PRIVACY ANALYSIS OF ONLINE AND OFFLINE SYSTEMS

By

TAO CHEN

Bachelor of Science in Communication Engineering
China Jiliang University
Hangzhou, Zhejiang
China
2011

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
JULY, 2019

PRIVACY ANALYSIS OF ONLINE AND OFFLINE SYSTEMS

Dissertation Approved:

Dr. Eric Chan-Tin

Dissertation Advisor

Dr. Nohpill Park

Dr. Christopher Crick

Dr. Weihua Sheng

Name: TAO CHEN

Date of Degree: JULY, 2019

Title of Study: PRIVACY ANALYSIS OF ONLINE AND OFFLINE SYSTEMS

Major Field: COMPUTER SCIENCE

Abstract: How to protect people's privacy when our life are banded together with smart devices online and offline? For offline systems like smartphones, we often have a passcode to prevent others accessing to our personal data. Shoulder-surfing attacks to predict the passcode by humans are shown to not be accurate. We thus propose an automated algorithm to accurately predict the passcode entered by a victim on her smartphone by recording the video. Our proposed algorithm is able to predict over 92% of numbers entered in fewer than 75 seconds with training performed once.

For online systems like surfing on Internet, anonymous communications networks like Tor can help encrypting the traffic data to reduce the possibility of losing our privacy. Each Tor client telescopically builds a circuit by choosing three Tor relays and then uses that circuit to connect to a server. The Tor relay selection algorithm makes sure that no two relays with the same /16 IP address or Autonomous System (AS) are chosen. Our objective is to determine the popularity of Tor relays when building circuits. With over 44 vantage points and over 145,000 circuits built, we found that some Tor relays are chosen more often than others. Although a completely balanced selection algorithm is not possible, analysis of our dataset shows that some Tor relays are over 3 times more likely to be chosen than others. An adversary could potentially eavesdrop or correlate more Tor traffic.

Further more, the effectiveness of website fingerprinting (WF) has been shown to have an accuracy of over 90% when using Tor as the anonymity network. The common assumption in previous work is that a victim is visiting one website at a time and has access to the complete network trace of that website. Our main concern about website fingerprinting is its practicality. Victims could visit another website in the middle of visiting one website (overlapping visits). Or an adversary may only get an incomplete network traffic trace. 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%.

TABLE OF CONTENTS

Chapter Page

I Introduction . . . **1**
1.1 Privacy of Offline Systems: Smartphone Passcode Prediction . . . 1
1.2 Privacy of Online Systems: Tor . . . 2
1.2.1 Measuring Tor Relay Popularity . . . 2
1.2.2 Anonymous Networks Website Fingerprinting . . . 3

II Smartphone Passcode Prediction . . . **5**
2.1 Introduction . . . 5
2.2 Related Work . . . 6
2.3 Passcode Prediction . . . 8
2.3.1 Video Recording . . . 9
2.3.2 Extract Frames . . . 9
2.3.3 Identify Region of Interest . . . 10
2.3.4 Detect Contour . . . 11
2.3.5 Rebuild Keypad and Retrieve Finger Contour . . . 12
2.3.6 Prediction Algorithm . . . 14
2.4 Results . . . 16
2.4.1 Data Collection . . . 17
2.4.2 Automated Passcode Prediction . . . 17
2.4.3 Analysis of Failed Predictions . . . 19
2.4.4 Algorithm Processing time . . . 20
2.5 Improvements and Discussions . . . 20
2.6 Summary . . . 22

III Measuring Tor Relay Popularity . . . **23**
3.1 Introduction . . . 23
3.2 Background . . . 25
3.3 Experiment Setup . . . 27
3.3.1 Data Collection . . . 27
3.3.2 Data Analysis . . . 28
3.4 Experimental Results . . . 29
3.4.1 Dataset Overview . . . 29
3.4.2 Metrics Used . . . 31
3.4.3 Analysis of Middle Relays . . . 32
3.4.4 Analysis of Exit Relays . . . 38
3.4.5 Overall . . . 47

Chapter Page

3.5 Related Work . . . 47
3.6 Conclusion . . . 49

IV Anonymous Networks Website Fingerprinting . . . 51
4.1 Introduction . . . 51
4.2 Background . . . 53
4.3 Analysis of Overlapping Traces . . . 55
4.3.1 Motivation . . . 55
4.3.2 Sectioning Algorithm . . . 56
4.3.3 Experiment Setup . . . 59
4.3.4 Results . . . 62
4.3.5 Predicting Overlapping Point . . . 66
4.3.6 Summary . . . 66
4.4 Analysis of Partial Traces . . . 67
4.4.1 Motivation . . . 67
4.4.2 Sectioning Algorithm on Partial Traces . . . 68
4.4.3 Results . . . 69
4.4.4 Summary . . . 71
4.5 Summary . . . 71

V Conclusion and Future Work . . . 72

REFERENCES . . . 74

LIST OF TABLES

Table Page

2.1 Number of correct predictions of the passcode detection algorithm on recorded videos from the left and right side of the victim. 19

2.2 Example of one passcode *2459* and the predictions along with the confidence of our algorithm. 19

2.3 Success rate for each number from both sides. 19

3.1 # of Tor relays, in terms of unique IP addresses, /24 subnet, /16 subnet, /8 subnet and AS number used when collecting data in our experiments over 5 months. 31

3.2 Results of comparing Tor relay nodes' IP address and target websites' IP address in the same circuit to determine if they are in the same subnet prefix or AS number. The total number of circuits is $145,918$. 47

LIST OF FIGURES

Figure Page

2.1 Overview of our attack to detect PIN entered on an observed smartphone. (a) Our attack starts with recording a video of a victim who is entering the passcode. The video consists of multiple frames. (b) TLD is applied to identify the frames where the victim is typing (clicking) on the screen. For a four-digit PIN, this will identify four frames. (c) A classifier is used to identify the Region of Interest (ROI) in each frame. (d) The Line Segments Detector outlines the edges (lines) of the smartphone. (e) Rebuild the numbers of the keypad on the screen from the user's point of view. (f) Apply the skin detector algorithm to filter out the fingers inside the screen and then predict the number the user touched on. 8

2.2 Detecting the contour of the smartphone by extracting the edges from the ROI image and filtering out noise. 12

2.3 Image of the ROI showing the person's hand. The person's skin color is filtered out to obtain the contour of the finger. 13

2.4 Final image where the contour of the smartphone screen, the contour of the finger, and the number keypad can be clearly seen. 13

2.5 Pass line by line to locate the finger position. 13

2.6 Illustration of our prediction algorithm. 17

2.7 Video Frames Count and Duration Distribution. 18

Figure Page

3.1 How Tor works. 3 nodes are selected from running Tor relays 26

3.2 This graph shows the number of running relays that have the flags "Running" and "Stable" assigned by the Tor directory authorities. The graph shows the range from November 2017 to March 2018 which is the duration of our experiments. 30

3.3 Point of view of all machines. The percentage of times a relay has been used as a middle relay for all circuits. This shows the top 30 most-used relays. 32

3.4 Point of view of all machines. The percentage of times a relay is selected as the middle relay in a circuit and that relay's corresponding bandwidth. 33

3.5 Point of view of all machines. The repeated percentage for each relay is chosen as a middle relay. This shows the top 30 most-used relays. 33

3.6 Point of view of all source machines. The percentage of selection of the top 30 most-selected /16 subnet IP addresses for middle relays. . 35

3.7 Percentage of # of middle relays in a /16 subnet. 35

3.8 Point of view of one target destination IP address *151.101.128.81*. The % of selection of the 30 most often selected Tor relays as the middle relays in circuits. 36

3.9 Point of view of one target destination IP address *151.101.128.81*. The percentage of selection of the top 30 most often selected Tor relays as the middle relays in circuits and these relays' corresponding bandwidth. 37

3.10 Point of view of target IP address *151.101.128.81*. The % of the 30 most often used relays being selected as middle relays, grouped by /16 subnets. 37

3.11 Percentage of # of middle relays that is in an AS. 39

Figure Page

3.12 Percentage of an AS shown as Middle relay of all circuits. 39
3.13 Point of view of all machines. The percentage of times a relay has been used as an exit relay for all circuits. This shows the top 30 most-used relays. 40
3.14 Point of view of all machines. The percentage of times a relay is selected as the exit relay in a circuit and that relay's corresponding bandwidth. 40
3.15 Point of view of all machines. The repeated percentage for each relay is chosen as an exit relay. This shows the top 30 most-used relays. . . 41
3.16 Point of view of all source machines. The percentage of selection of the top 30 most-selected /16 subnet IP addresses for exit relays. . . . 41
3.17 Percentage of # of exit relays in a /16 subnet. 43
3.18 Point of view of one target destination IP address 151.101.130.167. The % of selection of the 30 most often selected Tor relays as the exit relays in circuits. 44
3.19 Point of view of one target destination IP address 151.101.130.167. The percentage of selection of the top 30 most often selected Tor relays as the exit relays in circuits and these relays' corresponding bandwidth. 44
3.20 Point of view of target IP address 151.101.130.167. The percentage of the top 30 most often used relays being selected as exit relays, grouped by /16 subnets. 45
3.21 Percentage of # of exit relays that is under an AS. 46
3.22 Percentage of exit relays in an AS amongst all circuits. 46

4.1 Threat Model. 54
4.2 Steps of launching and evaluating a website fingerprinting attack. . . 54
4.3 Two website traces A and B overlap. 56

Figure Page

4.4 Prediction accuracy as more packets overlap in the two traces. 57
4.5 Outline of sectioning algorithm. 58
4.6 Overview of the sectioning algorithm. 60
4.7 Simulate overlapping: add packets to the beginning of trace. 61
4.8 Prediction accuracy of website A with varying number of sections and overlap %, using **a) sectioning by number of packets**. 63
4.9 Prediction accuracy of website B with varying number of sections and overlap %, using **b) sectioning by number of packets**. 64
4.10 Prediction accuracy of website A with varying number of sections and overlap %, using **b) sectioning by time duration**. 64
4.11 Prediction accuracy of website B with varying number of sections and overlap %, using **b) sectioning by time duration**. 65
4.12 Prediction accuracy of the overlapping parts and non-overlapping parts. 67
4.13 Accuracy of website fingerprinting when observing different percentages of network traffic traces. 68
4.14 Prediction accuracy when varying the number of sections and the % of missing packets from the beginning. 70
4.15 Prediction accuracy when varying the number of sections and the % of missing packets from the end. 70

CHAPTER I

Introduction

With wide spread use of smart devices and internet, there are tons of ways to get information or private data related to a person. When you are offline, applications on devices like smartphones store and collect your data when you are using them. When it comes to online, anyone between you and the website your tries to connect may access to your data. Thus, privacy leaking is becoming a challenge to almost everyone. People do not want to expose their identity, information or interests to others.

1.1 Privacy of Offline Systems: Smartphone Passcode Prediction

To protect their sensitive information on devices like smartphones, people generally lock their phones to prevent unauthorized access to their phones. The authentication methods for smartphones range from a four-digit number, a password, to a pattern (for Android devices only). Fingerprint, face, and voice recognition are also available but an alternative authentication such as a passcode is also required. For those people who choose to lock their smartphones, a four-digit PIN or patterns are most commonly used. Are they safe now? Let's consider a common situation. A user needs to access her smartphone in a public setting, such as on a subway or in the park. There are several other users around her that are watching her. This leaves an opportunity for them to shoulder-surf the user entering her passcode and attempt to guess what the passcode entered is. These users can then steal the smartphone and can thus have complete access to all the victim's files and pictures. Further, there are

lots of cameras like CCTV in a building or on the streets. These cameras are always recording and can record users' entering their passcode on their smartphone.

Therefore our research goal on offline is the design and implementation of an online algorithm to accurately predict a smartphone's four-digit passcode using a camera to record the victim entering the passcode [1]. With the correct passcode, we can access to victim's personal data. Hence, users should be more careful or use other methods like random keypads to protect their process of entering passcode.

1.2 Privacy of Online Systems: Tor

For privacy protection in the Internet , anonymous networks, like tor, is one way to protect against a common form of Internet surveillance known as "traffic analysis." [2] Traffic analysis can be used to infer who is talking to whom over a public network. Knowing the source and destination of your Internet traffic allows others to track your behavior and interests. This can impact your checkbook if, for example, an e-commerce site uses price discrimination based on your country or institution of origin. It can even threaten your job and physical safety by revealing who and where you are. For example, if you're travelling abroad and you connect to your employer's computers to check or send mail, you can inadvertently reveal your national origin and professional affiliation to anyone observing the network, even if the connection is encrypted.

1.2.1 Measuring Tor Relay Popularity

The Tor network is run by volunteer relays (over 6,000). Each Tor client telescopically builds a circuit by choosing three Tor relays and then uses that circuit to connect to a server. The Tor relay selection algorithm makes sure that no two relays with the same /16 IP address or Autonomous System (AS) are chosen. We measure Tor at large scale to explore the weak aspects of anonymous system and propose methods

to mitigate website fingerprinting attacks. By measuring Tor, we mean to collect information about the route of a visiting to a website, such as source/destination IPs, tor entry node record, middle node record, exit node record. Our objective is to determine the popularity of Tor relays when building circuits [3]. We explore if there are some relay nodes whichhave a higher chance to be selected in a circuit than other relays.Namely, these relays are more popular. We also look into the /8,/16, /24 subnets of all relay nodes to find out the popular subnets.We then look at the relationship between the bandwidth of relaynodes and the popularity of relay nodes.

1.2.2 Anonymous Networks Website Fingerprinting

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. The effectiveness of website fingerprinting has been shown to have an accuracy of over 90% when using Tor as the anonymity network. The common assumption in previous work is that a victim is visiting one website at a time and has access to the complete network trace of that website. Our main concern about website fingerprinting is its practicality. Victims could visit another website in the middle of visiting one website (overlapping visits). Or an adversary may only get a incomplete network traffic trace. When two website visits are overlapping or part of the network trace is missing (either the beginning or the end), the website fingerprinting accuracy falls dramatically.

Our research focuses on analyzing more realistic traffic data to launch website fingerprinting (WF) attacks [4]. We set up an automatic environment to collect Tor traffic data. The traffic data includes the traffic data with a WF defense tool enabled,

data of visiting two websites with a time overlap and so on. After that, we will apply our proposed algorithm "sectioning" and other machine learning algorithms to find patterns to predict websites.

The rest of this dissertation is organized as follows: Chapter II presents our analysis of predicting smartphone passcode. Chapter III shows the measurement of Tor relay popularity. In Chapter IV, we review website fingerprint attacks with more realistic data and our new algorithm. Finally, Chapter V concludes and provides some avenue for future work.

CHAPTER II

Smartphone Passcode Prediction

2.1 Introduction

As smartphones came to prominence several years ago, it has spread to many aspects of people's daily life. Instead of storing data in devices like laptops or desktops, people now store most of their data, such as pictures, documents, music, personal diaries, financial statements, contacts, etc... on their smartphones. Even if the documents are stored in the cloud, such as on Google Drive and Apple iCloud, access to these services are easily obtained without having to re-authenticate once access to the smartphone is obtained. For this reason, some people decide to lock their smartphones. The authentication methods for smartphones range from a four-digit number, a password, to a pattern (for Android devices only). Fingerprint, face, and voice recognition are also available but an alternative authentication such as a passcode is also required. For those people who choose to lock their smartphones, a four-digit PIN or patterns are most commonly used.

We focus on four-digit passcodes as they are available on all smartphones and not just Android devices like for the pattern passcode.

Let's consider a common situation. A user needs to access her smartphone in a public setting, such as on a subway or in the park. There are several other users around her that are watching her. This leaves an opportunity for them to shoulder-surf the user entering her passcode and attempt to guess what the passcode entered is. These users can then steal the smartphone and can thus have complete access to all the victim's files and pictures. Further, there are lots of cameras like CCTV

in a building or on the streets. These cameras are always recording and can record users' entering their passcode on their smartphone. For this reason, we focus on video recording to enable us to launch an online shoulder-surfing attack to predict users' smartphone passcode.

In order to perform the shoulder-surfing attack, we record videos from both the left and right side of the victim, simulating an attacker standing next to the victim in a public environment such as a bus. Our passcode prediction algorithm extracts the four frames where the victim is entering her four-digit passcode, rebuilds the grid of the keypad, and determines the location of the victim's fingertip. We can thus predict what number the victim is "typing". We evaluate our algorithm with 20 videos. We correctly predict 74 out of the 80 digits, a success rate of 92.5%. Our algorithm can also be deployed online, taking less than 75 seconds to make a prediction for a number.

Our main *contribution* is the design and implementation of an online algorithm to accurately predict a smartphone's four-digit passcode using a camera to record the victim entering the passcode. Our algorithm requires minimal training. It can also identify the smart device and rebuild the virtual keypad automatically regardless of the size and type of the device.

The rest of the paper is organized as follows. We review related work in Section 2.2. Section 2.3 shows the design of our passcode prediction algorithm. The experimental results are shown in Section 2.4. Improvements are discussed in Section 2.5. Finally, Section 2.6 concludes.

2.2 Related Work

Previous work has shown side channels attacks are possible in guessing key strokes. Timing attacks on SSH [5], reflections off windows or computer monitors [6, 7], and keyboard acoustics [8] can reveal the keystrokes of a user on a keyboard. If an ad-

versary has control over the mobile device, she can use the device's accelerometer [9], motion [10], or camera and microphone [11] to infer the user's keystrokes such as PIN or password. EyePassword [12] utilizes the user's gaze during password entry in an attempt to prevent shoulder surfing.

Shoulder surfing is a well-known problem [13–15]. Hoyle et al. found that life-logging cameras, such as Google glass, automatically recorded information that users preferred to keep private, including computer screens and bank card information. [16]. De Luca et al. created an authentication mechanism designed to protect against against shoulder surfing called XSide, which uses the front and back of smartphones [17]. This is a new authentication scheme and is not widely used. The most common form of authentication for mobile devices is PIN passcode. Roth et al. designed a PIN entry method to resist shoulder surfing [18]. Their scheme forces the user to perform some cognitive functions before entering their PIN; this makes the numbers entered for each authentication different each time assuming the attacker does not know the cognitive work to be performed.

Schiff et al. created a program to automatically recognize people based on visual markers [19]. Other research has looked at shoulder surfing vulnerabilities on a variety of a different types of password entry screens, including keyboards [20], graphical passwords [21, 22], and Android unlock screens [23, 24]. In their study of Android unlock screens, [23] created video recording software to guess Android passwords by tracking fingertip motions. In contrast, we look at PIN passcodes, which is a harder task because the numbers touched by a user have to be retrieved rather than a continuous pattern. Shukla et al. designed an attack to guess a user's PIN based on the movement of their hands as seen from behind a smartphone [25]. We attempt to retrieve their PIN from a different visual angle, namely behind the user. Moreover, we analyzed the smartphone screen together with the finger location instead of the hands' movement; we also achieved a higher accuracy.

Yue et al. [26] proposed an approach to detect the types of device used and based on the keypad arrangement of the device, predicts the PIN entered by analysing the user's hand. Compared to [26] which requires new training for new types of devices and new reference images of the software keyboard of the devices, our algorithm requires minimal training. Our algorithm can also identify the smart device and rebuild the virtual keypad automatically regardless of the size and type of the device.

