TRAFFIC MONITORING AND ANALYSIS FOR SOURCE IDENTIFICATION

LIMING LU

B.Comp.(Hons.), National University of Singapore

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2010

Acknowledgements

I sincerely thank my supervisors, coauthors, friends and family for their continuous support to my PhD study.

Firstly, I thank my thesis advisors Dr Chan Mun Choon and Dr Chang Ee-Chien, for making my thesis possible. Their extraordinary capability in systematic problem formulation, as well as critical thinking and analysis, gave positive impact to my research. Their demand for high standard did open up my eyes to a new horizon, and effected on ensuring the quality of my publications. I thank my former supervisor Dr Ong Ghim Hwee, who inspired my initial research interests and coached me on research methodologies before his retirement. I thank Dr Zhou Jianying (from I^2R) who had selflessly been my unofficial advisor for over a year. He fostered initiation and creativity of students by encouraging and mentoring them to pursue their research interests.

Secondly, I thank my coauthors of research papers for their memorable contribution. My coauthors include: Dr Roland Yap from National University of Singapore (NUS); PhD students from NUS: Choo Fai Cheong, Fang Chengfang and Wu Yongzheng; graduates from NUS: Peng Song Ngiam and Viet Le Nhu; Dr Li Zhoujun from Beihang University, China; and Yu Jie from National University of Defense Technology, China. It was a pleasant experience working with them. The piece of work on fingerprinting web traffic over Tor presented in Chapter 8 was greatly benefited from the collaboration with Choo Fai Cheong. I thank reviewers of my publications for sharing their genuine remarks and giving expert suggestions.

Thirdly, I thank my peers, including members of the security research group (especially Fang Chengfang, Liu Xuejiao, Sufatrio, Wu Yongzheng, Xu Jia, Yu Jie, Zhang Zhunwang) and members of the networking research group (especially Chen Binbin and Choo Fai Cheong), because they extended my knowledge and exchanged sparkles of research ideas. I thank postgraduates of my batch (especially Ehsan Rehman, Muhammad Azeem Faraz, Pavel Korshunov, Tan Hwee Xian, Xiang Shili, Yang Xue and Zhao Wei), as they offered generous friendship and sympathy.

Lastly, I deeply thank my family (my parents, sister, lovely niece and my husband), for giving me tremendous support to my PhD study. They gave me the mental strength to endure all sorts of difficulties and to persevere till obtaining the PhD certificate. They also gave me financial support, which reduced my worries on financial burden. I am inspired by my husband's passion about research, who regards research as the first priority in life. He is diligent and determined to penetrate obstacles in research problems, with countlessly sleepless nights and missed or delayed meals.

I cannot fully express in this acknowledgement my gratitude towards all the people who played a part in my PhD life.

Contents

Summary

Traffic source identification aims to overcome obfuscation techniques that hide traffic sources to evade detection. Common obfuscation techniques include IP address spoofing, encryption together with proxy, or even unifying packet sizes. On one hand, traffic source identification provides the technical means to conduct web access surveillance so as to combat crimes even if the traffic are obfuscated. Yet on the other hand, adversary may exploit traffic souce identification to intrude user privacy by profiling user interests.

We lay out a framework of traffic source identification, in which we investigate the general approaches and factors in designing a traffic source identification scheme with respect to different traffic models and analyst's capabilities.

Guided by the framework, we examine three traffic source identification applications, namely, tracing back DDoS attackers, passively fingerprinting websites over proxied and encrypted VPN or SSH channel, and actively fingerprinting websites over Tor.

In the analysis of identifying DDoS attackers, we find out that with the information of network topology, it is unnecessary to construct packet marks with sophisticated structures. Based on this observation, we design a new probabilistic packet marking scheme that can significantly improve the traceback accuracy upon previous schemes, by increasing the randomness in the collection of packet marks and hence the amount of information they transmit.

We develop a passive website fingerprinting scheme applicable to TLS and SSH tunnels. Previous website fingerprinting schemes have demonstrated good identification accuracy using only side channel features related to packet sizes. Yet these schemes are rendered ineffective under traffic morphing, which modifies the packet size distribution of a source website to mimic some target website. However, we show that traffic morphing has a severe limitation that it cannot handle packet ordering while simultaneously satisfying the low bandwidth overhead constraint. Hence we develop a website fingerprinting scheme that makes use of the packet ordering information in addition to packet sizes. Our scheme enhances the website fingerprinting accuracy as well as withstands the traffic morphing technique.

Extending from the passive website fingerprinting model, we propose an active website fingerprinting model that can be applied to essentially any low latency, encrypted and proxied communication channel, including TLS or SSH tunnels and Tor. Our model is able to recover web object sizes as website fingerprint features, by injecting delay between object requests to isolate the download of data for each object. The scheme we develop following the active model obtains high identification accuracy. It drastically reduces the anonymity provided by Tor.

Through our study, we find that protecting user privacy involves tradeoff between communication anonymity and overheads, such as bandwidth overhead, delay, and sometimes even computation and storage. Currently, the most reliable countermeasures against traffic source identification are packet padding and adding dummy traffic. The aggressiveness of applying the countermeasures and the willingness to trade off the overheads impact the effectiveness of the anonymity protection.

