145/2382196.2382260

[6] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium.*

[7] Hasan Faik Alan and Jasleen Kaur. 2019. Client Diversity Factor in HTTPS Webpage Fingerprinting. 279–290. https://doi.org/10.1145/3292006.3300045

[8] Xun Gong, Nikita Borisov, Negar Kiyavash, and Nabil Schear. 2012. Website Detection Using Remote Traffic Analysis. In *Proceedings of the 12th International Conference on Privacy Enhancing Technologies (PETS'12)*. Springer-Verlag, Berlin, Heidelberg, 58–78. https://doi.org/10.1007/978-3-642-31680-7_4

[9] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 1187–1203. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/hayes

[10] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. 2009. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial NaïVe-bayes Classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW '09)*. ACM, New York, NY, USA, 31–42. https://doi.org/10.1145/1655008.1655013

[11] Andrew Hintz. 2003. Fingerprinting Websites Using Traffic Analysis. In *Proceedings of the 2Nd International Conference on Privacy Enhancing Technologies (PET'02)*. Springer-Verlag, Berlin, Heidelberg, 171–178. http://dl.acm.org/citation.cfm?id=1765299.1765312

[12] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. 2014. A Critical Evaluation of Website Fingerprinting Attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, New York, NY, USA, 263–274. https://doi.org/10.1145/2660267.2660368

[13] Marc Liberatore and Brian Neil Levine. 2006. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*. ACM, New York, NY, USA, 255–263. https://doi.org/10.1145/1180405.1180437

[14] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. 2010. *Website Fingerprinting and Identification Using Ordered Feature Sequences.* Springer Berlin Heidelberg, Berlin, Heidelberg, 199–214. https://doi.org/10.1007/978-3-642-15497-3_13

[15] Se Eun Oh, Shuai Li, and Nicholas Hopper. 2017. Fingerprinting Keywords in Search Queries over Tor. *PoPETs* 2017 (2017).

[16] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. 2016. Website Fingerprinting at Internet Scale. In *Proceedings of the 23rd Internet Society (ISOC) Network and Distributed System Security Symposium (NDSS 2016).*

[17] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. 2011. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society (WPES '11)*. ACM, New York, NY, USA, 103–114. https://doi.org/10.1145/2046556.2046570

[18] Tor Metrics Portal. 2017. https://metrics.torproject.org/.

[19] Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard. 2016. Exploiting Data-Usage Statistics for Website Fingerprinting Attacks on Android. In *Proceedings of the 9th ACM Conference on Security &#38; Privacy in Wireless and Mobile Networks (WiSec '16)*. ACM, New York, NY, USA, 49–60. https://doi.org/10.1145/2939918.2939922

[20] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. 2002. Statistical Identification of Encrypted Web Browsing Traffic. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy (SP '02)*. IEEE Computer Society, Washington, DC, USA, 19–. http://dl.acm.org/citation.cfm?id=829514.830535

[21] Tor. 2017. https://www.torproject.org/.

[22] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proceedings of the 23rd USENIX Conference on Security Symposium (SEC'14)*. USENIX Association, Berkeley, CA, USA, 143–157. http://dl.acm.org/citation.cfm?id=2671225.2671235

[23] Tao Wang and Ian Goldberg. 2013. Improved Website Fingerprinting on Tor. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society (WPES '13)*. ACM, New York, NY, USA, 201–212. https://doi.org/10.1145/2517840.2517851

[24] Tao Wang and Ian Goldberg. 2016. On Realistically Attacking Tor with Website Fingerprinting. In *Privacy Enhancing Technologies Symposium (PETS).*

[25] Yixiao Xu, Tao Wang, Qi Li, Qingyuan Gong, Yang Chen, and Yong Jiang. 2018. A Multi-tab Website Fingerprinting Attack. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*. ACM, New York, NY, USA, 327–341. https://doi.org/10.1145/3274694.3274697