2.3 Passcode Prediction

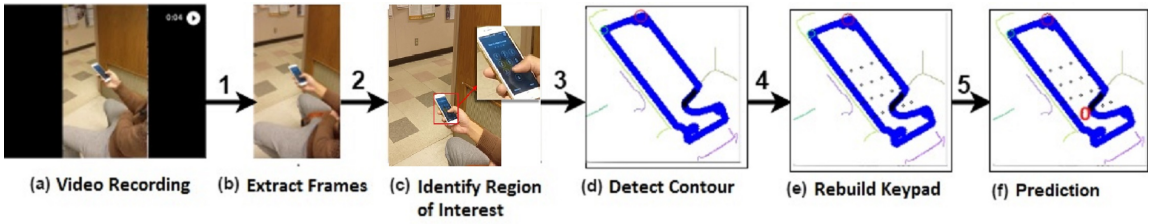


Figure 2.1: Overview of our attack to detect PIN entered on an observed smartphone. (a) Our attack starts with recording a video of a victim who is entering the passcode. The video consists of multiple frames. (b) TLD is applied to identify the frames where the victim is typing (clicking) on the screen. For a four-digit PIN, this will identify four frames. (c) A classifier is used to identify the Region of Interest (ROI) in each frame. (d) The Line Segments Detector outlines the edges (lines) of the smartphone. (e) Rebuild the numbers of the keypad on the screen from the user's point of view. (f) Apply the skin detector algorithm to filter out the fingers inside the screen and then predict the number the user touched on.

This section provides an overview of our passcode prediction scenario. An adversary is able to observe a victim entering her passcode on her smartphone. The observation could happen due to the ubiquity of people using smartphones and the emergence of head-mounted devices such as Google Glass, Microsoft Hololens, and Snapchat Spectacles. These devices can record videos without the knowledge of other people nearby. Moreover, users typically use their smartphones in public. If they use a passcode, such as a four-digit PIN passcode, then they have to enter that passcode each time they want to use their smartphone. An adversary thus can observe a victim entering her passcode multiple times and can record that observation without the

knowledge of the victim. Figure 2.1 outlines an overview of the processes involved in our passcode prediction. Our proposed attack consists of five steps: recording the videos, extracting the frames where the user is entering the PIN, identifying the region of interest (ROI) within each frame, detecting the contour of the smartphone in the region of interest, rebuilding the keypad, and making the prediction. Each step is described in more detail in the following subsections.

2.3.1 Video Recording

The first step, to perform passcode prediction, is to record a video of the victim entering her PIN. The video can be recorded from different positions from the point of view of the adversary. The different points of views are from the right side and the left side of the victim. Recording videos could be done stealthily as people use their smartphones all the time and could be recording at all times. Moreover, head-mounted devices will become more common in the future. A common scenario where this type of recording could take place is waiting at bus stations, when riding public transportation, or while waiting to check out in a line. Moreover, surveillance cameras such as CCTV can be recording the victim entering her passcode. The adversary has to record the victim entering the whole passcode but usually that still results in short videos of a few seconds. We note that most passcodes are four-digit numbers.

2.3.2 Extract Frames

Most smartphones can record videos at 30 frames per second. In a five-second video, there are $30 \times 5 = 150$ frames. Out of these 150 frames, four frames have to be extracted; these frames are the point when the user is touching one of the four numbers in the passcode. Some videos can be recorded at 60 frames per second. Similar to [23], we assume the following: 1) before and after unlocking, users often pause for a few seconds and 2) four consecutive on-screen number-touching operations with

short intervals because most people are used to unlocking their phones and are not expected to take long or make mistakes. Based on these assumptions, we process the video to reduce the number of frames that need to be analyzed by extracting only the unlocking process. Then, with the implementation of the Tracking-Learning-Detection (TLD) algorithm [27] in OpenCV [28], we can obtain the frames where the victim is entering her passcode by tracking the moving path of the fingertip. Based on the gradient (that is, analyzing a video in terms of frames) of frames in a video, a user typically stays on the same number for several frames; this is the time to touch the screen. By applying the second assumption to the result of the TLD algorithm, we can extract the relevant frames. In our case, it is 4 frames (four-digit passcode, one frame for each). TLD gives a tracking map of the fingertip movement over time; the frames we want are the ones with more dots or dark areas. Since a passcode is typically four numbers, the goal of this step is to extract the four frames associated with the victim entering the four numbers. This step uses the TLD algorithm to detect the typing moment and extract the frame. However, through experimental process, we noticed that sometimes the frame extracted by the TLD algorithm is not clear or the victim's finger is not exactly on the number. We thus also analyze the frame before and after. Figure 2.1(b) shows an extracted frame from a recorded video where the victim is entering the passcode.

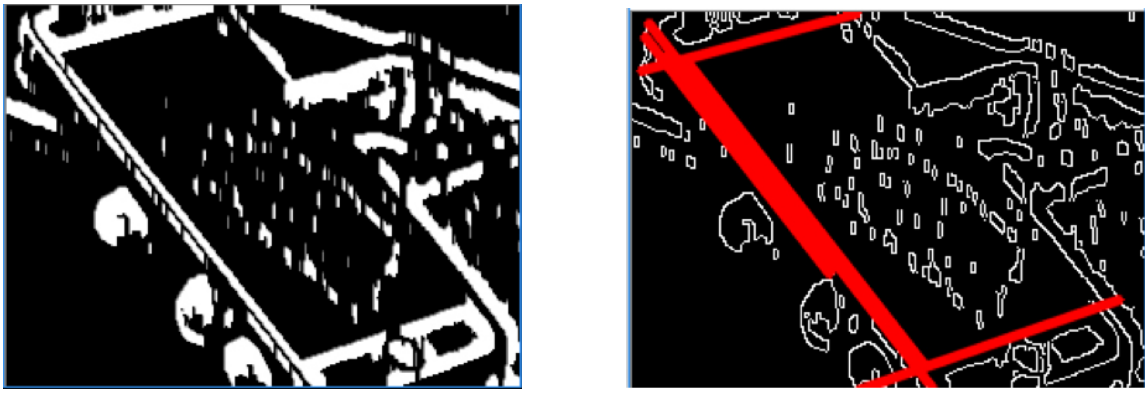
2.3.3 Identify Region of Interest

Once the four frames are extracted (in this case, it would be 12 frames since we consider the frame before and after the extracted frame), the region of interest (ROI) needs to be identified. In our case, the ROI is the smartphone so that the numbers being pressed can be detected. We use the Cascade Classifier Training [29] provided by OpenCV on our dataset. More specifically, object detection using the Haar feature-based cascade classifier [30] is an effective object detection method proposed by Paul

Viola and Michael Jones. It is a machine learning based approach where a cascade function is trained from many positive images and negative images. Positive images are images that contain the target object in the image while negative images are images that do not contain the target object in the image. The target object, in our case, is the smartphone. The classifier is then used to detect objects in our frames (each frame is an image). Figure 2.1(c) shows the identified ROI from the extracted frame.

2.3.4 Detect Contour

After the ROI is identified, the outline of the smartphone is then detected. This step is important for the following step of rebuilding the keypad on the smartphone. The position of the keypad depends on where the smartphone screen is. From Figure 2.1(c), the image must first be smoothed as shown in Figure 2.2(a). To achieve smoothing, there are two morphological operations we can use. These are dilation which adds pixels to the boundaries of the object in an image and erosion which does the opposite. After applying smoothing, we get a new image which allows the long edges of the smartphone screen to be extracted. The next step is to apply the Line Segments Detector(LSD) [31]. LSD is an algorithm that can help us to extract the edges of a smartphone from the ROI image. Figure 2.2(b) shows the edges of the smartphone highlighted. It will produce a set of lines in the image. We then apply the rules like line length and continuity to filter the noisy lines out. These rules are defined as following: 1) Line length should be no shorter than 30% of the height in the dimension of the ROI image; 2) Area of contour (continuous lines) should be no smaller than 10% of the whole ROI image. We tested different percentages and found that these give the best results for our experiments. Figure 2.2(b) shows the image after applying the rules.



(a) Smoothed ROI (b) Extracted Edges

Figure 2.2: Detecting the contour of the smartphone by extracting the edges from the ROI image and filtering out noise.

2.3.5 Rebuild Keypad and Retrieve Finger Contour

The last step before performing the passcode prediction is to rebuild the keypad on the smartphone. Based on the edges and the screen ratio which is 16 : 9 in this frame, the distance of the numbers of the keypad from the edges can be calculated. We use the Hue-Saturation-Value (HSV) values of the color of the person's skin which we want to filter out to remove the finger in the ROI. As shown in Figure 2.3, the skin of the person is highlighted and it can be removed so that only the smartphone screen is obtained. HSV color space is commonly used in computer vision due to its good performance when comparing RGB color space in varying illumination levels. Often thresholding and masking is done in HSV color space. So we apply the HSV values of a person's skin to get the outline of the person's finger. After this, we can cut the contour of the finger inside the phone screen out to predict the number being touched on the smartphone screen.

We combine the result of the previous two steps to project the contour of the finger to the key number grid to make a prediction of the current digit of the passcode. This is shown in Figure 2.4, which is the final resulting image of our passcode prediction algorithm. Our algorithm (see Section 2.3.6 for details) will do several passes to check the contour points beyond each line formed by 123, 456, 789, 0, 147, 2580, 369; this

Figure 2.3: Image of the ROI showing the person's hand. The person's skin color is filtered out to obtain the contour of the finger.

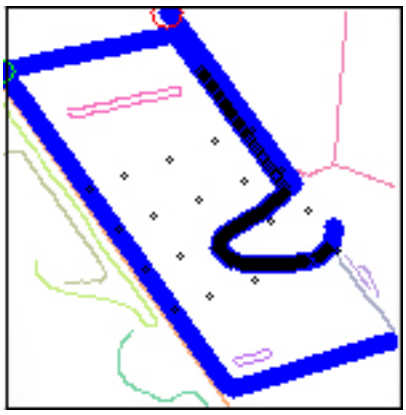
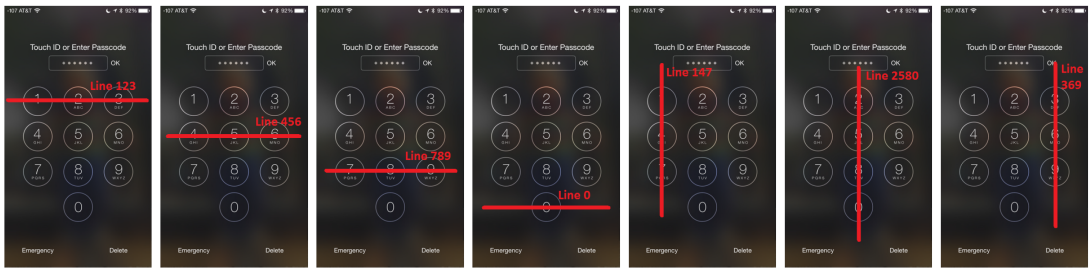


Figure 2.4: Final image where the contour of the smartphone screen, the contour of the finger, and the number keypad can be clearly seen.

is illustrated in Figure 2.5. For each pass, the probability of the number will be increased inversely proportional to their distances to the center of the contour.



(a) Line 123 (b) Line 456 (c) Line 789 (d) Line 0 (e) Line 147 (f) Line 2580 (g) Line 369

Figure 2.5: Pass line by line to locate the finger position.

2.3.6 Prediction Algorithm

We now describe in more detail the algorithm used to predict the passcode. In Section 2.3.5, we rebuilt the keypad on the smartphone screen and were able to outline the finger contour of the victim. Essentially, the prediction algorithm is about identifying the location of the fingertip of the victim's finger on the keypad. If the finger is not on an exact number, then the number closest is chosen. To determine the position of the fingertip, we calculate the number of points (pixels) of the finger contour "line by line." Each line is the set of horizontal or vertical lines formed by the keypad. For example, the four horizontal lines are formed by the set of numbers 123, 456, 789, and 0, and the three vertical lines are formed by the set of numbers 147, 2580, and 369 (see Figure 2.5). Algorithm 1 shows the process of first removing the points/pixels of the finger contour that are not inside the phone screen.

Algorithm 1 Eliminate points of the finger contour outside phone screen

```
1: Function get_inside_points(FingerContour, ScreenEdges)
2: inside_points ← [ ]
3: for each point point in FingerContour do
4:    if point is in ScreenEdges then
5:       inside_points.add( p)
6:    end if
7: end for
8: Return inside_points
9: EndFunction
```

The next step is to calculate the percentage of points "beyond" each of the horizontal lines. "beyond" means the points that are under the line, that is, the line that the points are associated with. For horizontal lines, this means, the points above the numbers while for vertical lines, this means points to the left of the numbers. This process cuts the finger contour into pieces by the lines 123, 456, 147, and so on and then counts the points beyond or left of each line to get the location of the finger contour. This is illustrated in Figure 2.6. For example, the percentage of points beyond the horizontal line 123, $P(123)$, is calculated as

$$P(123) = \frac{\texttt{\# of points beyond 123}}{\texttt{total \# of finger contour points}} \tag{2.1}$$

Based on the calculated percentages, we can tell if the fingertip contour is rightside up or down. The percentages are also used as the weight to calculate the probability for each number on the keypad. This is calculated as the average distance of each point to each of the ten numbers on the keypad.Each number is represented by a center point of a number round which is shown in keypad. So that, we can calculate the distance to the number as the distance to the center point of it. This average distance of all the points to the number 1 is calculated as

$$avg\_dist(1) = \frac{\sum_{i=1}^{n} \sqrt{(point_i.x - 1.x)^2 + (point_i.y - 1.y)^2}}{\texttt{number of points beyond line 123}} \tag{2.2}$$

where $point_i.x$ is the point i's x-position and $1.x$ is the number $1's$ x-position. Based on all the percentages, we can calculate which number is closer to the points of the finger contour. Algorithm 2 shows this process for all the numbers of the keypad. For example, $P(123)$ gives the percentage of all three numbers 1, 2, and 3 of being selected and $P(147)$ gives the percentage of all three numbers 1, 4, and 7 of being selected. In this case, the number 1 appears twice and the probability of the number 1 being the correct number selected by the user can be calculated. The calculation for the number 1 is as follows

$$P(1) = \left(1 - \frac{avg\_dist(1)}{avg\_dist(1) + avg\_dist(2) + avg\_dist(3)}\right) * P(123) + \left(1 - \frac{avg\_dist(1)}{avg\_dist(1) + avg\_dist(4) + avg\_dist(7)}\right) * P(147) \tag{2.3}$$

Algorithm 2 Make prediction by calculating probability of each number

Function{make_prediction}{$FingerContour, PhoneNumber1to9$}
    ▷ Horizontal lines
$points\_beyond\_line123 \leftarrow FingerContour.points\_beyond(line123)$
$points\_beyond\_line456 \leftarrow FingerContour.points\_beyond(line456)$
$points\_beyond\_line789 \leftarrow FingerContour.points\_beyond(line789)$
$points\_beyond\_line0 \leftarrow FingerContour.points\_beyond(line0)$
    ▷ Vertical lines
$points\_beyond\_line147 \leftarrow FingerContour.points\_beyond(line147)$
... ▷ Calculate percentage of points beyond each line
$P(123) \leftarrow \#\_points\_beyond\_line123$ / $total\_\#\_points$
$P(456) \leftarrow \#\_points\_beyond\_line456$ / $total\_\#\_points$
...
    ▷ Calculate average distance of each point to each line
$avg\_dist_1 \leftarrow avg_dist(points\_beyond\_line123, point\_num1))$
$avg\_dist_2 \leftarrow avg_dist(points\_beyond\_line123, point\_num2))$
...
    ▷ Calculate probability of each number
$P(1) = (1 - \frac{avg\_dist(1)}{avg\_dist(1)+avg\_dist(2)+avg\_dist(3)}) * P(123) + (1 - \frac{avg\_dist(1)}{avg\_dist(1)+avg\_dist(4)+avg\_dist(7)}) * P(147)$
...
3 numbers with highest probabilities $\leftarrow max\_3\{P(0), P(1), ..., P(9)\}$
Return **3 numbers with highest probabilities**
EndFunction

2.4 Results

We now go over the results of our passcode prediction.

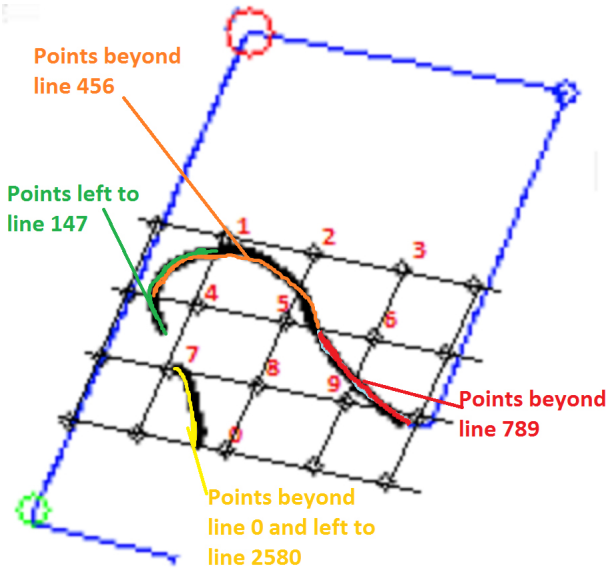


Figure 2.6: Illustration of our prediction algorithm.

2.4.1 Data Collection

We recorded 20 videos with ten videos from both angles: from the left side of the "victim" and from the right side of the "victim" with a distance about 1.5 meters (4.9 feet). This distance is common in daily life situation like at a bus station. In our experiment, the victim's smartphone was a white iPhone 6s. An OnePlus 3T Android phone with resolution 1080×1920 and frame rate of 30 frames per second was used to record the videos. Figure 2.7 shows the duration and total number of frames of each video. The average duration of a video is 3.53 seconds and the average frame count of a video is 105.75.

2.4.2 Automated Passcode Prediction

Based on our passcode prediction design from Section 2.3, we now present the results of our algorithm on the recorded videos. Table 2.2 shows the prediction for the passcode (2459) of the video .Each column shows the predicted number from our algorithm along with the confidence percentage associated with that number. It can be seen from the table that the predicted numbers are physically close to each other

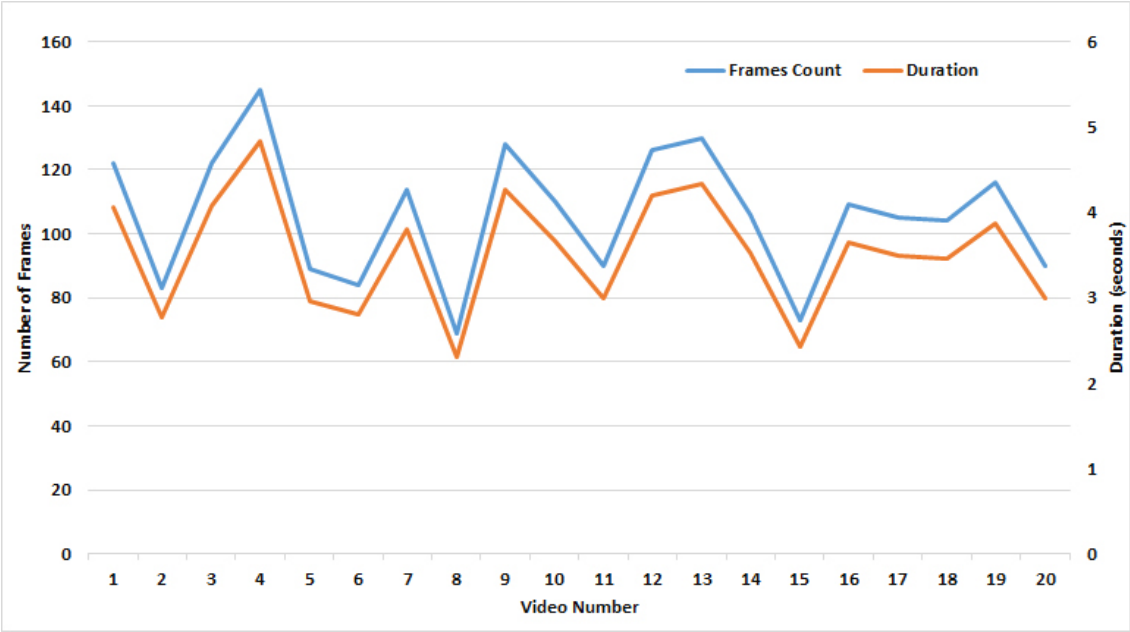


Figure 2.7: Video Frames Count and Duration Distribution.

on the number keypad. In this particular case, the first prediction was correct. We note that making three predictions is still practical since the algorithm provides a confidence percentage for each predicted number.