List of Tables

List of Figures

List of Abbreviations

ACK	Acknowledgement
AES	Advanced Encryption Scheme
AMS	Advanced and Authenticated Marking Scheme
BCH code	Error correcting code by Bose, Ray-Chaudhuri and Hocquenghem
DDoS attack	Distributed-Denial-of-Service attack
DNS	Domain Name Service
DPF	Distributed Packet Filtering
DSSS	Direct Sequence Spread Spectrum
EER	Equal Error Rate
ESP	Encapsulating Security Payload
FIT	Fast Internet Traceback
FMS	Fragment Marking Scheme
FNR	False Negative Rate
FPR	False Positive Rate
Gossib attack	Groups of Strongly SImilar Birthdays attack
HMAC	Hash-based Message Authentication Code
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPsec	Internet Protocol Security
ISDN	Integrated Services Digital Network
ISP	Internet Service Provider
iTrace	ICMP traceback
JAP	Java Anonymous Proxy
MTU	Maximum Transmission Unit
PPM	Probabilistic Packet Marking
RnL	Randomize and Link
RPM	Random Packet Marking
SHA	Secure Hash Algorithm
SSH	Secure Shell
SSL	Secure Socket Layer
SYN	Synchronizing Segment
TCP	Transfer Control Protocol
TLS	Transport Layer Security
TTL	Time to Live
UDP	User Datagram Protocol

URL	Uniform Resource Locator
VoIP	Voice over Internet Protocol
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
XSS attack	Cross-Site Scripting

Chapter 1

Introduction

This introductory chapter gives the motivation of our thesis research, outlines the scope of problems we tackle and presents the main contributions we make.

Privacy and anonymity are some central issues in network security. Both legitimate and malicious traffic sources may apply obfuscation techniques in web browsing to bypass surveillance. Common traffic source obfuscation techniques include IP spoofing, encryption and proxy. Yet even if some obfuscation techniques are applied, remote traffic sources may still be identified using traffic analysis. As practical applications lead to deployed obfuscation techniques, and the loopholes of obfuscation techniques in turn lead to exploits by traffic source identification schemes, limitations following from the existing source identification schemes and countermeasures motivate us to study the approach to designing good source identification techniques and defences.

This introductory chapter is organized as follows. Firstly, motivations that lead us to research on the traffic source identification problem are discussed in Section 1.1. Next the models and assumptions used in our investigation are outlined in Section 1.2. We then summarize the main contributions of our research in Section 1.3. Finally, we outline the thesis organization in Section 1.4.

1.1 Motivation

We are motivated to research on the problem of identifying obfuscated web traffic sources, as it has important applications and the current techniques and countermeasures have some insufficiencies. We elaborate on the motivation by first giving example application scenarios where web traffic source identification is the central technical issue. Next we briefly review deployed techniques that are in support of anonymous web surfing. We point out not only the protections each technique provides, but also the loopholes they leave open. Then we discuss existing traffic analysis techniques that exploit loopholes of the obfuscation techniques to identify web traffic sources. We point out the insufficiencies of existing traffic source identification schemes, which motivates us to research on the web traffic source identification problem.

Application Scenarios

Web traffic source identification techniques can be utilized by legislative warden to ensure cyber security as well as be exploited by adversary to compromise web access anonymity.

The World Wide Web carries an abundance of information which is convenient to access. From reading local newspaper to checking stock quotes, surfing the web has become a vital tool. Web surfing is undoubtedly a prevalent Internet application. However, web browsing itself can be turned into a threat to user privacy. User identity or interests often are sensitive information, yet user interests can be profiled by studying the websites they surf. Such valuable information can be sold, or used to inject customized advertisements to get commissions from any resulting clicks. When users receive loads of uninvited advertisements targeting their specifics, their privacy is at obvious risk. Sensitive personal information, such as medical, financial, or family issues, are on the verge of being exposed. Therefore, there is a strong demand for privacy from users as with whom they are communicating or which websites they surf. Yet for business, there are strong financial incentives to identify the websites users surf so as to harvest web surfers' interests.

Under network attacks, web servers and clients alike want to identify the culprit (the attack traffic source). Distributed Denial of Service (DDoS) attack is an effective means to disable web services of business rivals. The attack is easy to launch with the support of voluminous bot networks, but difficult to prevent. Besides, there is strong financial interest to launching a DDoS attack. Business owners will be eager to hunt down the bots if their websites are attacked. While servers can be victims of DDoS attack, clients can be victims of Cross Site Scripting (XSS) attack. XSS attack has taken up a rising proportion in network attacks in recent years. Malicious codes are automatically downloaded when users visit some infected web severs in XSS attack. By identifying the websites that victims visit, it narrows down the suspects who spread the malicious codes.

The ability to identify web traffic sources is desired by parents, governments, law enforcement agencies, and many others. Parents want to monitor web accesses to protect youngsters from the influence of outrageous contents. Governments want to detect any breach of censorship to online political contents. Law enforcement agencies need to conduct electronic surveillance so as to combat crime, terrorism, or other malicious activities exploiting the Internet.

Overall, both defences against and techniques to identify web traffic sources are needed in practice, but in different contexts. They can positively benefit attack forensics and web access surveillance over obfuscated communication channel. Yet they can also compromise user privacy if misused.

Advantages and Drawbacks of Current Protection Tools

The currently deployed techniques which support data obfuscation in web surfing include different constructions over encryption and proxy, such as VPN or Tor.

In a simple communication environment without encryption or proxy, user identity can be revealed and associated with browsing a particular website by information in the transmitted data, e.g. user ID, phone number, or IP addresses. Note that it is even easier for ISPs to identity the websites that a user browse, since ISPs have the IP-to-user mappings and can easily log the IP addresses of websites that the user visits. However, proliferation of anonymous communication systems has posed significant challenges to the task.

Encryption and proxy are two main deployed tools that protect the privacy in web browsing. One possible construction is for users to access websites via a proxy, and encrypt the link between user and proxy. Proxy hides the direct connection between user and web server by rewriting the source and destination pairs. Encryption provides confidentiality of data to prevent identifying websites from contents. The encrypted links are possible at the link layer using WEP/WPA to a wireless base station, or at the network layer using IPsec ESP mode to a VPN concentrator, or at the transport layer using an SSH tunnel to an anonymizing proxy [10].

The single proxy construction does not protect the communication privacy from the proxy. Another construction employs multi-layered encryption with multiple proxies. Such construction is implemented in Tor, which is a Peer-to-Peer network that provides anonymous communication service. Furthermore, Tor unifies the packet sizes. Fixed packet size significantly increases the difficulty to distinguish websites by size related features. Encrypted communication through multiple proxies make the endpoints indistinguishable from the relaying proxies. No one proxy knows both the source and destination.

Encryption based protocols have given users a false impression of confidentiality of web surfing. An encrypted connection is not sufficient to remove traffic patterns that often reveal the website a user visits. For example, size of the base HTML file of a webpage already leaks much identifying information [16].

Although one or more intermediate proxies can hide the direct connection between a user and the web server, they have not broken the correlation in the volume or timing of the incoming and outgoing traffic. The propagation of a burst of traffic can iteratively reveal the communication path through multiple proxies. Even if information from the size channel is blocked, the timing channel still exposes much information. Because of the low latency requirement, anonymous communication systems that support web browsing all refrain from intentionally changing the packet delays. Packets sent and received in an interval are correlated at the sending and receiving ends.

Insufficiencies of Current Traffic Source Identification Techniques

Traffic source identification techniques exploit the loopholes of data obfuscation techniques to identify traffic sources. Traffic source identification techniques include packet marking, flow marking and fingerprinting. Each technique applies to certain traffic models.

The class of probabilistic packet marking (PPM) schemes [69, 74, 35, 93] applies to tracing bots manipulated to launch DDoS attacks using spoofed IP addresses. In PPM, routers embed partial path information into headers of probabilistically sampled packets they transmit. A victim server having received a collection of packet marks, reconstructs the DDoS attack paths from pieces of path information. The schemes have shown good performance in identifying one or more attack paths. However, the structures of packet marks as proposed in different schemes have only subtle differences. It remains unclear how to fairly compare which structure is better, given their minor differences in assumptions. Can we envision an optimal PPM scheme and improve the current designs towards the optimal? These questions are not yet answered.

Fingerprinting applies to identifying the websites user accesses through low latency encrypted tunnels. In website fingerprinting attack, adversary observes some traffic patterns of websites when they are fetched via an encrypted tunnel. From the side channel data of encrypted HTTP streams, the adversary builds a database of website fingerprints. Victim's web traffic is matched against the fingerprint database to infer the website identity. Previous works have demonstrated the feasibility of using size related features to fingerprint websites in the single proxy case [40, 78, 10, 50]. However, most website fingerprinting schemes rely on size related features alone, which makes them unable to withstand the countermeasure of traffic morphing [90]. We are interested to find out if there are additional relevant website fingerprint features, so as to defend against traffic morphing and to enhance the website fingerprint identification accuracy.

Flow marking has been demonstrated to be capable of associating a pair of communicating sources in traffic confirmation attacks against Tor [87, 94]. Flow marking techniques require control at both ends of the communication. One for watermark embedment, and the other for watermark verification. The question arises whether we can extend the website fingerprinting model from VPN to Tor (the de facto standard of anonymous communication system), so as to identify websites accessed over Tor by monitoring only the client end of the communication. Existing website fingerprinting attacks are not suitable for direct application to Tor. The reason lies in that the packet size related fingerprint features they rely on are not observable over Tor, because Tor transmits messages in fixed length cells. Experiments on one such scheme [39] show that it performs very well on identifying websites over SSH or TLS tunnels (up to 97% identification accuracy among 775 URLs), but it gives low accuracy when applied over Tor (3% accuracy among 775 URLs). There is not yet any reliable website fingerprinting models or techniques on Tor.

1.2 Purpose and Scope

Bearing in mind the unsolved problems discussed in the previous section, we further investigate the traffic source identification problem in this thesis. The purpose of this thesis is to study the models and mechanisms to identify obfuscated traffic sources in web browsing traffic, such as in attack circumstances where IP addresses are spoofed or in encrypted and proxied communications.

We develop a framework of traffic source identification that gives a taxonomy of its sub models. The domain of traffic source identifications is classified by attributes of traffic model or investigator capability. From the dependency among model components, we analyze criteria that guides the scheme design. The principles are substantiated in our scheme constructions in several problem scenarios.

Under the framework, we investigate three specific traffic models, (*i*) flooding DDoS attack with IP spoofing, (*ii*) encrypted and proxied communication, e.g. through SSH or SSL/TLS tunnel, and (*iii*) low latency mix network, or Tor. The traffic sources to identify in DDoS attacks are the attack paths or bots swamping a victim server, while the sources to identify in web browsing through SSH or SSL/TLS tunnels or Tor are the sensitive websites user accesses. We are not dealing with anonymized traffic logs to associate servers with their pseudonyms in this thesis.

We propose an analysis model for the class of probabilistic packet marking schemes for IP traceback, and we propose an active website fingerprinting model that works on any low latency, encrypted and proxied communication channel, including SSH, SSL/TLS tunnels and Tor network.

The source identification techniques we focus on are packet marking, passive and active traffic fingerprinting. Along the process of scheme development, we analyze the effectiveness of certain countermeasures, and propose our own countermeasures.