Table 2.1 shows the number of correct number predictions. Recall that we recorded ten videos from the left point of view of the victim and ten videos from the right point of view of the victim, and that each passcode is a four-digit PIN. Thus, there are a total of 80 numbers to be predicted. The table shows the breakdown from each point of view. When allowing our algorithm to only make one prediction, that is, predicting the most likely number, then it is able to correctly predict 26 out of the 40 numbers from the right side and 16 out of 40 numbers from the left side. Although the success rate is not high, if we increase the number of predictions to the top two most likely numbers, then the algorithm is able to predict 34 out of 40 numbers from the right side and 29 out of 40 numbers from the left side. When we increase the number of prediction to 3, then our passcode prediction algorithm is able to correctly predict 92.5% of the numbers, or 37 out of the 40 numbers from either side.

The success rate of our algorithm in correctly predicting each of the ten possible

numbers as shown in Table 2.3. It can be seen that most of the numbers on the edges of the keypad can be perfectly predicted while the numbers in the "middle" are harder to predict, like 4, 5, 7, and 9.

Table 2.1: Number of correct predictions of the passcode detection algorithm on recorded videos from the left and right side of the victim.

	Right Side	Left Side
Total # of numbers	40	40
# of correct first predictions	26/40	16/40
# of correct second predictions	8/40	13/40
# of correct third predictions	3/40	8/40
# of correct predictions	37/40	37/40

Table 2.2: Example of one passcode *2459* and the predictions along with the confidence of our algorithm.

User PIN	First Prediction (confidence)	Second Prediction (confidence)	Third Prediction (confidence)
2	**2 (38.5%)**	3 (37.5%)	5 (23.9%)
4	**4 (44.3%)**	1 (29.5%)	5 (26.2%)
5	**5 (45.0%)**	8 (32.0%)	2 (23.1%)
9	0 (37.0%)	**9 (33.0%)**	6 (30.0%)

Table 2.3: Success rate for each number from both sides.

Number	Success Rate
0	100.00%
1	100.00%
2	100.00%
3	100.00%
4	77.78%
5	92.86%
6	100.00%
7	80.00%
8	100.00%
9	71.43%

2.4.3 Analysis of Failed Predictions

Most of the failed guesses in predicting the passcode in our experiments are caused by distorted images we obtained from the two aspects of image processing of our

algorithm. The first aspect is identifying the Region of Interest (see Section 2.3.3). The second aspect is to correctly rebuild the edges of the phone screen. We will discuss these two aspects in more detail in the following section.

2.4.4 Algorithm Processing time

The training for the cascade classifier to identify the Region of Interest takes about 7 days to reach a positive rate of 95%. However, this training only needs to be done once. Our prediction algorithm, using the OpenCV software, performs the prediction process of one passcode (four numbers) in less than 30 seconds. The TLD algorithm takes less than 5 minutes to analyze and output the target frames. Overall, the prediction process takes about 5 minutes for the whole 4-digit passcode. Predicting one number will thus take about 75 seconds which can be considered to be real-time.

2.5 Improvements and Discussions

We discuss improvements that can be made to the passcode prediction algorithm, along with discussion of future work.

Four out of the six failed predictions were due to the failure of the algorithm to detect the Region of Interest in the video frame. Since identifying the Region of Interest depends largely on the Haar feature-based cascade classifier, more positive and negative images with training at a higher positive rate will likely resolve this issue. However, this will increase the training time for the classifier, but we note that the training only needs to be performed once. The other failed predictions were incorrect predictions. Upon examining the frames manually, it is hard to tell which number the user was touching.

A possible improvement to our prediction algorithm is to use machine learning to do the prediction. The training data will consist of images of users touching each of the ten numbers from several angles. The first part of our algorithm still needs to be

performed to identify the four frames. Afterwards, the machine learning algorithm can take over to predict based on the image.

We used only one smartphone as the victim's device. We plan on using different types of smartphones in the future; we don't expect this will affect our algorithm as the screen edges can be identified regardless of the phone size and the number keypad is built based on where the edges are. More videos taken at different angles such as from the top and front of the victim can be recorded to generalize the algorithm further. More videos will highlight the accuracy of the prediction algorithm. Different angles will increase the complexity of the prediction; for example, for the top view, it is hard to detect the movement of the fingertip while for the front view, the smartphone blocks the fingertip.

Since we have shown that an online passcode prediction attack on smartphones is possible and relatively fast (less than 75 seconds), the next step is to determine mitigation mechanisms. One possibility is to use the other hand to hide the passcode authentication process, similar to how ATM keypads are hidden while entering the ATM PIN. Another possibility is for the user to move her finger randomly so as to fool the algorithm in identifying the four frames. Randomizing the keypad number is another possibility. The user could also hold her phone in one hand and use multiple fingers of the other hand to enter the passcode. All these mitigation techniques decrease the convenience for the user which may lead the user to not have a passcode. We leave analysis of these mitigations as future works . We note that these mechanisms provide a trade-off between security and convenience.

Our passcode prediction algorithm can also be extended to recognize passwords entered instead of just PINs. The algorithm will have to be extended to rebuild the keyboard rather than the number keypad and will also have to account for different types of keyboards. We also emphasize that no special hardware is needed – we are using a commodity smartphone to record the videos. A better quality camera will

likely improve the accuracy of our algorithm.

2.6 Summary

In this paper, we design and develop an attack to accurately predict the passcode entered by a victim on her smartphone. The attack relies on recording a video of the victim using a common smartphone in a public environment. Our algorithm achieves an overall accuracy of 92.5%. This result demonstrates that online shoulder-surfing attacks on PIN-based authentication are possible. Also, these results show that choosing a good random PIN cannot prevent this type of online attack as the algorithm can still predict the PIN entered.

CHAPTER III

Measuring Tor Relay Popularity

3.1 Introduction

Tor is one of the popular anonymous communication systems that can protect users from leaking private information, such as the IP address, when users are browsing the Internet and communicating with other users. Tor also allows users to circumvent censorship effectively to reach blocked websites or documents. Some companies (such as Facebook [32]) are even using Tor hidden services when they publish their services or websites. Tor has more than $6,000$ volunteer-operated relay nodes (servers/routers) [33]. Instead of connecting users directly to web servers or each other, a Tor client makes the connections go through three of the $6,000$ relay nodes, then to the destination. The Tor relay selection algorithm makes sure that no two relays with the same /16 IP address are chosen. The three randomly selected relay nodes form a Tor circuit. A circuit is re-used to transfer several TCP streams with a maximum lifetime of 10 minutes [34].

As Tor becomes more popular, it becomes subject to a number of attacks and respective countermeasures. The packet counting [35], end-to-end timing attack [36], active and passive end-to-end confirmation attacks [36–38] are shown to be possible in the Tor network. Tor aims to protect users against traffic analysis. If an adversary has control or can eavesdrop over both entry and exit relay nodes, then a statistical correlation attack can be performed by using the packets' timing or packets' size information. Entry guards, which Tor client selects from a few relays at random as the entry points, help against this kind of correlation attacks. Hence, our focus is on

the middle and exit relays.

In this paper, we explore if there are some relay nodes which have a higher chance to be selected in a circuit than other relays. Namely, these relays are more popular. We also look into the /8, /16, /24 subnets and Autonomous System (AS) number of all relay nodes to find out the popular subnets. We then look at the relationship between the bandwidth of relay nodes and the popularity of relay nodes.

Our contributions are listed as follows:

- **Popular middle and exit relays**: Based on our dataset, we find there are some middle and exit relays which are more popular than other relays. Some Tor relays are 3 times more likely to be chosen than others, in building Tor circuits. If Tor relays are randomly chosen, then some Tor relays have over 10 times higher chance of being selected.

- **Popular /8, /16, /24 subnets**: We analyze all relay nodes based on their /8, /16, /24 subnets. Several subnets stand out as more popular than others. We also see that correlation attacks are still possible on a small fraction of circuits where an adversary controlling a /16 subnet could monitor network traffic of both the client and the target website.

- **Popular ASes**: Some ASes are more popular than others, based on the number of Tor relays belonging to those ASes. However, our results show that some of these ASes have a much higher percentage of being selected, regardless of how many Tor relays are in these ASes.

- **Correlation attacks are still possible**: We find that about 11% of circuits built can be correlated, that is, both the client and server identified.

3.2 Background

Tor [34] is a popular low-latency anonymity network built over TLS connections and based on onion routing. Tor is used by over 2 million unique users [33]. The Tor system is run mostly by volunteer relays, with over $6,000$ relays [33]. Each relay reports its IP address, public key, bandwidth, and contact information for the owner to the centralized directory servers. When a Tor user (client) wants to use the Tor network to connect to a server, it first contacts the directory servers to obtain a consensus document of all the Tor relays. It then selects three Tor relays based on a relay selection algorithm, see Figure 3.1. Tor relays are also referred to as nodes or Tor routers. They are responsible for receiving and forwarding Tor traffic. The three Tor relays chosen by the client are contacted telescopically. The client establishes a secure connection with the first relay. Then going through the first relay, the client establishes a secure (and anonymous) connection with the second relay. Going through the first and the second relays, the client finally establishes a connection with the third relay. This process builds a circuit for the client to use to connect to a server.

Tor traffic is sent in fixed-size cells where each cell is 512 bytes [39]. When the user makes the request, the user/client builds a circuit consisting of three relays (entry guard, middle node, and exit node) before connecting to the destination server. The constructed circuit can be shared by many TCP streams. Tor clients construct circuits preemptively and substitute previously used ones with newly built circuits. Each circuit lifetime is 10 minutes. Figure 3.1 shows the data flow in the Tor network where the request of a client will pass through three Tor relays before reaching the web server. In the circuit, the entry guard knows that the client is communicating with the middle relay, but not who the exit relay or the destination server are. The middle relay knows the entry guard is communicating with the exit relay but not who the client or the destination server are. Similarly, the exit relay knows the middle

relay is communicating with the server, but not who the client is. The server only knows that the exit relay is acting as the client, but does not know who the real client is.

Early onion routing systems initially specified that clients should select relays uniformly [40] at random. With the increasing number of Tor users and relays, it became necessary to improve relay selection strategy to balance traffic load with the available Tor relay bandwidth. The choice of relays is determined by a weighting function that includes the bandwidth, status flags of the relays and multiple other considerations [41]. To be chosen, a relay has to have the following flags: stable, running, and valid. A Tor relay is considered to be "Running" if it has been successfully contacted within the last 45 minutes [41]. Tor does not take the locations of relays relative to the clients into consideration [42]. It will reuse the same circuit for new data streams for 10 minutes. An exit relay has extra considerations since it has to have an open outgoing port to the server (e.g. port 80 for web servers); many Tor relays do not allow outgoing traffic outside of the Tor network.

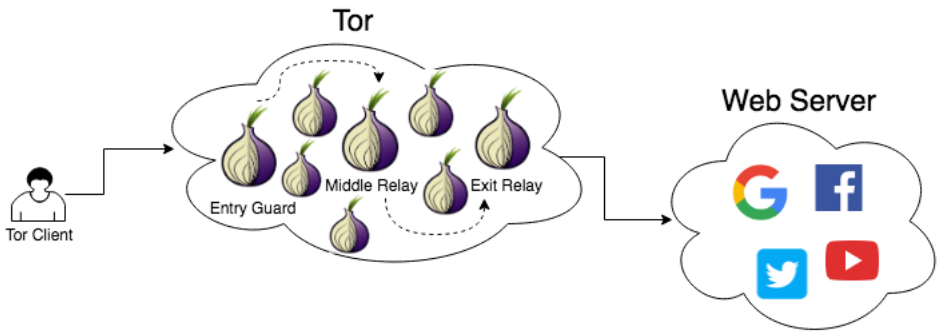


Figure 3.1: How Tor works. 3 nodes are selected from running Tor relays

To reduce the probability of disclosing the client information to attackers, Tor users randomly select a few relays to use as entry guards, and use only those relays for their first hop. The entry guard knows the identity of the client and the middle node for each circuit. The same entry guard is kept for 2 months. The relay can be considered as an entry guard only if it is fast, stable and has higher bandwidth

than a specific threshold. More details of selecting entry guards can be found in [41]. The entry guard, middle relay, and exit relay are chosen from different /16 subnet IP addresses.

3.3 Experiment Setup

This section provides an overview of the design of our experiments and how we collect and analyze data.

3.3.1 Data Collection

We deployed our experiments on 44 different machines from different locations. The machines are from Google cloud instances and the PlanetLab network [43]. PlanetLab is a global research network that supports the development of new network services. On each machine, we installed Tor and Stem [44] version 1.5.2, which is a Python controller library for Tor.

We set up a script supported by Stem to automate visiting websites. We visited the homepages of Alexa top 100 websites [45] sequentially through Tor for each of the 44 machines. During each visit to a website, we discard the TCP packets. Only metadata information of Tor circuits are collected. This includes the following information.

1. Tor Relays. There are 3 Tor relays during a visit to a website: entry guard, middle relay and exit relay. The IP address and port, fingerprint, nickname, locale, and advertised bandwidth are collected. We also converted the IP address to a geolocation (city and country) and to an AS number. The advertised bandwidth is the volume of traffic, both incoming and outgoing, that a relay is willing to sustain, as configured by the operator and as observed from recent data transfers.

2. Source. The IP address and the port number of the Tor client. This will be one of the 44 machines used.

3. Target. The IP address and the port number of the target destination. This will be one of the 100 top Alexa websites.

The experiments were run for 5 months, from November 2017 to March 2018. Our dataset consists of 145,918 entries. Each entry contains the information above: source IP address, target IP address, the three Tor relays' IP address, fingerprint, nickname, locale, AS number, and bandwidth. The list of the top 100 Alexa websites was downloaded on October 15, 2017.

We used our own clients to visit known websites. Other than the Tor relays information such as IP address and bandwidth, which are already public information, we do not collect any private data. Our automated experiments are also spaced out such that the extra 44 clients would not affect the normal operation of the Tor network.

3.3.2 Data Analysis

We next describe the type of analysis performed on our dataset. Since Tor sets the entry guard to be the same for an extended period of time, we mainly focus our analysis on the middle relay and exit relay. More specifically, our goal is to determine if some relays are chosen more often than others meaning some Tor relays are more popular. This could lead to privacy issues as an attacker can utilize that knowledge to target the anonymity of users.

- "By Source": we first analyze the dataset from the point of view of the 44 client machines. We look mostly at the popularity of Tor relays, that is, how often they are selected for circuits and how often they are in the top k relays for each machine. We set $k = 30$ for our analysis. Next, we group the Tor relays by subnets; we analyze /8, /16, and /24 subnets. If two Tor relays are in the same /16 or /24 subnets, then they are likely in the same AS or controlled/observable by the same entity.

- "By Target": we perform the same analysis, but this time considering the point of view of the 100 target websites. The goal here is to determine whether some Tor relays are more popular based on the website visited. This could mean some websites are more targeted or could increase the likelihood of an adversarial entity (such as an ISP) being able to determine the target website.

- Overall: we then perform a holistic view of our dataset to identify who the most popular relays are.

3.4 Experimental Results

This section presents the results from our experiments and the analysis based on the results for the middle and exit relays. For each approach as described in the previous section ("by source", "by target", and overall), the analysis is based on the IP address, /24 subnet IP address, /16 subnet IP address, /8 subnet IP address, and AS number for the Tor relays. Due to space restrictions, we show the results for the /16 subnet as a representative result. The results and conclusions drawn from the /8 and /24 subnets are similar. We also show the result by AS number.

3.4.1 Dataset Overview

Our 44 machines visited 145,918 sites in total. On average, each machine visited 3,316 websites. This means that each of the top 100 Alexa websites received 33 visits on average from each machine. During each website visit, we collected the IP address for the client, the entry guard, the middle relay, the exit relay, and the target website. Figure 3.2 shows a graph from the Tor metrics website [33]. It contains the number of running relays that have flags "Running" and "Stable" assigned by the Tor directory authorities. Over the 5 months, from November 2017 to March 2018, there are about 4,800 stable relays and about 6,000 running relays. As can be seen from the figure, new relays join the network and old relays leave the network due to churn. During

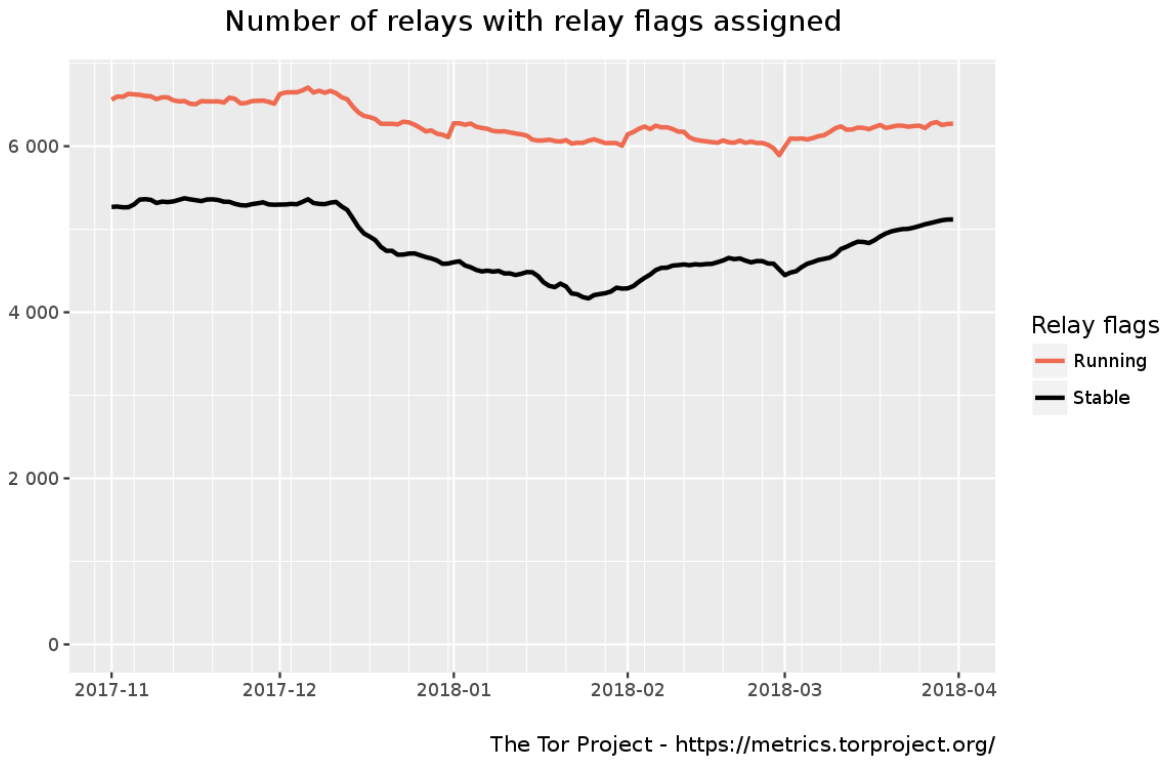


Figure 3.2: This graph shows the number of running relays that have the flags "Running" and "Stable" assigned by the Tor directory authorities. The graph shows the range from November 2017 to March 2018 which is the duration of our experiments.

our experiments, our machines connected to 8,523 unique relays.