In Distributed Denial of Service (DDoS) attacks, many compromised hosts flood the victim with an overwhelming amount of traffic. The victim's resources are exhausted and services to users become unavailable. During a DDoS attack, attack nodes often perform address spoofing to hide their identities and locations. IP traceback aims to overcome address spoofing and uncover the attack paths or sources. Identifying the attack sources enables legislature to ascertain the responsible persons. It can also be performed prior to remedy actions, such as packet filtering, to isolate the attacker traffic. While traceback is motivated by DDoS attacks, it also benefits analysis of legitimate traffic. Potential applications of traceback include traffic accounting and network bottleneck identification.

Traceback schemes assume network routers are cooperative in embedding the required packet marks for inspection by victim servers. We do not explicitly handle "stepping stones" along the attack paths, but rely on autonomous systems to exchange and compile their results after analysis. We focus on designing a good quality packet marking scheme for cooperative routers for IP traceback.

Web browsing traffic through VPN (Virtual Private Network) are encrypted by SSL and proxied by the VPN server. VPN is a technology that allows users physically outside the private network to bring themselves virtually inside it, thus gaining access to all the resources that would be available if the users are physically inside the network. Users who browse websites with VPN can bypass censorship at their physical network. Website fingerprinting provides a means to track the website accessed, utilizing the side channel information leaked from the encrypted and proxied HTTP streams. We improve upon existing website fingerprinting scheme by re-examining the selection of fingerprint features. Our VPN website fingerprinting scheme also work on other SSH or SSL/TLS encrypted tunnels.

Extending from the passive website fingerprinting approaches over SSH or SSL/TLS tunnels, we tackle website fingerprinting over Tor. As communication relationship is sometimes sensitive information, Tor aims to protect anonymity of the conversing parties. Tor is designed based on mix network, where multiple Tor nodes act as proxies in transmitting fixed length packets with layered encryption. However, website fingerprinting is a threat to user privacy in web browsing, even over Tor. Tor conceals much size related features in traffic, which makes passive traffic analysis difficult. We design an active website fingerprinting model that retrieves certain feature values so as to fingerprint and identify the website from an HTTP stream anonymously transmitted by Tor. The active website fingerprinting model and scheme we design also apply to website fingerprinting over VPN, SSH or SSL/TLS tunnels.

Same as existing website fingerprinting models, our models assume that HTTP streams of simultaneous accesses to different websites, e.g. tagged browsing, are successfully separated for identification. We focus on developing systems that identify the website from each monitored HTTP stream.

1.3 Main Contributions

We build a framework that encompasses different web traffic source identification scenarios. The framework is useful for deriving source identification approaches suitable for the underlying traffic models.

We investigated source identification approaches in three traffic models under the framework, (*i*) DDoS attack traffic, i.e. flooding packets with spoofed IPs, (*ii*) VPN traffic, i.e. encrypted and proxied traffic, and (*iii*) Tor traffic, i.e. encrypted and proxied traffic with fixed packet sizes.

DDoS Traceback

We model Probabilistic Packet Marking (PPM) schemes for IP traceback as an identification problem of a large number of markers. Each potential marker is associated with a distribution on *tags*, which are short binary strings. To mark a packet, a marker follows its associated distribution in choosing the tag to write in the IP header. Since there are a large number of (for example, over 4,000) markers, what the victim receives are samples from a mixture of distributions. Essentially, traceback aims to identify individual distribution contributing to the mixture. The general model provides a platform for PPM schemes comparison and helps to identify the appropriate system parameters. We show that entropy is a good evaluation metric of packet marking quality such that it effectively predicts the traceback accuracy. We find that embedding hop count in tags reduces the entropy.

We propose Random Packet Marking (RPM), a simple but effective PPM scheme, guided by the general PPM model. RPM does not require sophisticated structure or relationship among the tags, and employs a hop-by-hop reconstruction similar to AMS [74]. Simulations show improved scalability and traceback accuracy over prior works. In a large network with over 100K nodes, 4,650 markers induce 63% of false positives in terms of edges identification using the AMS marking scheme; while RPM lowers it to 2%. The effectiveness of RPM demonstrates that with prior knowledge of neighboring nodes, a simple and properly designed marking scheme suffices in identifying large number of markers with high accuracy.

VPN Fingerprinting

We examine web traffic transmitted over an encrypted and proxied channel to discern the website accessed. Profiles of popular websites are gathered, which contain website fingerprints developed from side channel features of the encrypted and proxied HTTP streams, then we identify the website accessed in a test trace by comparing the trace fingerprint against the library of website profiles to find a good match. Under a passive traffic analysis model, we develop a scheme to fingerprint websites that utilizes two traffic features, namely the packet sizes and ordering. Packet ordering was not thoroughly exploited in website fingerprinting previously.

Our scheme yields an improved identification accuracy over prior work (Liberatore and Levine's scheme [50] is used as our *reference scheme*) in both classification and detection scenarios. The detection scenario is more difficult since it is unknown if a test stream comes from the profiled websites. In the classification scenario, our scheme achieves an accuracy of 97% on analyzing 30-second long OpenVPN test streams from 1,000 websites. On identifying the 6-second long OpenSSH test streams of 2,000 websites, our accuracy reaches 81% with 11% improvement from the reference scheme. In the detection scenario, the equal error rate of our scheme is 7%, while that of the reference scheme is 20%; and the minimum total error rate (i.e. sum of false positive rate and false negative rate) is at 14% for our scheme versus 36% for the reference scheme.

Traffic morphing [90] defends against website fingerprinting by changing the packet size distribution to mimic the traffic from a target website, while minimizing the bandwidth overhead. Our scheme withstands traffic morphing by using packet ordering to differentiate websites that have similar packet size distributions. Our scheme distinguishes 99% of the morphed traffic from the mimicked target, while the reference scheme distinguishes only 25%. We note that there is tradeoff between security and bandwidth efficiency in morphing. If morphing considers some ordering information to strengthen anonymity, then its bandwidth efficiency is severely degraded to be worse than a simple mice-elephant packet padding.

We empirically analyze the fingerprint consistency, under different pipeline settings, with static and dynamic websites, and over time. Evaluation shows that our website fingerprints are robust to variations in HTTP pipelining configurations, and that they are stable over time with only 6% needs reprofiling after a month.

Active Tor Fingerprinting

Tor is the de facto standard of low latency anonymous network. It protects anonymous communication with layered encryption, onion routing and fixed length cells in data transmission. We propose an active traffic analysis model to perform website fingerprinting over Tor. In our active approach, the adversary, who acts as a man in the middle between the user and the Tor entry node, holds any HTTP requests till the ongoing response is fully transmitted. It undoes interleaved object transmissions to reveal individual web object sizes. Object sizes and ordering are used as our website fingerprint features. While previous work exists on traffic confirmation attacks, ours is the first to consider website fingerprinting over Tor, which does not require controlling both ends of an anonymous communication. Our scheme achieves an identification accuracy of over 67.5% of 200 websites. In contrast, random guess is expected to correctly identify a website with probability 0.5%. We show that our active model is feasible to fingerprint websites accessed through Tor.

To summarize, the main contributions of this thesis are:

- Proposed a general model of Probabilistic Packet marking (PPM) schemes for IP traceback. Proposed using entropy in the collection of packet marks as an evaluation metric to predict the traceback accuracy with an optimal path reconstruction algorithm.

- Proposed a traceback scheme that increases the randomness of packet marks and hence improves the traceback accuracy.

- Proposed a passive website fingerprinting scheme over VPN that introduced packet ordering into fingerprint features, in addition to packet sizes.

- Defeated the traffic morphing technique, by exploiting its limitation on lack of packet ordering consideration or its constraint on bandwidth efficiency.

- Proposed an active website fingerprinting model applicable to any low latency encrypted and proxied communication channel, and developed a scheme following the model that significantly reduces the anonymity provided by Tor.

1.4 Thesis Organization

Body of this thesis is organized as follows:

1. Chapter 1 is this introductory chapter.

2. In Chapter 2, we present the background on HTTP stream patterns over VPN, and Tor traffic characteristics. We also review the current developments of web traffic source identifications. They serve for the comparative analysis in our research.

3. We lay out a framework for traffic source identification in Chapter 3. It classifies the domain of traffic source identification by the attributes of traffic model or investigator capability. The criteria for designing a source identification scheme under several traffic models and analyst's capabilities are laid out. The principles are substantiated in our schemes proposed for the investigation of traffic sources in several specific problem scenarios, namely, DDoS traceback, passive website fingerprinting over VPN and active website fingerprinting over Tor, which are presented in subsequent chapters.

4. We provide a general model of probabilistic packet marking (PPM) schemes for IP traceback in Chapter 4. We propose in the model an evaluation metric to fairly

evaluate packet marking qualities. The model analyzes under flooding DDoS attacks, the common structures in packet marks adopted by existing schemes, and compares their assumptions and approaches to path reconstruction.

5. Inspired by the findings from the previous chapter, we design a PPM scheme named *Random Packet Marking* in Chapter 5. The scheme improves the quality of packet marks and hence increases the DDoS attacker identification accuracy over previous schemes.

6. We develop a passive website fingerprinting scheme that utilizes packet ordering information in conjunction to packet sizes in Chapter 6. The scheme applies over VPN, SSH or SSL/TLS encrypted tunnels to identify the website accessed in an HTTP stream. We can use side channel features of packet ordering and sizes to fingerprint websites because encryption and proxy have not severely change them.

7. We analyze the effectiveness of our passive website fingerprinting scheme against traffic morphing in Chapter 7. Traffic morphing is designed to defend against website fingerprinting with a bandwidth efficiency constraint, which makes it unable to handle website fingerprinting schemes that account for packet ordering. We suggest a countermeasure to website fingerprinting that exploits randomization in packet sizes and HTTP request ordering, with the aim to aggressively remove traffic features in both sizes and timing channels.

8. Built on the passive website fingerprinting techniques over VPN, we propose an active website fingerprinting model against Tor and demonstrate that it is feasible to identify websites from Tor protected traffic in Chapter 8. The fixed packet size in this traffic model significantly increases the difficulty in website fingerprinting. We therefore use an active approach to obtain the sizes and ordering related features to be website fingerprints.

9. We conclude the thesis and point out directions for future work in Chapter 9. We summarize the findings from constructing and analyzing traffic source identification schemes under different traffic models. We also point out practical constraints that we have yet to address, and the direction to enriching the models and improving the scheme designs.

Supporting materials are organized as appendices:

- We briefly present in Appendix A the primitives of similarity measurement which are applicable to website fingerprint comparisons. Pseudocodes of the Wagner-Fischer algorithm for the computation of Levenshitein's edit distance are presented for reference.

- Pseudocodes of the extended edit distance we design are given in Appendix B.

Chapter 2

Background

We present the background on VPN and Tor traffic characteristics, and we review the current development of traffic source identification techniques, for further comparative analysis.

2.1 Overview

A traffic source is identified by a unique IP address to facilitate network communication. Yet IP address can be spoofed when response is not expected, such as in attack scenarios. In particular, bots launching a Distributed Denial of Service (DDoS) attack usually spoof their IP addresses to evade detection. A substantial amount of IP traceback schemes have been proposed to trace the attack paths so as to uncover the bots.