The number of unique IP addresses, /24, /16, and /8 subnet IP addresses of the relays used during our experiments are listed in Table 3.1. 8,523 unique relay nodes are used in the experiments. These relay nodes are almost all the running nodes during the time period of our experiments. Since there are around 6,000 relay nodes at any time, on average, each relay node will be used in 24 circuits or website visits. Out of 145,918 total circuits created, this comes up to $24/145918 = 0.02\%$. Each of our 44 source machines connected to 2,061 relay nodes (or 2,055 unique relays because some relays can be used as either middle or exit relay) on average (for the 3,316 circuits built). Breaking it down to different types of relays: on average, this comes up to 3 entry guards, 1,550 middle relays, and 508 exit relays for each machine. Each middle relay would then be used, on average, $3316/1550 = 2.14$ times

in total. This means that out of the 3,316 circuits, a Tor relay has a chance of 2.14/3316 = 0.065% chance of being selected as a middle relay. Each exit relay is used, on average, 3316/508 = 6.52 times in total. This means that a Tor relay has a 6.52/3316 = 0.20% chance of being selected as an exit relay. All the relays found belonged to 1,096 ASes, which means there were on average 466 relays per AS.

Table 3.1: # of Tor relays, in terms of unique IP addresses, /24 subnet, /16 subnet, /8 subnet and AS number used when collecting data in our experiments over 5 months.

	# of relays	Avg # relays per machine	Avg # entry guards per machine	Avg # middle relays per machine	Avg # exit relays per machine
IP addresses	8,523	2,054.93	3.02	1,550.39	508.47
/24 subnets	6,949	1,695.11	3.02	1,392.18	355.77
/16 subnets	2,885	851.91	2.95	725.82	237.16
/8 subnets	184	129.43	2.82	119.84	82.07
AS	1,096	466.52	2.73	384.18	157.91

3.4.2 Metrics Used

We use the following metrics to determine the popularity of Tor relays.

1. **Relay percentage**: this is the percentage of visits that include the relay node in the circuit. In "by source", this means the percentage out of 3,316 visits for each machine. In from all sources, it is the percentage out of all 145,918 visits. This will likely be a low number but the goal is to determine if some relays are used more often in circuits than others.

2. **Repeated percentage**: this is the percentage of relay nodes that appear in the top 30 most-used relays of one source machine and also appear in the top 30 most-used relays for the other 43 machines. For example, relay node A is in the top 30 most-used nodes for machine S. This means that out of the 3,316 circuits/visits for machine S, the relay A is in the list of 30 most-used relays for these 3,316 circuits. To continue the example, let's say that A also appears in the top 30 most-used relays for 32 other machines (out of 44 machines in total). Then the repeated percentage for relay A is (32+1)/44, which is 75%.

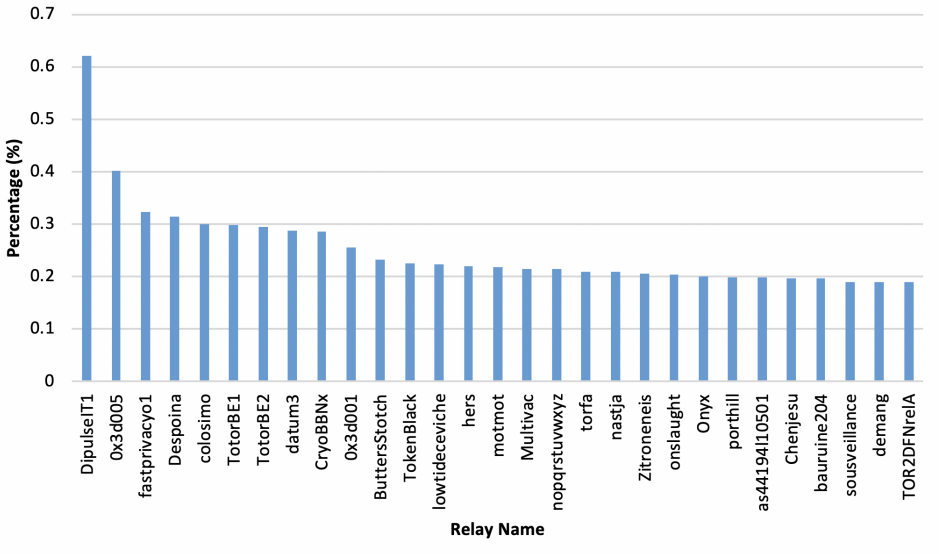


Figure 3.3: Point of view of all machines. The percentage of times a relay has been used as a middle relay for all circuits. This shows the top 30 most-used relays.

3. **Relay bandwidth**: this is the advertised bandwidth from [33]. It is the volume of traffic, both incoming and outgoing, that a relay is willing to sustain, as configured by the operator, and observed from recent data transfers.

4. **Relay popularity**: if a relay is popular, the relay node will get more traffic/visits/selections than other nodes. In our analysis, a factor for popularity is the percentage of visits a node has. Previously, we showed the average relay percentage is 0.02% (of total 145, 918 visits). In terms of one source machine, each middle node gets a relay percentage of 0.065% (of 3316 visits) and each exit node gets a relay percentage of 0.20% (of 3316 visits) on average.

3.4.3 Analysis of Middle Relays

We first analyze the popularity of relays chosen as middle relays in all the circuits created by the 44 machines.

1) By source IP address: We first look at the popularity of middle relays from the point of view of the source machines (clients). Figure 3.3, Figure 3.4,

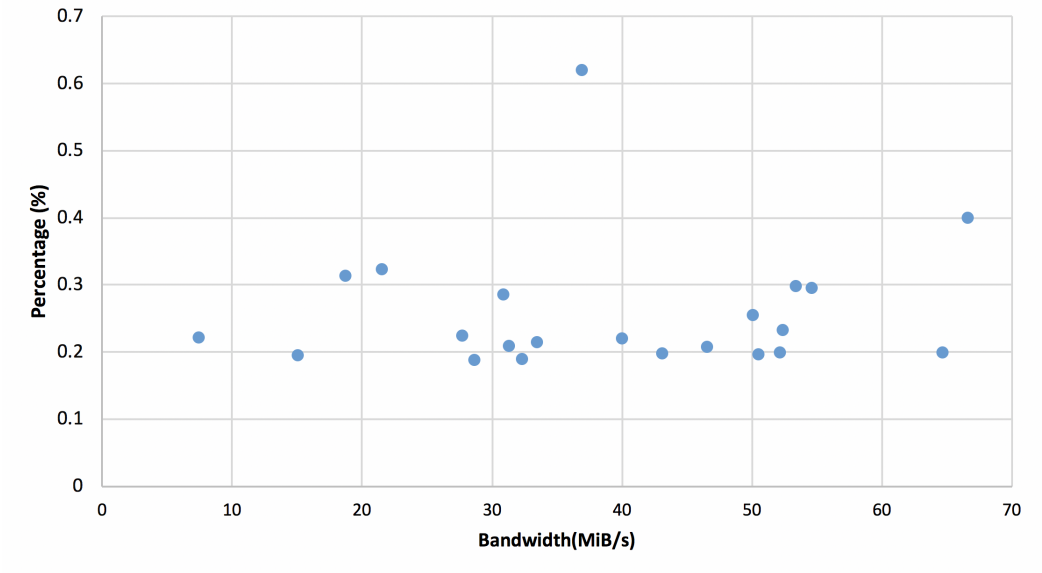


Figure 3.4: Point of view of all machines. The percentage of times a relay is selected as the middle relay in a circuit and that relay's corresponding bandwidth.

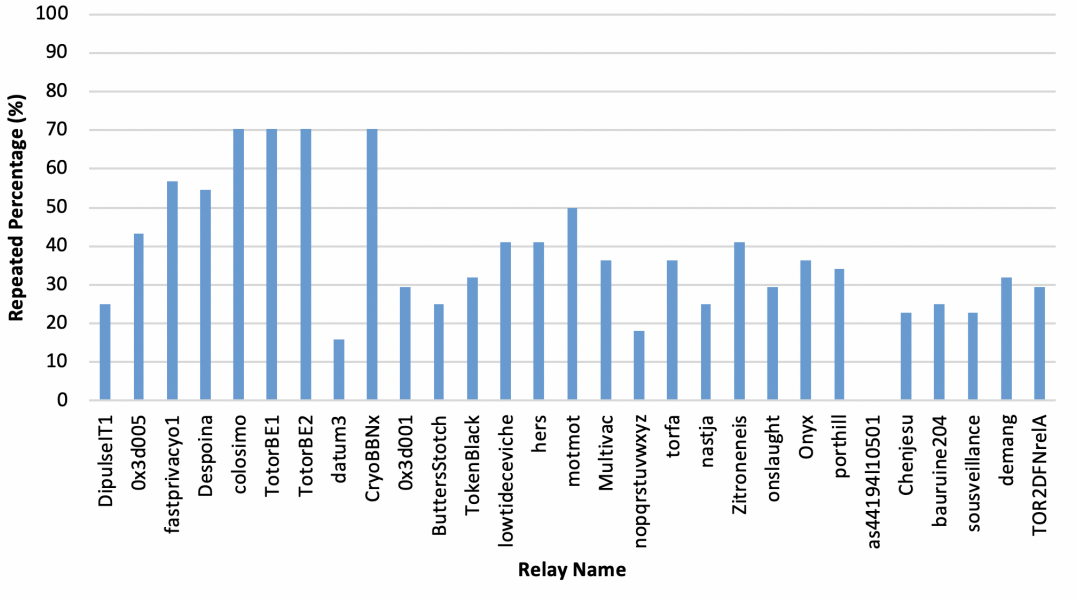


Figure 3.5: Point of view of all machines. The repeated percentage for each relay is chosen as a middle relay. This shows the top 30 most-used relays.

and Figure 3.5 are the analysis results when considering all our source machines together, that is analyzing all circuits from all source machines together. Figure 3.3 shows the 30 most-used middle relays and the number of times as a percentage that they have been selected as Tor middle relays for all circuits. From Figure 3.3, the relay named *DipulseIT1* with IP address *62.210.82.83* has the highest probability of being selected, at around 0.63%. That is about 10 times higher than the average 0.065% we mentioned in Section 3.4.1. In Figure 3.4, we have the % of a middle relay being selected as a middle relay in circuits to websites and the bandwidth of that relay. There is no obvious relationship between the percentage of a relay being selected and its bandwidth in the middle relay selection. Figure 3.5 shows the percentage of the top 30 most-often-selected relays that are also in the top 30 most-often-selected relays when considering machine by machine separately. As an example, let's consider the relay named *CryoBBNx* with IP address *51.254.45.43*, the relay named *TotorBE1* with IP address *5.39.33.176*, and the relay named *TotorBE2* with IP address *5.39.33.178*. They all have a repeated percentage of about 70%. That means, they are also in the top 30 most-used relays of 70% of all machines. This further confirms their popularity. Over all the relays, we can see from the figures that some relays have a much higher chance of being selected as a middle relay in Tor circuits than others, regardless of their bandwidth. These relays are not only popular for one client machine, but also for other client machines, regardless of the source IP address.

2) By source /16 subnet: Instead of considering each middle relay by their IP address, we now group the middle relays' IP address in /16 subnets. Figure 3.6 shows the top 30 most-selected /16 subnet IP addresses for middle relays, along with their percentage of being selected from all source machines. Figure 3.6 shows that the relays in the subnet *51.15.*.** and *163.172.*.** have higher percentages being selected during Tor circuit building. These two /16 subnets are in the top 30 most

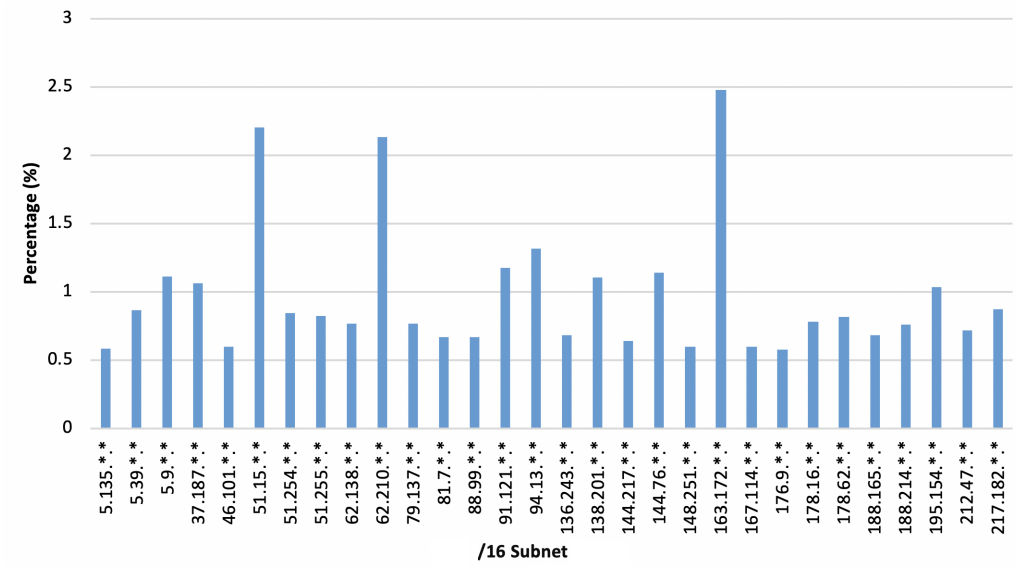


Figure 3.6: Point of view of all source machines. The percentage of selection of the top 30 most-selected /16 subnet IP addresses for middle relays.

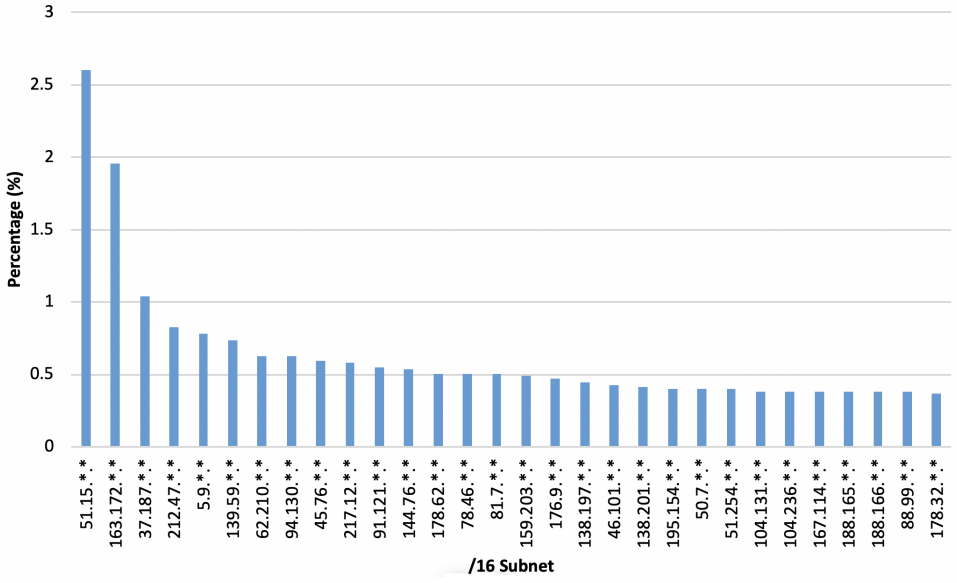


Figure 3.7: Percentage of # of middle relays in a /16 subnet.

often selected middle relays of each source machine as well. The subnet *163.172.*.** have the highest percentage near 2.5%. This shows that certain subnets are more popular than others. This could mean that these subnets contain more Tor relays. It could also mean that an adversary in these subnets will have a higher chance of being selected as a middle relay than others. When analyzing /8 and /24 subnets, we see a similar result.

It could be argued that subnets with more relays will obviously have a higher chance of being selected. Figure 3.7 shows the number of middle relays under each /16 prefix subnet (only the top 30 subnets are shown). In the figure, *163.172.*.** has less than 2.0% middle relays in it when compared to the 2.5% chance of being selected in circuits. However, *62.210.*.** has nearly 0.6% middle relays and it has a 2.1% chance of being selected. This indicates that the chance of a /16 subnet being selected is not proportional to the number of relays it contains.

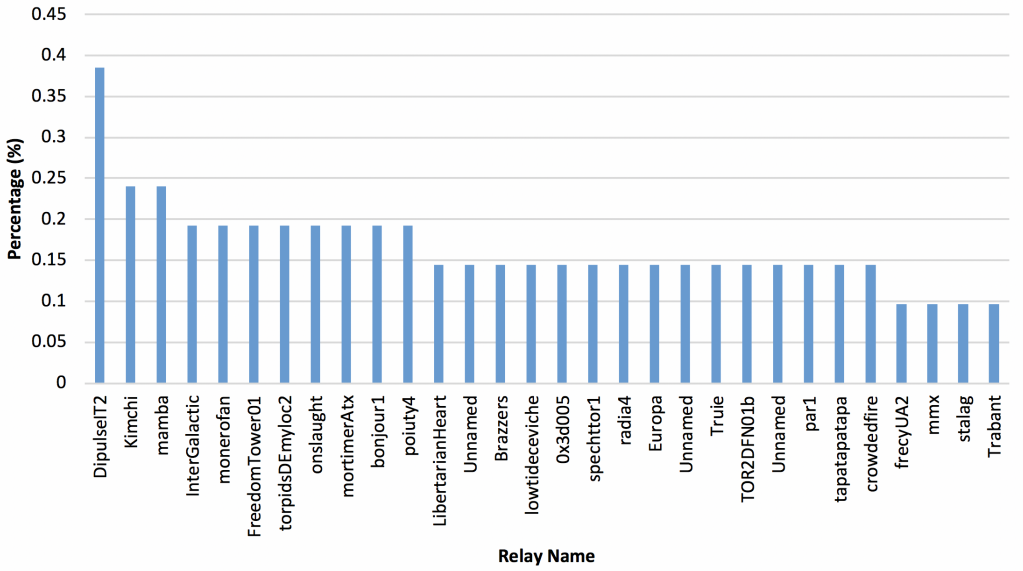


Figure 3.8: Point of view of one target destination IP address *151.101.128.81*. The % of selection of the 30 most often selected Tor relays as the middle relays in circuits.