IP addresses can also be obfuscated for confidentiality of communication. Techniques deployed in practice that may hide the communication endpoints include encryption with proxy (SSH tunnel or SSL/TLS tunnel), and Tor.

We use side channel information of web traffic which is revealed despite the use of encrypted tunnel to identify the website user accesses. Web traffic characteristics that are observed over encrypted tunnels enable the selection of effective side channel features. The feature values of each website are treated as its fingerprint for identification. Feasibility of website fingerprinting has been demonstrated in a number of schemes.

A related problem to website fingerprinting is on recovering web server identities from anonymized traffic log. However, the problem we address is different from it in two aspects. (*i*) It identifies websites from a collection of access logs where flow statistics are preserved, while we identify based on each web access. (*ii*) It analyzes traffic log of anonymized but not encrypted data, in which most features of individual connections are observable, including start time and end time, amount of data transmitted. Similar data are not available from encrypted and proxied communication. We are not studying this problem in further details in this thesis, but existing techniques to anonymize or deanonymize traffic logs are surveyed.

Tor is the most trusted and most widely used in servicing anonymous communication. It is a realization of mix network, but it removes the operation that batches packets, in order to have low latency in supporting interactive applications, e.g. web browsing and VoIP. Yet Tor operates under a controversial security model, which assumes limited capability of adversaries who can only make non-global observations and control a fraction of Tor nodes. Even under Tor's somewhat restricted threat model, various passive and active attacks have been proposed to correlate traffic across Tor nodes. Most notably are traffic confirmation attacks where a unique watermark is embedded into the timing channel to confirm a suspected communication relationship.

This chapter is organized as follows. Web traffic characteristics observed over VPN is discussed in Section 2.2. Tor architecture and its security model are described in Section 2.3. These two sections provide backgrounds on traffic patterns for analysis, then we present in the subsequent sections existing traffic source identification schemes under different traffic models. Techniques to trace DDoS attack paths are categorized in Section 2.4. Website fingerprinting and flow watermarking schemes are compared in Section 2.5. Attacks that break the anonymity of Tor are classified in Section 2.6. Techniques to anonymize or deanonymize traffic logs are surveyed in Section 2.7. Finally Section 2.8 summarizes this chapter.

2.2 Web Traffic Behavior over VPN

Webpage contents are retrieved from the web server using HTTP and presented by the client browser. SSL or its successor TLS provides a layer of protection to data secrecy by encrypting the HTTP traffic before transmitting them over TCP/IP. VPN provides an SSL tunnel such that the server address is encapsulated into the packet payload and encrypted.

We find two important characteristics of web traffic over VPN by observing tunneled and encrypted HTTP streams. By an *HTTP stream*, we mean the stream of packets sent and received for a web access. The observations we make are: (*i*) webpage download by HTTP is highly structured; and (*ii*) encryption and proxy do not severely alter the packet sizes, nor the packet ordering. Although our discussions are based on VPN, the traffic patterns generally apply to other low latency SSH or SSL/TLS encrypted tunnels. The observations help us to infer the consistent website features for fingerprint. The protocol properties related to our feature selection are discussed in this section.

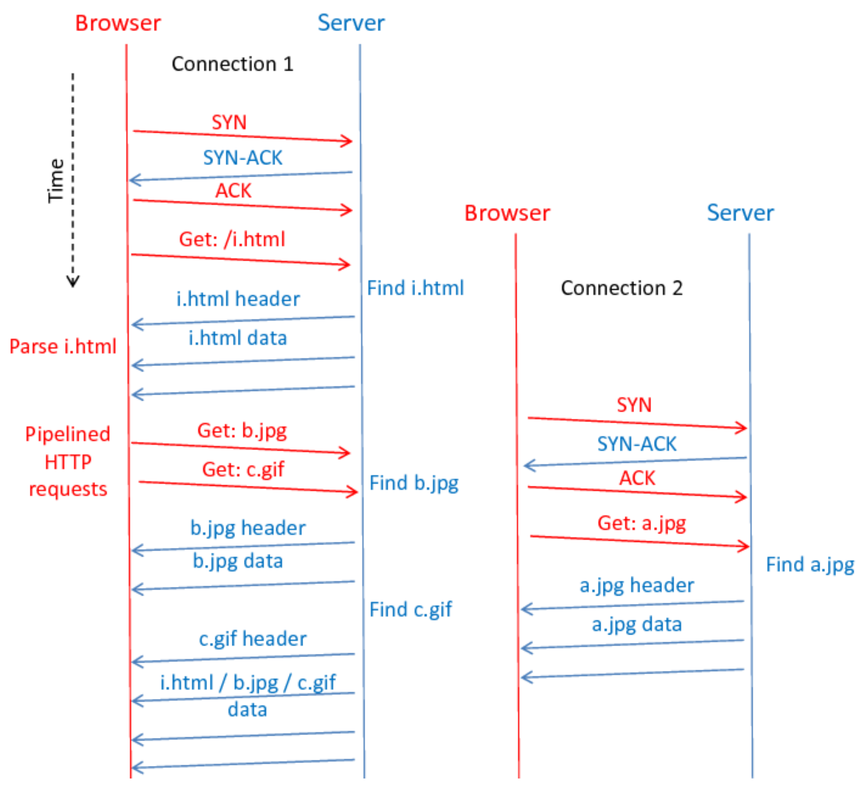


Figure 2.1: Communication between browser and server. ACKs to data packets are omitted for clarity.

HTTP Stream

Here we argue that the layout of webpages changes only infrequently, and because of the regular behavior of HTTP and browser, the object requests and responses of a webpage have consistent order and sizes.