3) By target IP address: Looking at all Tor relay routes from the perspective of the target websites (the web servers), we perform a similar analysis as earlier.

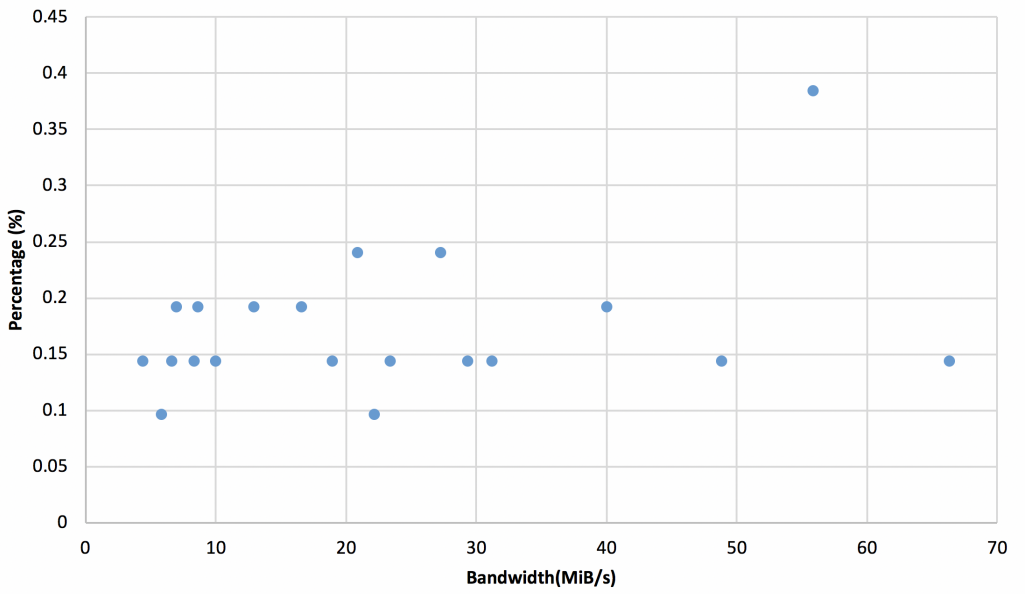


Figure 3.9: Point of view of one target destination IP address *151.101.128.81*. The percentage of selection of the top 30 most often selected Tor relays as the middle relays in circuits and these relays' corresponding bandwidth.

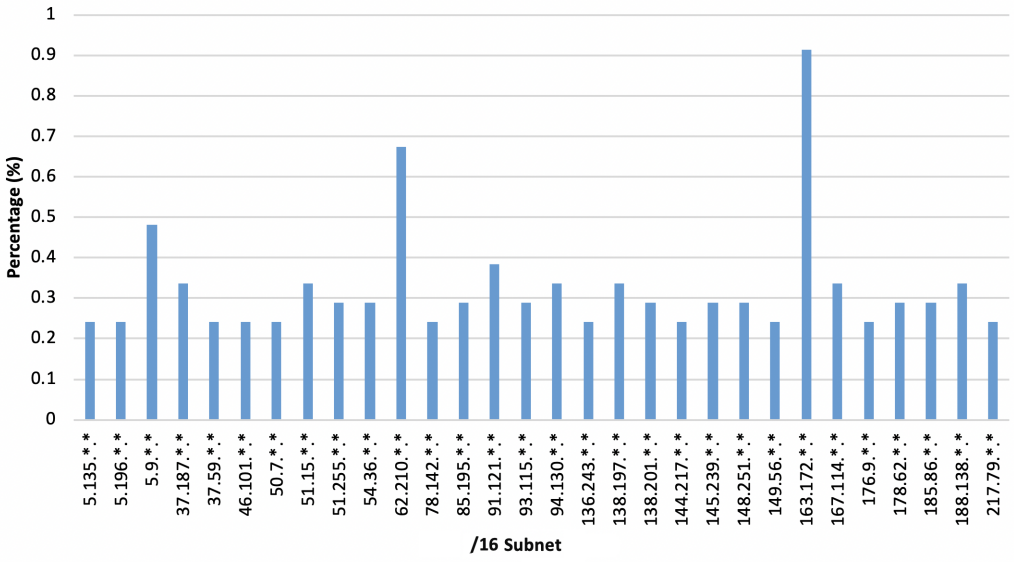


Figure 3.10: Point of view of target IP address *151.101.128.81*. The % of the 30 most often used relays being selected as middle relays, grouped by /16 subnets.

Figure 3.8 shows the percentage of each relay being selected as the middle relay when the target IP address is *151.101.128.81*. The figure includes the top 30 most often selected relays. The relay named *DipulseIT2* with IP address *62.210.82.83* is the most often selected relay with a percentage of 0.385%. Figure 3.9 shows the percentage of being selected and the bandwidth of these top 30 middle relays. From the figure, there is a slight trend of the relay node with more bandwidth having a higher percentage of being selected as a middle relay. This is different from our analysis by source machines.

4) By target /16 subnet: Here, we analyze middle relays based on their /16 subnet prefix and the target websites' IP addresses. Figure 3.10 shows the top 30 subnets with their percentage of being selected. It is similar to previous analysis in that we have relays in subnets *163.172.*.**, *62.210.*.**, and *5.9.*.** having a higher chance of being selected than others. Subnet *163.172.*.** has the highest percentage at 0.91%. This means that some subnets are more popular than others regardless of the client IP address or the target IP address.

5) By AS: We further look into middle relays at the AS level. Figure 3.11 shows the top 30 ASes that have the highest percentage of number of middle relays that are in that AS. Figure 3.12 shows the top 30 ASes being selected in all circuits. *AS16276* has the highest percentage of 16% being selected with only less than 8% of middle relays in it. On the contrary, *AS3320* contains nearly 7% middle relays with less than 0.9% chance being selected. Some ASes in Figure 3.11 are not even in Figure 3.12. Hence, this shows that some ASes are more popular than others not because they have more middle relays in them.

3.4.4 Analysis of Exit Relays

In this section, we analyze the popularity of relays chosen as exit relays in all the circuits created by the 44 machines.

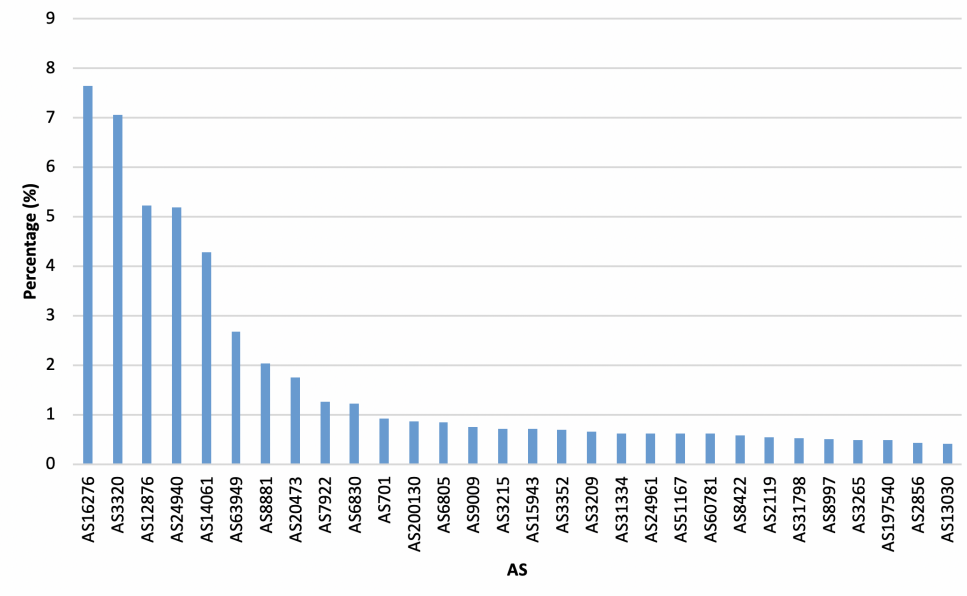


Figure 3.11: Percentage of # of middle relays that is in an AS.

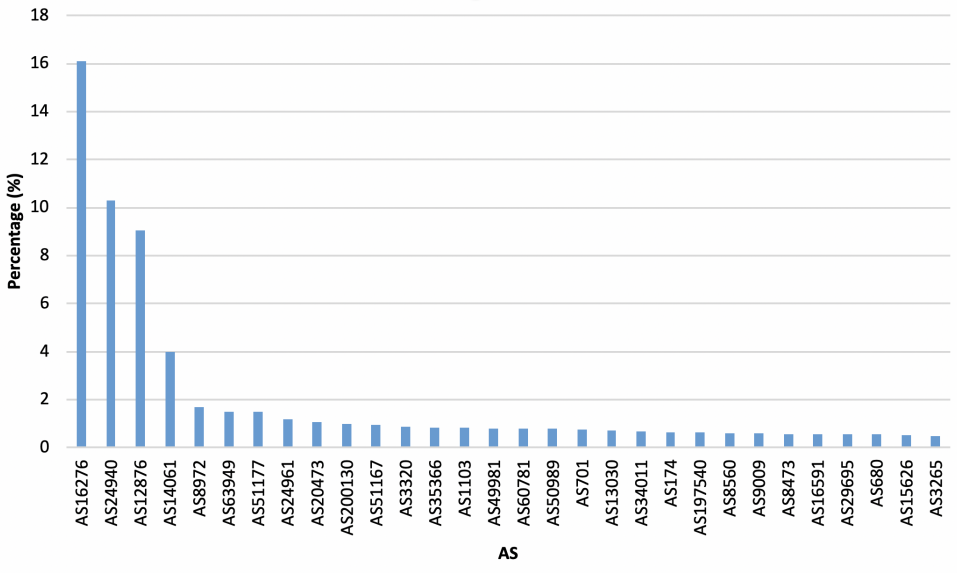


Figure 3.12: Percentage of an AS shown as Middle relay of all circuits.

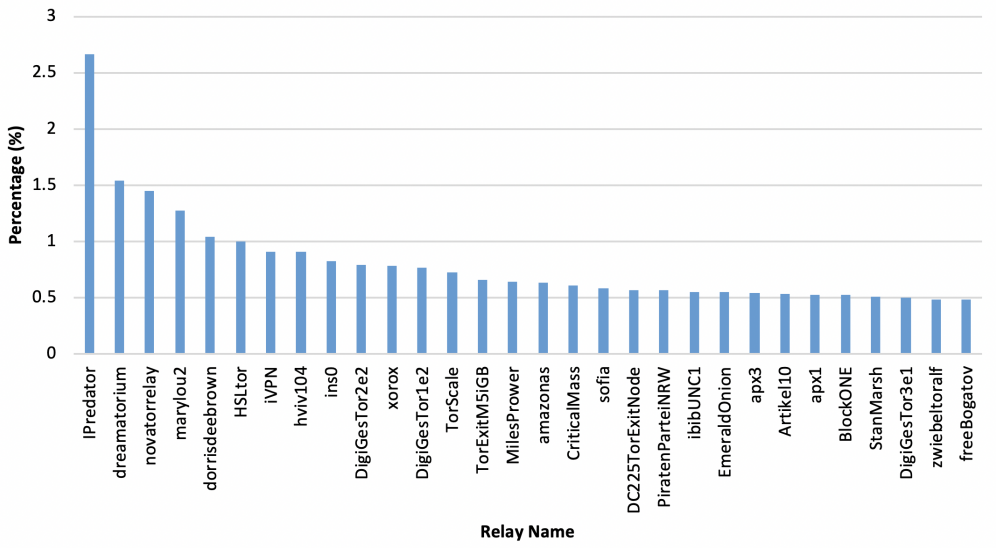


Figure 3.13: Point of view of all machines. The percentage of times a relay has been used as an exit relay for all circuits. This shows the top 30 most-used relays.

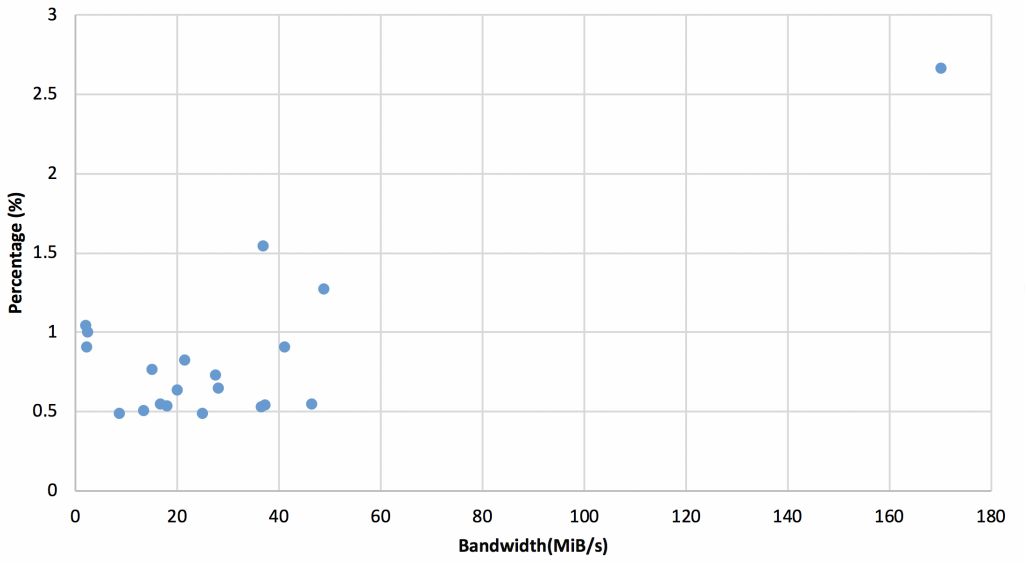


Figure 3.14: Point of view of all machines. The percentage of times a relay is selected as the exit relay in a circuit and that relay's corresponding bandwidth.

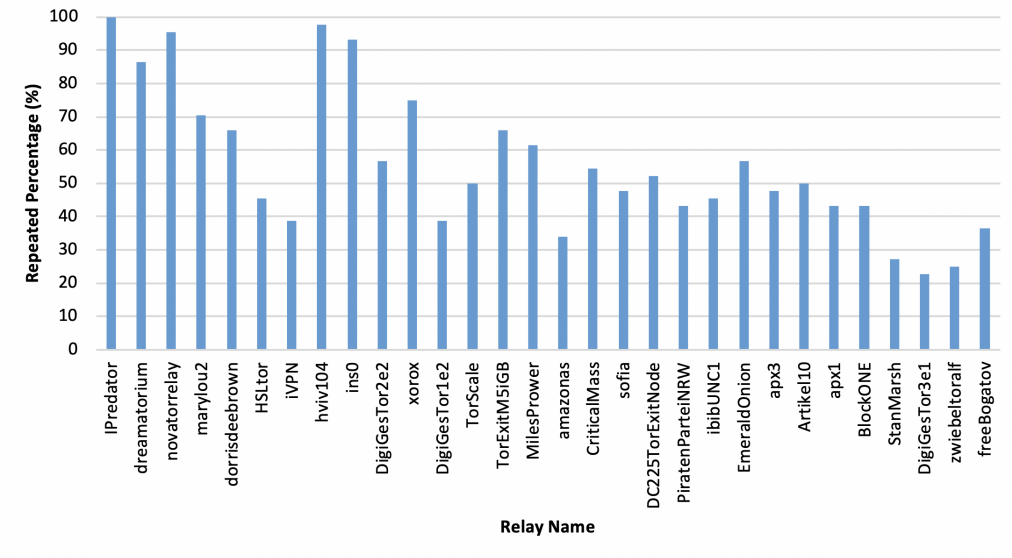


Figure 3.15: Point of view of all machines. The repeated percentage for each relay is chosen as an exit relay. This shows the top 30 most-used relays.

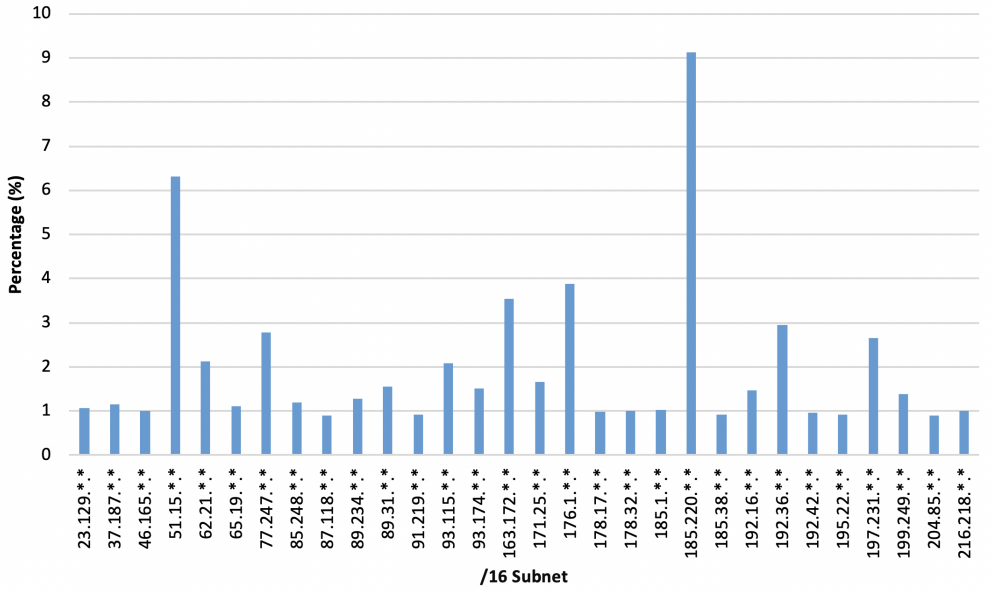


Figure 3.16: Point of view of all source machines. The percentage of selection of the top 30 most-selected /16 subnet IP addresses for exit relays.

1) By source IP address: We first look at the popularity of exit relays from the point of view of the source machines (clients). Figure 3.13, Figure 3.14, and Figure 3.15 are the analysis results when considering all our source machines together. From Figure 3.13, we can see the 30 most often used exit relays and the percentage that they have been selected during all the visits. In the figure, the relay named *IPredator* with IP address *197.231.221.211* has the highest percentage, 2.67%, of being used as an exit relay among all the relays. That is about 14 times higher than the average percentage 0.20% which we mentioned in Section 3.4.1. In the results of top relays in the Tor circuits when considering only one single source machine, there are relays being selected as exit relays more often than other relays. Also these source machines share several exit relays in their own top 30 exit relays list among all Tor circuits, as shown in Figure 3.15, such as *novatorrelay* with IP address *93.174.93.71* and *hviv104* with IP address *192.42.116.16*. These exit relay nodes appear in more than 90% of the top 30 exit relays lists from all source machines. This means that, over all the relays, some of the relays have a higher chance to be selected when Tor builds a circuit. This further confirms their popularity. Figure 3.14 shows the % of a relay selected as exit nodes during all the visits of one machine to websites and the bandwidth of the relays. From the figure, it can be seen that a relay node with more bandwidth has a better chance to be selected as an exit node during circuit building.