Websites with dynamic contents are usually generated based on design templates. Their layout does not often change, because redesigning the template requires manual work, which is time consuming and costly. Although the referenced embedded objects can change, their sizes tend to vary within a range. Websites that do not rely on templates only update infrequently.

HTTP has regular behavior in webpage loading, as illustrated in Figure 2.1. Client browser downloads the *base object* of a webpage, which is usually an HTML file, and parses it. The parsing can be done while the base object is still downloading. When it encounters a reference to an *embedded object*, such as graphics or stylesheets, it issues an HTTP request to fetch it. Since web servers are required to serve HTTP requests in a first-come-first-serve manner [28], the order of the response data is correlated to the order of the requests.

HTTP chunked encoding is widely used. It modifies an HTTP message body to transfer it in a series of chunks, so as to allow clients to display the data immediately after receiving the first chunk. It is especially useful when the message is dynamically generated with several components, the server need not wait to calculate the total Content-Length before it starts transmitting the data. For example, a large graphics file is compressed and transferred by pixel blocks. If chunked encoding is not supported, then all packets of an object response have sizes equal to the path MTU, except for the last one which sends the remaining data. Otherwise, an HTTP response can contain several packets of sizes not equal to the path MTU, which are the last packets of data chunks. These packet sizes are characteristic and their order is in sequence of the object data.

To speed up the presentation of a webpage, (*i*) multiple TCP connections are opened to download several objects in parallel, (*ii*) on each connection, HTTP requests can be *pipelined*, meaning a client is allowed to make multiple requests without waiting for each response. Because of multiple TCP connections and HTTP pipelining, data packets belonging to different embedded objects can interleave, thus object sizes cannot be determined by accumulating the amount of data received between adjacent HTTP requests. Multiple TCP connections and HTTP pipelining are the main sources of variation in the otherwise consistent order of object requests and responses. In addition, the order of objects may be browser specific, and the dynamics in network condition causes some random noise.

There are a few causes to the slight variation of an object size in transmission. An HTTP response message contains zero to more optional headers, while the default is associated with the object type and the browser. Certain types of objects are always compressed during transmission for efficiency, e.g. multimedia contents. The default compression algorithm is browser specific, though it can be negotiated between the server and browser. Hence the length of the optional headers and the size of a compressed object are consistent for each object and browser, with slight variation across browsers.

SSL 3.0 and TLS 1.0

SSL [31] or TLS [6, 41] protects data secrecy of HTTP messages by encryption and data integrity by hashing. Long HTTP messages are fragmented to be sent in multiple SSL/TLS records. The record length approximately preserves the HTTP message fragment length, but appended with a message authentication code and padded for encryption. Depending on the HMAC-cipher suite negotiated, the SSL/TLS record length is increased by 16 to 28 bytes from the message fragment. Part of a record and/or several short records can be packed into one packet, bounded by the path MTU, but each record is required to be flushed. The length of each SSL/TLS record can be observed from the record header. When the packet payload is not available, the packet length can be taken as an estimate of the record length, though it may represent the total length of several SSL records and is less accurate.

In SSL 3.0, the padding added prior to encryption is the minimum amount required to make the message length a multiple of the cipher's block size. In contrast, TLS 1.0 defines random padding in which the padding can be any amount up to 255 bytes that results in a multiple of the cipher's block size. However, it is reported that random padding is not implemented in browsers [88]. Our research reiterates that the random padding to packets should be enforced to thwart traffic analysis based on message sizes.

OpenVPN

The data authenticity, confidentiality and integrity provided by OpenVPN [60] is based on SSL/TLS. It encrypts both the data and control channels using the OpenSSL library. By comparing the packets in plaintext and ciphertext, we find that the increase in packet sizes is consistent at about 100 bytes.

The OpenVPN server also acts as a proxy for web clients, deploying the IPSec ESP protocol for packet tunneling. All the TCP/UDP communications between a client and the VPN server are multiplexed over a single port. This prevents revealing the number of TCP connections opened or the number of objects in a webpage.

2.3 Tor's Architecture and Threat Model

Tor [81] is the second generation Onion Router that was initially sponsored by the US Naval Research Laboratory. The principle of Tor is mix network [14], but Tor sacrifices some security for reducing the latency and bandwidth overhead.

Several email anonymization systems were built based on mixes, notably *Babel* [37], *Mixmaster* [61, 54] and its successor *Mixminion* [23]. The latency of these systems is tolerable for email, but not suitable for interactive applications. Some mix based systems were developed to carry low latency traffic, notably ISDN mix [64] anonymizes phone conversations, and Java Anonymous Proxy (JAP) [9] anonymizes web traffic. ISDN mix was designed for circuit switched network where all participants transmit at a continuous and equal data rate, whereas Tor supports the more dynamic packet switched Internet in anonymizing TCP streams. Traffic flowing through JAP goes through several nodes arranged in a fixed cascade, while Tor allows path selection and is more versatile than JAP by supporting hidden services. There are a list of commercially deployed anonymizing networks, including `findnot.com` and `anonymizer.com`. Yet Anonymizer is a single hop proxy.

The mix based design makes Tor trusted and popular as a real anonymous communication system. Currently Tor has over 1600 nodes acting as routers and much more end users.

Mixes

The concept of using "Mixes" to provide anonymous communication was introduced by Chaum in 1981 [14]. Each mix node acts as a relay that hides the correspondence between its input and output messages. Mix system provides most promising balance between anonymity and efficiency in terms of bandwidth, latency and computation overheads.