2) By source /16 subnet: Instead of analyzing each exit relay by their IP address, we now group the exit relays' IP address in /16 subnets. Figure 3.16 shows the 30 most-often-used exit relays with same /16 subnet prefix and the percentage of circuits which have that relay as an exit relay. The relay nodes with subnet *51.15.*.** and *185.220.*.** stand out in the figure. *51.15.*.** has a percentage of 6.32% and *163.172.*.** has a percentage of 5.44%. *185.220.*.** is the most popular one with a percentage of 9.13%. Also, *176.1.*.** and *163.172.*.** have the higher percentage. Hence, with results from all these figures, there are several /16 subnet prefixes that

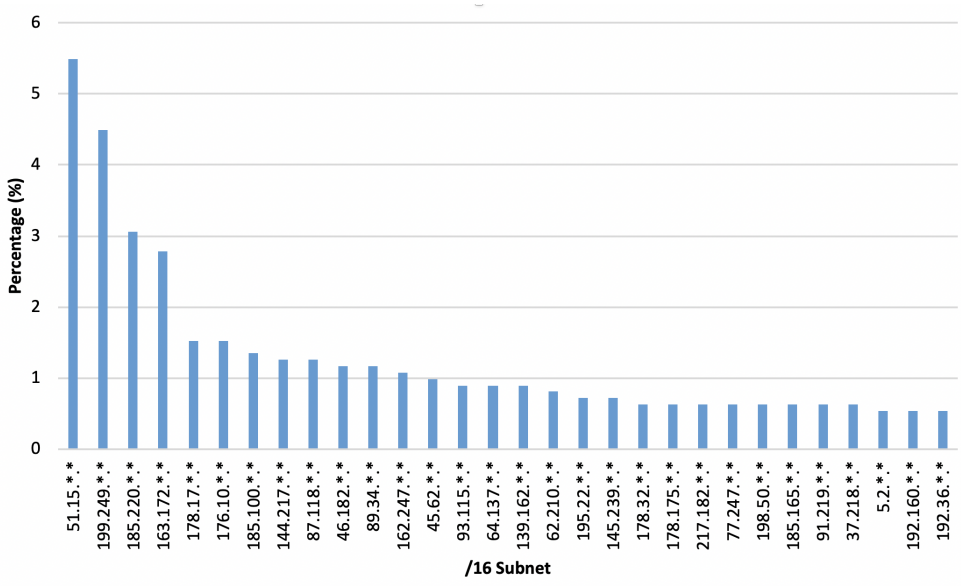


Figure 3.17: Percentage of # of exit relays in a /16 subnet.

have a better chance to be chosen as exit nodes. When analyzing /8 and /24 subnets, we see a similar result. In Figure 3.17, we show the percentage of number of exit relays under a /16 subnet. For example, *51.15.*.** has 5.5% of all exit relays in it. However, comparing Figure 3.17 with Figure 3.16, we can see that a /16 subnet may not have a higher chance to be selected even though it has a high percentage of number exit relays in it. From Figure 3.17, *199.249.*.** subnet contains 4.5% exit relay. However it only has less than 1.5% chance to be selected.

3) By target IP address: We now analyze the popularity of relays from the point of view of the target websites. Figure 3.18 lists the top 30 most often selected exit relays' names from all source machines to the website *151.101.130.167*. The relay *IPredator* with IP address *197.231.221.211* is the one with the largest percentage of being selected at 1.02%. Figure 3.19 shows the percentages of a relay being used as an exit node and its bandwidth during our experiments. *IPredator* has a bandwidth of 175.13 MiB/s with the highest percentage of 1.02%. The relay named *marylou1b* with IP address *89.234.157.254* has a bandwidth of 41.65 MiB/s with the second highest percentage of 0.56%. From that figure, it can be seen that a relay with higher

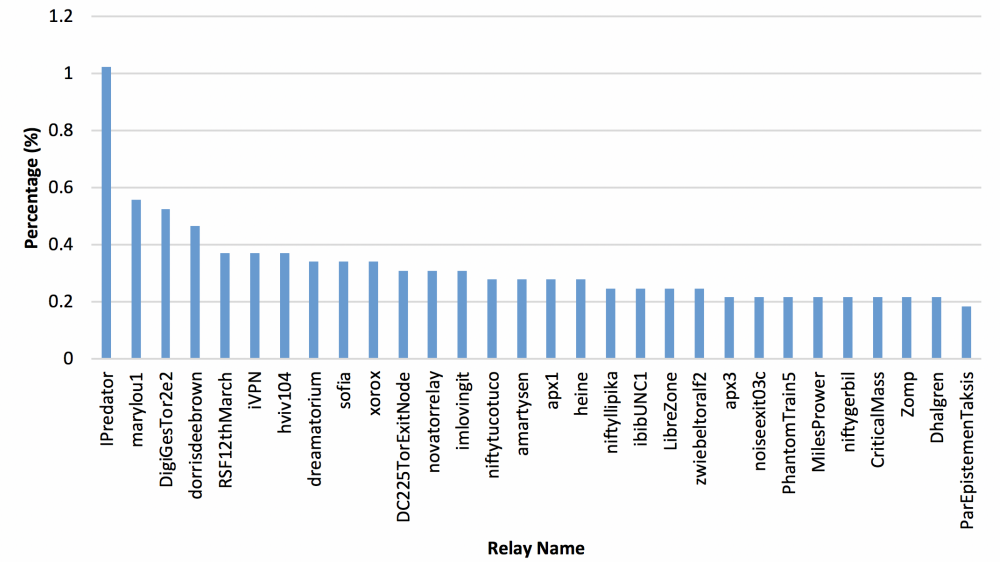


Figure 3.18: Point of view of one target destination IP address 151.101.130.167. The % of selection of the 30 most often selected Tor relays as the exit relays in circuits.

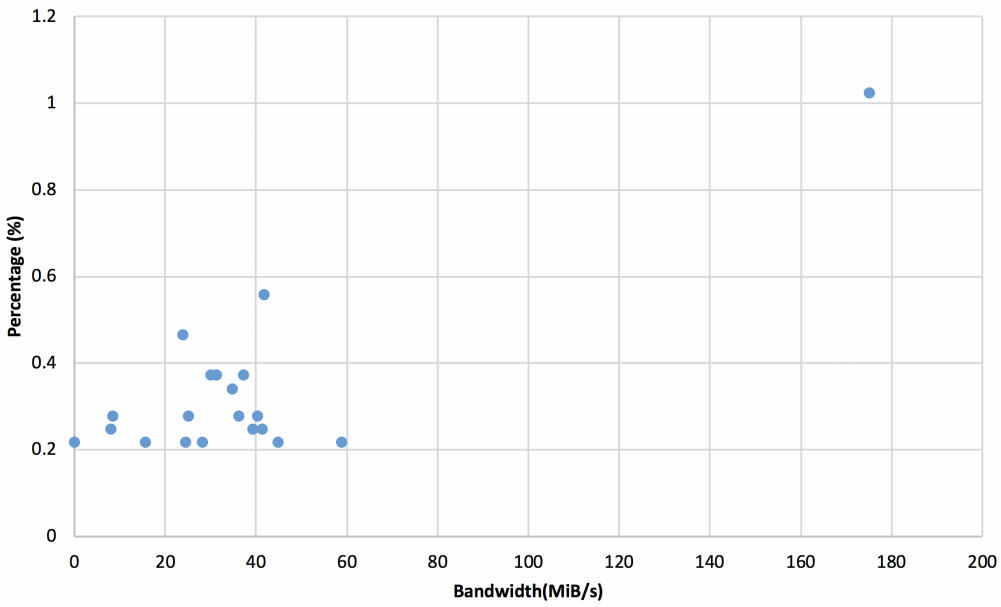


Figure 3.19: Point of view of one target destination IP address 151.101.130.167. The percentage of selection of the top 30 most often selected Tor relays as the exit relays in circuits and these relays' corresponding bandwidth.

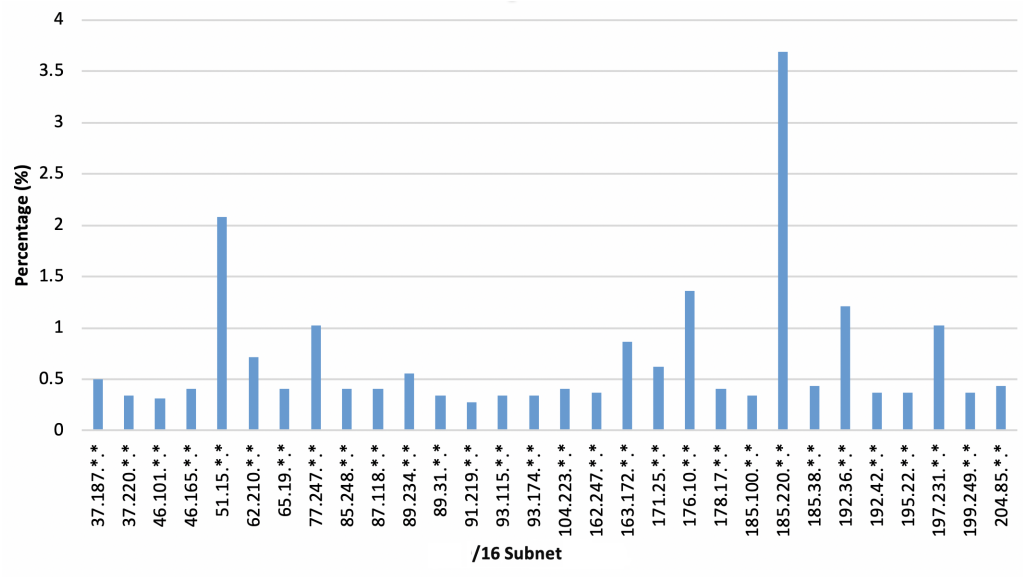


Figure 3.20: Point of view of target IP address 151.101.130.167. The percentage of the top 30 most often used relays being selected as exit relays, grouped by /16 subnets.

bandwidth has a higher chance of being selected.

4) By target /16 subnet: Figure 3.20 shows the top 30 subnets with /16 prefix along with the percentage of being selected as an exit relay that is within the subnet. From Figure 3.20, we see that *185.220.*.** has the highest percentage at 3.69%. *185.220.*.** is also in the list of top 30 exit relays from all source machines in Figure 3.16. *51.15.*.** is also one of the subnets that appeared in both figures. It has a percentage of 2.08% in Figure 3.20. This result again shows that some subnets have a higher percentage of being selected than other subnets.

5) By AS: We analyze exit relays in AS level now. Figure 3.21 and Figure 3.22 show the top 30 AS that have the highest percentage of number of exit relays that is in an AS and the top 30 AS being selected in all circuits respectively. In Figure 3.22, we can see AS like *AS12876*, *AS200052* are more popular than other AS. *AS12876* has the highest percentage of more than 12% being selected even though only 8% of exit relays are in that AS. Some ASes in Figure 3.21, such as *AS63949*, are not even in Figure 3.22. This leads us to the conclusion that there are certain ASes that are more popular than other ASes and this is not proportional to the number of exit

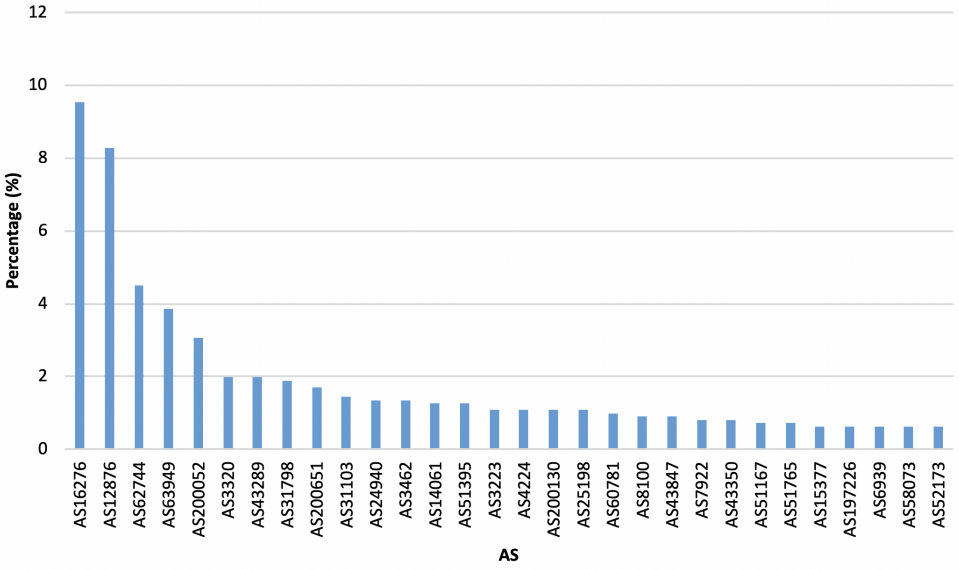


Figure 3.21: Percentage of # of exit relays that is under an AS.

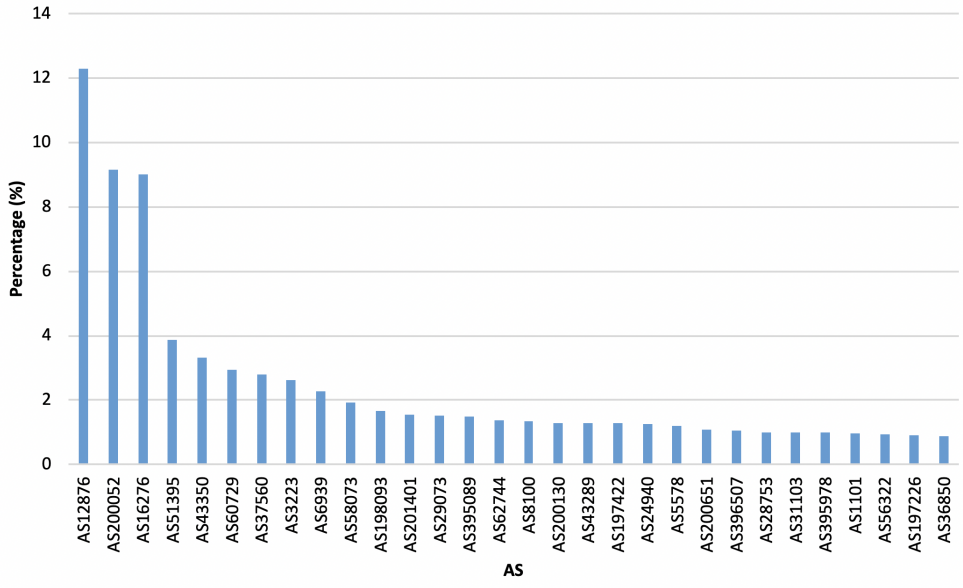


Figure 3.22: Percentage of exit relays in an AS amongst all circuits.

Table 3.2: Results of comparing Tor relay nodes' IP address and target websites' IP address in the same circuit to determine if they are in the same subnet prefix or AS number. The total number of circuits is 145,918.

	Entry vs Target	Entry vs Middle	Entry vs Exit	Middle vs Target	Middle vs Exit	Exit vs Target
/24 subnet prefix	0	0	0	2	0	1
/16 subnet prefix	104	0	0	8	0	10
/8 subnet prefix	3,906	6,750	10,456	2,769	8,598	3,792
AS	150	24,439	16,866	287	11,308	36

relays in them.

3.4.5 Overall

We now provide a more holistic view of our dataset. We compare the Tor entry guard, middle relay and exit relay in a circuit to see if they are in the same subnet prefix (/24, /16, and /8) and same AS. We also compare the relays and the target website IP address. This comparison is done for each circuit built. Table 3.2 shows the results of our comparison: there is not much overlap at the /24 subnet, but at the /16 subnet, 104 pairs of entry guard IP address and target IP address are in the same /16 subnet. This could lead to correlation attacks launched to determine who the user is. At a /8 subnet prefix and AS level, there are more pairs that match. This is expected, but could be an issue if an adversary controls a large swath of IP address space. Looking at the AS numbers, about 11% of all circuits could be compromised as these circuits have the client or entry relay and the target or exit relay in the same AS. This is a significant number and shows that ASes can correlate clients and targets. This does not include Internet Exchange Points, and we expect this is worse when these are considered.

3.5 Related Work

Tor was introduced and began operating in 2003 [39], providing service that enabled users to access the Internet anonymously [37]. When communicating with others, Tor clients choose a three-hop circuit from the set of available volunteer relays in the

network. Tor allows researchers to the Tor network data such as relay bandwidth, the number of active Tor relays, etc, through the Tor Metrics Portal [33]. Although Tor provides anonymous service, Tor users are vulnerable to an adversary that can observe some parts of the Tor relays [46]. Tor can also be blocked since all the Tor relays are public information [47, 48]. We used 44 independent machines located in different areas of the world to connect to popular websites through Tor.

Adversaries can exploit the nature of Internet routing by performing network traffic analysis [36, 49, 50] to increase the chance of observing users' communications traffic. They gain the visibility of Tor traffic either by compromising Tor relays, or by invading and manipulating underlying network communication like the Autonomous Systems (ASes) [46, 51–53]. If an attacker can observe the traffic from both the client to the entry guard and the exit relay to the server, then the leaked information, including the packet timing and sizes, is enough for attackers to infer the identities of the clients and servers from timing analysis [54]. This is a correlation attack [46, 55]. Our results show that some relays are much more popular than others. Moreover, correlation attacks could be performed as some Tor entry guards and exit relays/target websites are in the same /8 or /16 subnets.

Tor's path selection algorithm uses the estimated bandwidth of the nodes as a central feature. To mitigate the threat of AS-level adversaries, AS-aware path selection algorithms were proposed that consider the bandwidth and IP address when choosing relays while creating Tor circuits [56, 57]. They attempted to infer AS path from incomplete knowledge of the Internet topology and tried to avoid picking entry-exit pairs routing through the same AS or that may be subject to correlation attacks. This minimizes the amount of information gained by the adversary. Our work provided the list of popular relays that will benefit selection algorithm design by comparing theoretical analysis with our results.

3.6 Conclusion

We provide a comprehensive analysis of the popularity of Tor relays. Our dataset consists of Tor relay nodes, collected by visiting the Alexa top 100 websites through the Tor network for 5 months, by using 44 different source machines. Our dataset records the information of each Tor relay in circuits: the relay node IP address, fingerprint, geolocation, and advertised bandwidth. Our dataset also contains the IP address of the source machine used and the target website. Then we analyze the dataset from many different perspectives: by source machine, by target IP address, by IP address, by /8 subnet, by /16 subnet, by /24 subnet and by AS.

The results show that some Tor relays and some subnets (either /8, /16, or /24) are more popular when being selected as middle relays or exit relays. From analysis of middle relays, the Tor relay named *TotorBE1* with IP address *5.39.33.176*, is 3 times more likely to be chosen than other relays and 10 times more likely to be chosen than an average relay. Our data also show that the bandwidth of a relay does not affect its chance of being selected as a middle relay node when a Tor client builds a circuit. When grouping Tor relays' IP addresses into /16 subnets, some subnets, such as *51.15.*.** and *163.172.*.**, are more popular than other subnets. Additionally, our analysis indicates that the chance of a /16 subnet being selected is not proportional to the number of relays it contains. For example, *62.210.*.** has nearly 0.6% middle relays however it has a 2.1% chance of being selected in circuits. When it comes to AS level, *AS16276* stand out. It has less than 8% of middle relays in it while it has a chance of 16% being selected.

From analysis of exit relays, the Tor relay named *IPredator* with IP address *197.231.221.211* and the Tor relay named *dreamatorium* with IP address *89.31.57.58* are 6 times more likely to be chosen as an exit relay, compared with other relays. There also seems to be a correlation between a relay's bandwidth and its popularity as an exit relay. Similarly, in /8, /16, and /24 subnets, we found that some subnets

like *185.220.*.** and *51.15.*.** are 6 times more likely to be selected as exit relays than other subnets. From the results, we can see that a /16 subnet may have a higher chance to be selected even though it has a lower percentage of number exit relays in it. Like *185.220.*.**, it has 3.15% exit relays and it has a percentage of 9.13% of being selected. At the AS level, we can see ASes like *AS12876*, *AS200052* are more popular than other AS. *AS12876* has more than 12% chance of being selected even though only 8% of exit relays are in the AS.

For future work, we plan to explore more aspects of Tor relays in terms of popularity at the geolocation level. We will further explore the correlation between bandwidth and popularity as a middle relay or exit relay. Based on this result, we will also find ways to perform correlation attacks on Tor and learn how to make the relay selection algorithm more balanced.

Acknowledgments: We thank Mr. Ippei Okamura for his help with collecting the data, and Google for providing us Google Cloud credits to increase our number of vantage points.

CHAPTER IV

Anonymous Networks Website Fingerprinting

4.1 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 [58, 59] is one of the most popular low-latency anonymity-providing network. It is used by millions of people daily [33]. Tor protects users' privacy through a telescoping three-hop circuit and encrypting 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%, 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 [60]. A number of studies on this topic have been released since then [61–63], showing high accuracy in predicting websites in both the open and closed world models.

All 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 focus on the situation the victim visits a second page before the first one finishes loading (overlapping visits). When two website visits are overlapping or part of the network trace is missing (either the beginning or the end), the website fingerprinting accuracy falls dramatically. Hence, we propose a new algorithm "sectioning" algorithm to deal with these overlapping traces and partial traces.

The contributions of this paper are summarized as follows.

- **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 parts 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.

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

This paper is structured as follows: in Section 4.2, we give the related background and terminology of this paper. We propose a new "sectioning" algorithm to improve the accuracy in overlapping traces in Section 4.3 and in partial traces in Section 4.4. We conclude and provide avenues for future work in Section 4.5.

4.2 Background

- **Definitions**. We first define some terms we use throughout the paper.
  - 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.
  - 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.
  - Partial Trace. A network traffic trace with part of it is missing (either the beginning or the end).
- **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 4.1 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.
- **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 at-

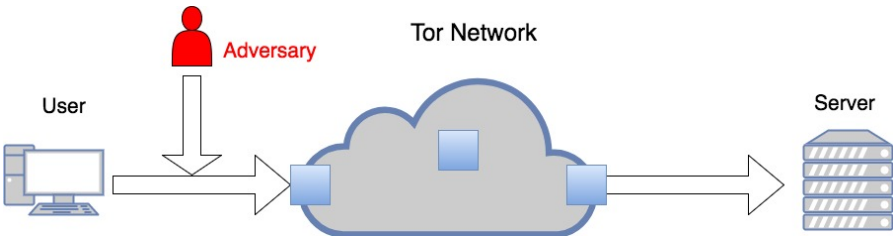


Figure 4.1: Threat Model.

tack 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 4.2 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.

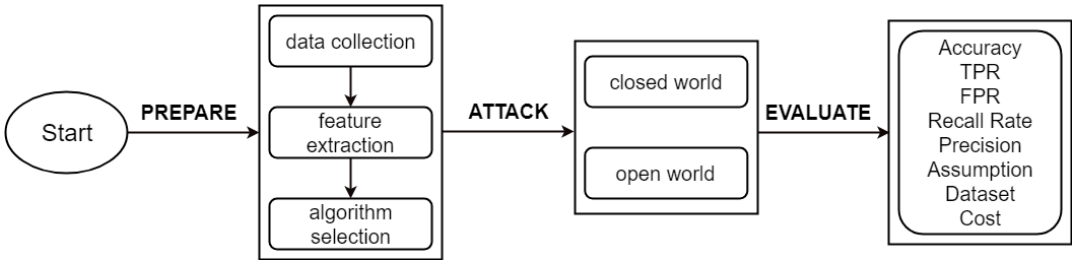


Figure 4.2: Steps of launching and evaluating a website fingerprinting attack.

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.

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 [61, 63–72] 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 [61], our experiments in Section 4.3, and Section 4.4 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.

4.3 Analysis of Overlapping Traces

4.3.1 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 [63] has looked at overlapping website visits. Figure 4.3 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 4.4 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.

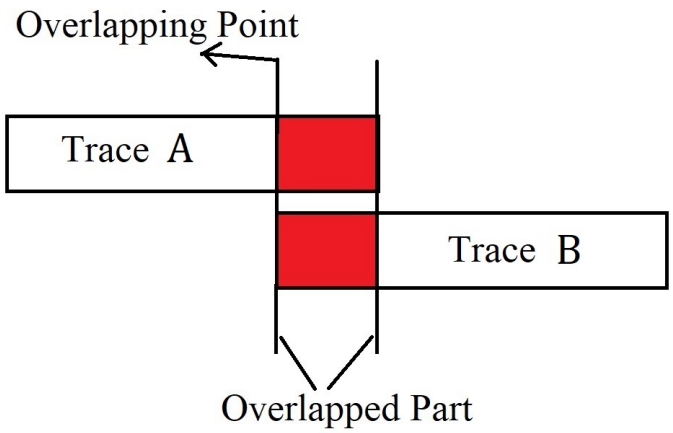


Figure 4.3: Two website traces A and B overlap.

4.3.2 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

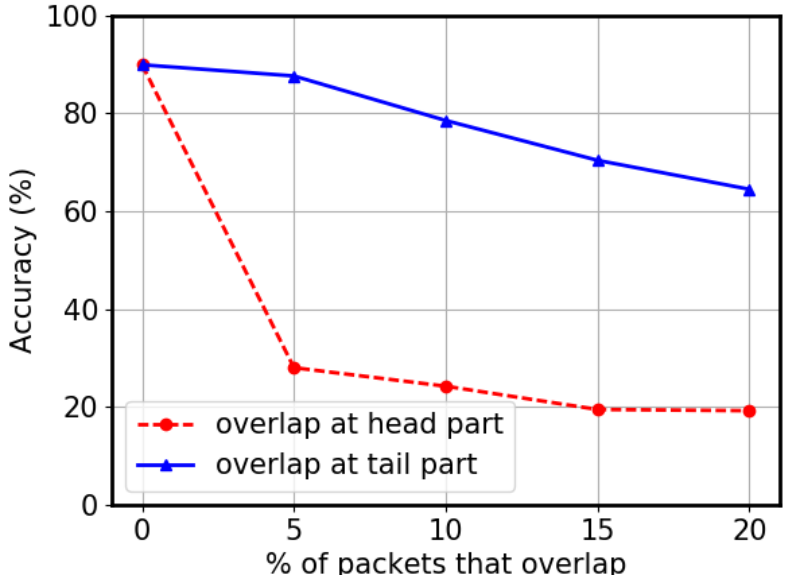


Figure 4.4: Prediction accuracy as more packets overlap in the two traces.

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 4.5 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 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,

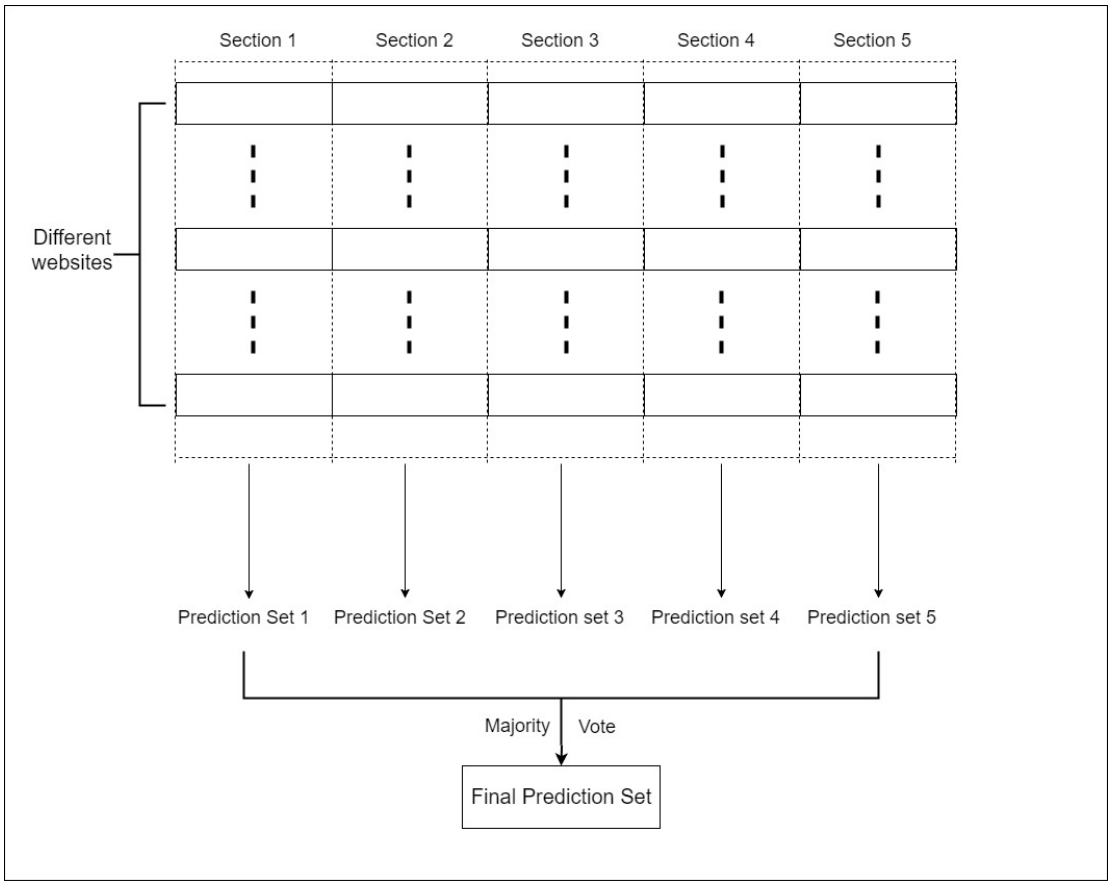


Figure 4.5: Outline of sectioning algorithm.

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 4.5 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 4.3, 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.

4.3.3 Experiment Setup

Figure 4.6 shows our sectioning algorithm. The steps are as follows: 1) split dataset into training and testing sets (Figure 4.6(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 4.6(a)); 3) Partition into n sections for both training and testing sets accordingly (Figure 4.6(b)); 4) Apply machine learning classifier (for example, k-NN) to each section ((Figure 4.6(c)); 5) A majority vote will be performed for the predictions from the different sections (Figure 4.6(d)); 6) Repeat to do 10-fold

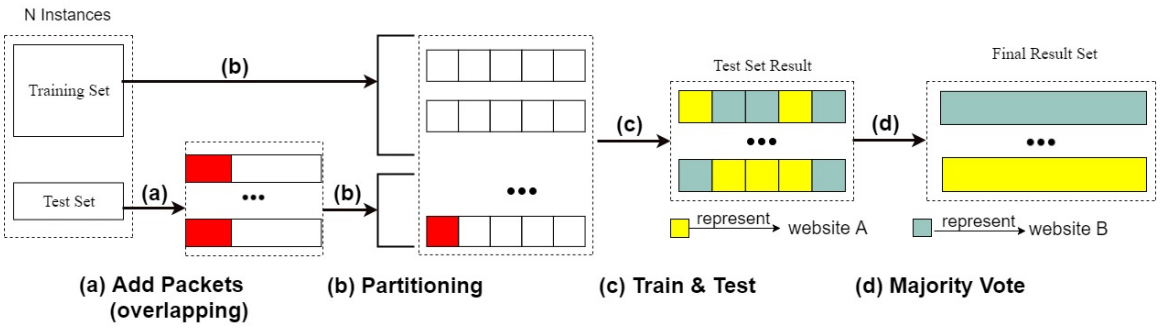


Figure 4.6: Overview of the sectioning algorithm.

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 [61]. 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 4.3 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 [63] 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 4.3.5. Figure 4.7 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.

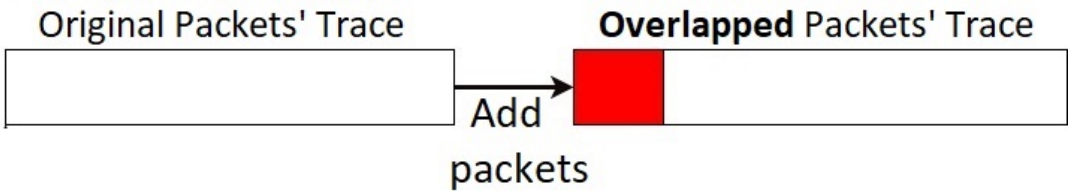


Figure 4.7: Simulate overlapping: add packets to the beginning of trace.

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 [61].

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 website or different websites. Figure 4.5 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.

4.3.4 Results

a) Sectioning by number of packets: Figure 4.8 and Figure 4.9 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 4.8 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 4.9, 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 [73] and later in

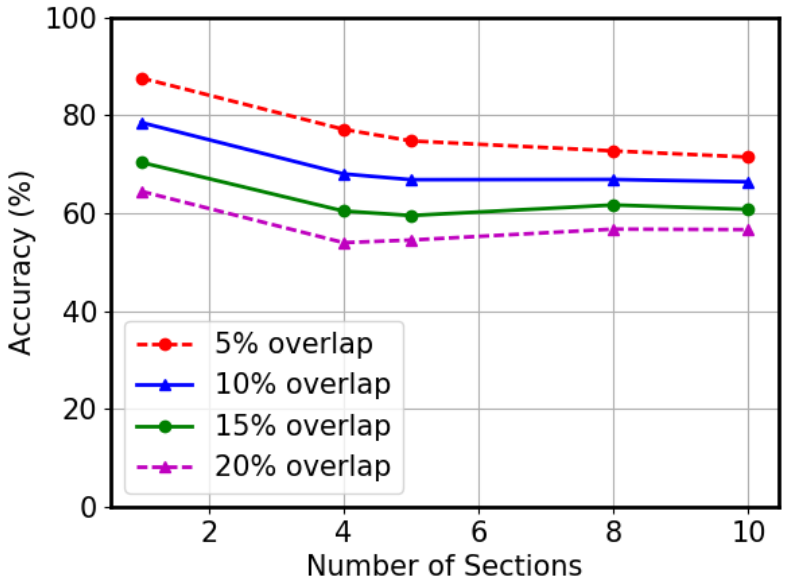


Figure 4.8: Prediction accuracy of website A with varying number of sections and overlap %, using **a) sectioning by number of packets**.

Section 4.4, 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 4.10 and Figure 4.11 show the accuracy result in correctly predicting websites A and B when using sectioning by time duration. Figure 4.10 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 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 4.11, 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

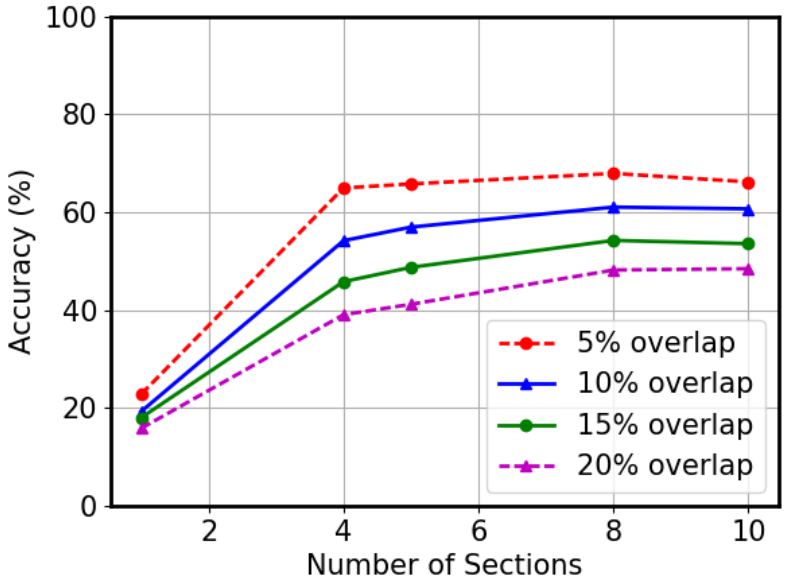


Figure 4.9: Prediction accuracy of website B with varying number of sections and overlap %, using **b) sectioning by number of packets**.

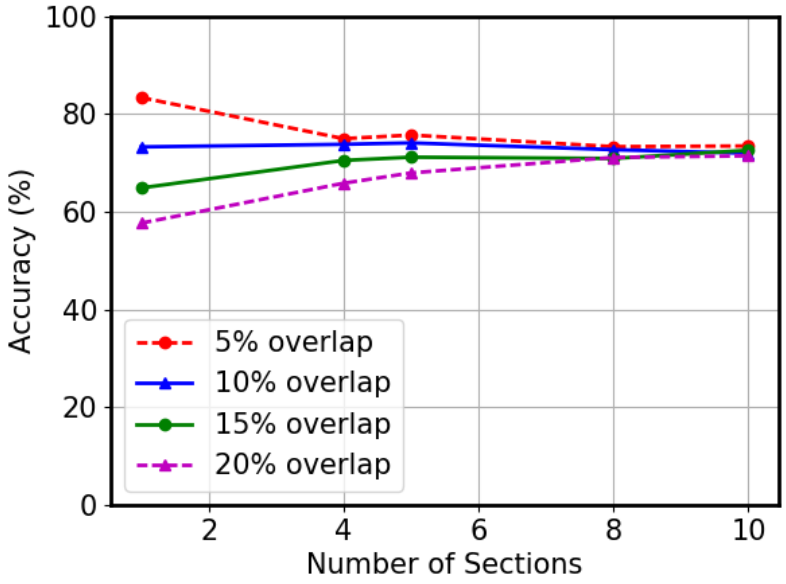


Figure 4.10: Prediction accuracy of website A with varying number of sections and overlap %, using **b) sectioning by time duration**.

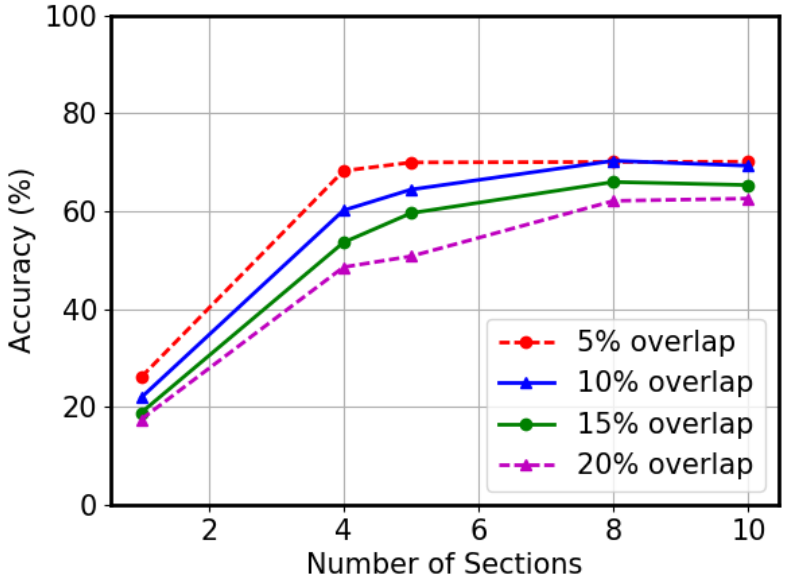


Figure 4.11: Prediction accuracy of website B with varying number of sections and overlap %, using **b) sectioning by time duration**.

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.

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).

4.3.5 Predicting Overlapping Point

Previous work [63] showed that the accuracy to find the split point in overlapped trace is 32%. 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 4.12 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.

4.3.6 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.

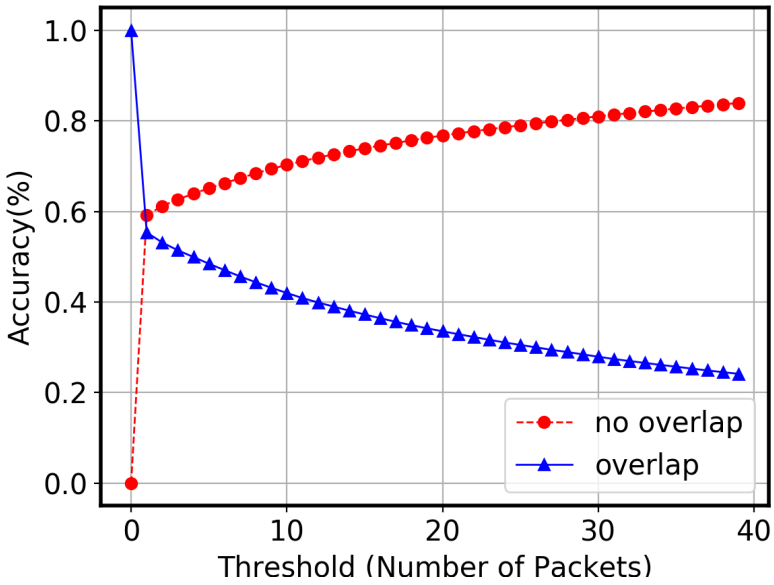


Figure 4.12: Prediction accuracy of the overlapping parts and non-overlapping parts.

4.4 Analysis of Partial Traces

4.4.1 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 4.13 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 [73]. The figure also shows that as the percentage of the trace available decreases, the accuracy decreases significantly.

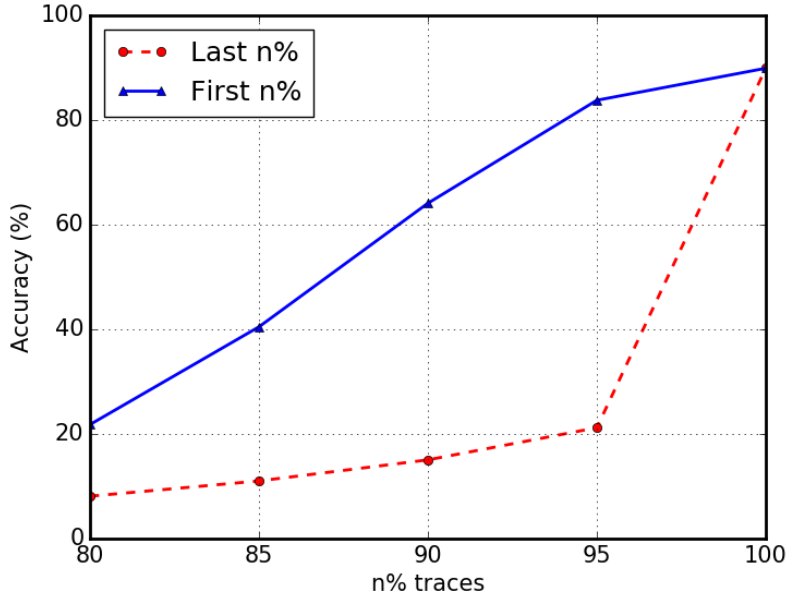


Figure 4.13: Accuracy of website fingerprinting when observing different percentages of network traffic traces.

4.4.2 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.

4.4.3 Results

Figure 4.14 and Figure 4.15 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 4.14, 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 4.15 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%.

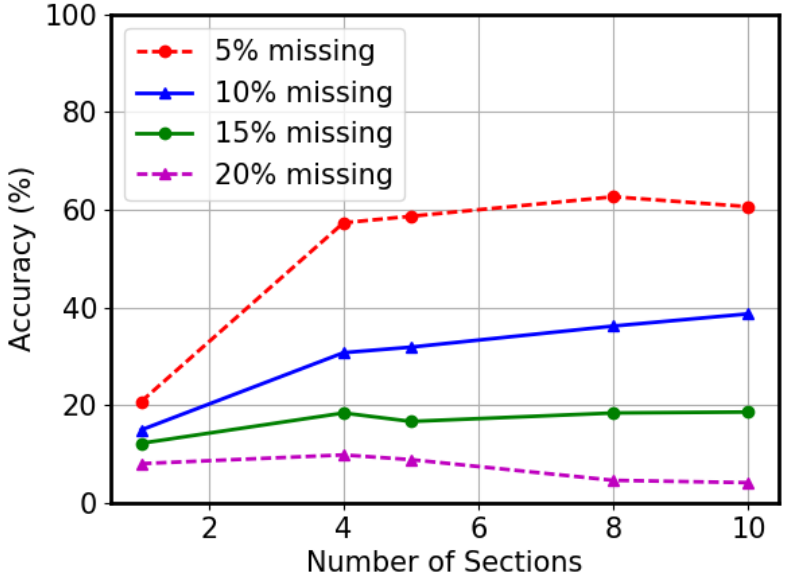


Figure 4.14: Prediction accuracy when varying the number of sections and the % of missing packets from the beginning.

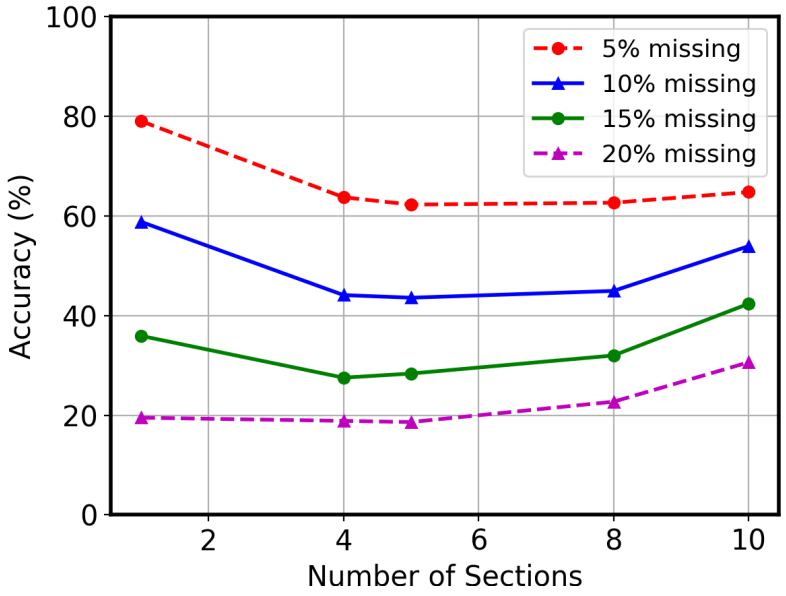


Figure 4.15: Prediction accuracy when varying the number of sections and the % of missing packets from the end.

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

4.5 Summary

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

1. 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 overlap. Moreover, we have showed 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.

CHAPTER V

Conclusion and Future Work

In this dissertation, we analyze the privacy of online and offline systems by focusing on Tor networks and smartphones. First of all, for privact analysis of the offline systems, we design and develop an attack to accurately predict the passcode entered by a victim on her smartphone. The attack relies on recording a video of the victim using a common smartphone in a public environment. Our algorithm achieves an overall accuracy of 92.5%. This result demonstrates that online shoulder-surfing attacks on PIN-based authentication are possible. Also, these results show that choosing a good random PIN cannot prevent this type of online attack as the algorithm can still predict the PIN entered.

Then, we look at Tor systems and try to find its relay popularity. Our results show that some Tor relays, subnets (either /8, /16, or /24) and ASes are more popular when being selected as middle relays or exit relays. Our data also show that the bandwidth of a relay does not affect its chance of being selected as a middle relay node when a Tor client builds a circuit. However, there seems to be a correlation between a relay's bandwidth and its popularity as an exit relay. For future work, we plan to explore more aspects of Tor relays in terms of popularity at the geolocation level. We will further explore the correlation between bandwidth and popularity as a middle relay or exit relay. Based on this result, we will also find ways to perform correlation attacks on Tor and learn how to make the relay selection algorithm more balanced.

Finally, for website fingerprinting on anonymous networks like Tor, We propose a "sectioning" algorithm to improve the accuracy in website prediction of two over-

lapping traces from 22.80% to 67.9% and partial traces from 20.76% to 62.66%. As the future work, we will test our algorithm in the open world setting and will consider the scenario when more than two pages are overlap. Moreover, we have showed some promising results in predicting exactly where two webpages overlap; we plan to investigate this further.

REFERENCES

[1] T. Chen, M. Farcasin, and E. Chan-Tin, "Smartphone passcode prediction," *IET Information Security*, 2018.

[2] "Tor: Overview." `https://www.torproject.org/about/overview.html.en`, 2018. [Online; accessed 20-Aug-2018].

[3] T. Chen, W. Cui, and E. Chan-Tin, "Measuring tor relay popularity," in *International Conference on Security and Privacy in Communication Networks*, Springer, 2019.

[4] W. Cui, T. Chen, C. Fields, J. Chen, A. Sierra, and E. Chan-Tin, "Revisiting assumptions for website fingerprinting attacks," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pp. 328–339, ACM, 2019.

[5] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh.," in *USENIX Security Symposium*, vol. 2001, 2001.

[6] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm, "Seeing double: Reconstructing obscured typed input from repeated compromising reflections," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 1063–1074, ACM, 2013.

[7] R. Raguram, A. M. White, Y. Xu, J.-M. Frahm, P. Georgel, and F. Monrose, "On the privacy risks of virtual keyboards: automatic reconstruction of typed input from compromising reflections," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 3, pp. 154–167, 2013.

[8] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 1, p. 3, 2009.

[9] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "Accessory: password inference using accelerometers on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, p. 9, ACM, 2012.

[10] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion.," *HotSec*, vol. 11, pp. 9–9, 2011.

[11] L. Simon and R. Anderson, "Pin skimmer: Inferring pins through the camera and microphone," in *Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, SPSM '13, (New York, NY, USA), pp. 67–78, ACM, 2013.

[12] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd, "Reducing shoulder-surfing by using gaze-based password entry," in *Proceedings of the 3rd symposium on Usable privacy and security*, pp. 13–19, ACM, 2007.

[13] M. Eiband, M. Khamis, E. von Zezschwitz, H. Hussmann, and F. Alt, "Understanding shoulder surfing in the wild: Stories from users and observers," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 4254–4265, ACM, 2017.

[14] "Identities snatched in blink of eye," 2004.

[15] J. Wagstaff, "Shoulder-surfing: the old new phishing," 2005.

[16] R. Hoyle, R. Templeman, D. Anthony, D. Crandall, and A. Kapadia, "Sensitive lifelogs: A privacy analysis of photos from wearable cameras," in *Proceedings of the 33rd Annual ACM conference on human factors in computing systems*, pp. 1645–1648, ACM, 2015.

[17] A. De Luca, M. Harbach, E. von Zezschwitz, M.-E. Maurer, B. E. Slawik, H. Hussmann, and M. Smith, "Now you see me, now you don't: protecting smartphone authentication from shoulder surfers," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2937–2946, ACM, 2014.

[18] V. Roth, K. Richter, and R. Freidinger, "A pin-entry method resilient against shoulder surfing," in *Proceedings of the 11th ACM conference on Computer and communications security*, pp. 236–245, ACM, 2004.

[19] J. Schiff, M. Meingast, D. K. Mulligan, S. Sastry, and K. Goldberg, "Respectful cameras: Detecting visual markers in real-time to address privacy concerns," in *Protecting Privacy in Video Surveillance*, pp. 65–89, Springer, 2009.

[20] F. Schaub, R. Deyhle, and M. Weber, "Password entry usability and shoulder surfing susceptibility on different smartphone platforms," in *Proceedings of the 11th international conference on mobile and ubiquitous multimedia*, p. 13, ACM, 2012.

[21] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget, "Design and evaluation of a shoulder-surfing resistant graphical password scheme," in *Proceedings of the working conference on Advanced visual interfaces*, pp. 177–184, ACM, 2006.

[22] H. Zhao and X. Li, "S3pas: A scalable shoulder-surfing resistant textual-graphical password authentication scheme," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, vol. 2, pp. 467–472, IEEE, 2007.

[23] G. Ye, Z. Tang, D. Fang, X. Chen, K. I. Kim, B. Taylor, and Z. Wang, "Cracking android pattern lock in five attempts," 2017.

[24] E. Von Zezschwitz, A. De Luca, P. Janssen, and H. Hussmann, "Easy to draw, but hard to trace?: On the observability of grid-based (un) lock patterns," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 2339–2342, ACM, 2015.

[25] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, "Beware, your hands reveal your secrets!," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 904–917, ACM, 2014.

[26] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind recognition of touched keys on mobile devices," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1403–1414, ACM, 2014.

[27] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[28] OpenCV, "`http://opencv.org/`," 2017.

[29] O. C. C. Training, "`http://docs.opencv.org/2.4.13.2/doc/user_guide/ug_traincascade.html`," 2017.

[30] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2001.

[31] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[32] Facebook Hidden Service. https://blog.torproject.org/facebook-hidden-services-and-https-certs, 2014.

[33] T. M. Portal. https://metrics.torproject.org/, 2017.

[34] Tor. https://www.torproject.org/, 2017.

[35] G. O'Gorman and S. Blott, "Large scale simulation of tor," in *Annual Asian Computing Science Conference*, pp. 48–54, Springer, 2007.

[36] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against Tor," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)*, (Washington, DC, USA), October 2007.

[37] L. Øverlier and P. Syverson, "Locating hidden servers," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, IEEE CS, May 2006.

[38] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell counter based attack against tor," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 578–589, ACM, 2009.

[39] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, (Berkeley, CA, USA), pp. 21–21, USENIX Association, 2004.

[40] P. F. Syverson, G. Tsudik, M. G. Reed, and C. E. Landwehr, "Towards an analysis of onion routing security," in *Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[41] T. protocol specifications. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt, 2018.

[42] M. Imani, M. Amirabadi, and M. Wright, "Modified relay selection and circuit selection for faster tor," *CoRR*, vol. abs/1608.07343, 2016.

[43] "Planetlab." https://www.planet-lab.org/.

[44] S. T. Controller. https://stem.torproject.org/, 2017.

[45] Alexa. http://www.alexa.com/topsites, 2017.

[46] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "Raptor: routing attacks on privacy in tor," in *24th USENIX Security Symposium (USENIX Security 15)*, pp. 271–286, 2015.

[47] "China blocking tor." https://blog.torproject.org/blog/china-blocking-tor-round-two.

[48] "Iran blocks tor." https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix.

[49] K. Loesing, S. J. Murdoch, and R. Dingledine, "A case study on measuring statistical data in the tor anonymity network," in *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)*, LNCS, Springer, January 2010.

[50] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, IEEE CS, May 2005.

[51] R. Anwar, H. Niaz, D. Choffnes, I. Cunha, P. Gill, and E. Katz-Bassett, "Investigating interdomain routing policies in the wild," in *Proceedings of the 2015 Internet Measurement Conference*, IMC '15, (New York, NY, USA), pp. 71–77, ACM, 2015.

[52] N. Feamster and R. Dingledine, "Location diversity in anonymity networks," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, (Washington, DC, USA), October 2004.

[53] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by internet-exchange-level adversaries," in *Proceedings of the 7th International Conference on Privacy*

Enhancing Technologies, PET'07, (Berlin, Heidelberg), pp. 167–183, Springer-Verlag, 2007.

[54] N. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on tor using long paths," in *Proceedings of the 18th USENIX Security Symposium*, August 2009.

[55] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on tor by realistic adversaries," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, (New York, NY, USA), pp. 337–348, ACM, 2013.

[56] R. Nithyanand, O. Starov, A. Zair, P. Gill, and M. Schapira, "Measuring and mitigating as-level adversaries against tor," *arXiv preprint arXiv:1505.05173*, 2015.

[57] M. Edman and P. F. Syverson, "AS-awareness in tor path selection," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009* (E. Al-Shaer, S. Jha, and A. D. Keromytis, eds.), pp. 380–389, ACM, 2009.

[58] Tor. https://www.torproject.org/, 2017.

[59] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[60] A. Hintz, "Fingerprinting websites using traffic analysis," in *Proceedings of the 2Nd International Conference on Privacy Enhancing Technologies*, PET'02, (Berlin, Heidelberg), pp. 171–178, Springer-Verlag, 2003.

[61] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at internet scale," in *Proceedings of the 23rd*

Internet Society (ISOC) Network and Distributed System Security Symposium (NDSS 2016), 2016.

[62] X. Cai, R. Nithyanand, and R. Johnson, "Cs-buflo: A congestion sensitive website fingerprinting defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, (New York, NY, USA), pp. 121–130, ACM, 2014.

[63] T. Wang and I. Goldberg, "On realistically attacking tor with website fingerprinting," in *Privacy Enhancing Technologies Symposium (PETS)*, 2016.

[64] L. Lu, E.-C. Chang, and M. C. Chan, *Website Fingerprinting and Identification Using Ordered Feature Sequences*, pp. 199–214. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[65] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11, (New York, NY, USA), pp. 103–114, ACM, 2011.

[66] X. Gong, N. Borisov, N. Kiyavash, and N. Schear, "Website detection using remote traffic analysis," in *Proceedings of the 12th International Conference on Privacy Enhancing Technologies*, PETS'12, (Berlin, Heidelberg), pp. 58–78, Springer-Verlag, 2012.

[67] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, (New York, NY, USA), pp. 201–212, ACM, 2013.

[68] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM*

Conference on Computer and Communications Security, CCS '12, (New York, NY, USA), pp. 605–616, ACM, 2012.

[69] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, (Berkeley, CA, USA), pp. 143–157, USENIX Association, 2014.

[70] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 263–274, ACM, 2014.

[71] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "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, (New York, NY, USA), pp. 227–238, ACM, 2014.

[72] S. E. Oh, S. Li, and N. Hopper, "Fingerprinting keywords in search queries over tor," *PoPETs*, vol. 2017, 2017.

[73] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 1187–1203, USENIX Association, Aug. 2016.

VITA

TAO CHEN

Candidate for the Degree of

Doctor of Philosophy

Dissertation: PRIVACY ANALYSIS OF ONLINE AND OFFLINE SYSTEMS

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the degree of Doctor of Philosophy in Computer Science at Oklahoma State University, Stillwater, Oklahoma, in JULY 2019

Completed the requirements for the Bachelor of Science in Communication Engineering at China Jiliang University, Hangzhou, Zhejiang, China in JUNE 2011

Experience:

1.Software Engineer FTE, State Street Technology (Zhejiang) Co., Ltd. (Mar. 2011 - Jun. 2014)
2.Teaching Assistant and Research Assistant, Oklahoma State University (Aug 2014 - Present)