A mix has three essential functions, namely, providing bitwise unlinkability, batching messages, and generating dummy traffic. Unlinkabililty breaks the association in bit patterns of incoming and outgoing messages of a mix. To achieve unlinkability, messages are divided into blocks and incrementally encrypted with the chain of mixes' public keys. Upon receiving a message, a mix decrypts the message, strips off the next relay's address contained in the plaintext block, then appends a random block to keep the message size invariant. A mix buffers a number of messages to process in a batch. The messages are reordered before being sent out. Batching makes tracing based on time or the ordering of messages difficult. Dummy traffic covers the genuine messages amidst noise. Dummy messages can be created by message senders or mixes alike.

Tor Architecture

An anonymous communication channel via Tor flows through a *circuit*, which is a path composed of (by default) three randomly selected Tor routers that connects from the client proxy to the desired destination. The first node is called the *entry node* or *entry guard*, and the last node is called the *exit node*. Entry guard is chosen preferentially among high uptime and high bandwidth Tor routers.

Tor routers are responsible for obscuring from observers the correspondence between their incoming and outgoing data streams. Data of TCP streams are divided into *cells* each of 512 bytes and wrapped in layered encryptions to maintain unlinkability. Each hop on the circuit removes a layer of encryption till a cell is fully decrypted at the exit node, where cells are reassembled into TCP packets and forwarded to the user's intended destination. This process is known as *onion routing* [34].

Tor does not perform any batching of messages or generating any dummy traffic, because of concerns on latency and bandwidth efficiency. When TCP data are packaged into cells, the data are padded if there are less than 512 bytes. This reduces the latency in data buffering and facilitates interactive procotols, such as SSH that sends short keystroke messages. Unlike email mixes, Tor does not intentionally introduce any delay. Its typical latencies are in the range of 10-100 ms [55].

Every Tor circuit has a maximum lifetime, to prevent anonymity being compromised due to using the same circuit for a long time. The maximum lifetime of a circuit is configurable and it is 10 mins by default. After maximum lifetime expires, idle circuits are torn down and replacements are set up.

Threat Model

Security researches often assume a powerful adversary to guarantee systems that protect against it are secure in real world conditions. A powerful adversary to anonymous communication would arguably be able to monitor all network links, to inject, delete or modify messages along any links, and to compromise a subset of network nodes. Tor assumes a weaker adversary.

Tor has a security model commonly used by low latency anonymous systems, e.g. Freenet [18], MorphMix [66] and Tarzan [30]. It protects against a non-global adversary who can observe and control only a portion of the network, and can inject, drop or alter traffic only along certain links. Given any mix could be compromised, each Tor circuit contains a chain of mixes so that if at least one mix in the chain is not malicious, some anonymity is provided. Tor entry node knows the user IP, while the exit node knows the destination, but no one single Tor node knows both ends of a communication.

Tor does not protect against a global passive adversary. For quality of service concerns, Tor has not explicitly altered the delay of packets nor hidden the traffic volume by adding dummy traffic. Hence, it cannot break the correlation of traffic by timing or volume. The class of statistical disclosure attacks correlate the timing of inbound and outbound traffic among all nodes the adversary monitors to determine long term communication patterns. Tor is also vulnerable to traffic confirmation attacks, where an adversary monitors the traffic at two suspected parties to validate if they are communicating with each other.

We propose an active website fingerprinting attack under Tor's security model. Our approach does not require a global adversary, nor controlling links at both ends of a communication. It only assumes the Tor entry node (or a router between the user and the Tor entry node) is compromised so as to observe and delay certain packets.

2.4 DDoS Packet Marking Schemes

Existing DDoS traceback schemes can be classified into two categories. (*i*) Routers are queried on the traffic they have forwarded. The routers may not need to log packets. (*ii*) The receiver locally reconstructs the attack paths from a collection of packets. Each packet carries partial path information. The packets are either probabilistically marked by routers or specially generated for traceback. The first category includes online query and variations of hash based logging schemes [73, 49]. The second category includes variants of probabilistic packet marking (PPM) [69], ICMP traceback (iTrace) [8], and algebraic encoding [24].

Probabilistic Packet Marking Schemes

In iTrace [8], routers sample packets with a small probability. A sampled packet is duplicated in an ICMP packet, plus information of the router's upstream or downstream neighbor forming an edge with itself. Based on the ICMP packets, the victim reconstructs the attack paths by linking up the edges. Note that routers farther away generates fewer iTrace packets to the victim. A variant of iTrace, called intention-driven iTrace [51], introduces an intension indicator to inform remote routers to raise their probability in generating iTrace packets.

Instead of adding network traffic, PPM (probabilistic packet marking) probabilistically embeds partial path information into packets. Savage *et al.* [69] proposed the Fragment Marking Scheme (FMS). Two adjacent routers, forming an edge, randomly insert their information into the packet ID field. The path information thus spreads over multiple packets for reassembly. However, for multiple attack paths, the computation overhead of path reconstruction is high, due to explosive combinations of edge connections. Subsequent proposals: Advanced and Authenticated Marking Schemes (AMS) [74], Randomize-and-Link (RnL) [35], and Fast Internet Traceback (FIT) [93] improve the scalability and the accuracy of traceback. Dean *et al.* [24] adopted an algebraic approach for traceback, by encoding path information as points on polynomials. The algebraic technique requires few marked packets per path for reconstruction. However, the processing delay on the marked packets can be large if a long sequence of routers performs marking. On the other hand, if short sequences of routers perform marking, the reconstruction overhead will be large due to combinatorial search. The scheme does not scale for multiple attackers.

2.5 Website Fingerprinting and Flow Watermarking Schemes

Packet marks are embedded into unencrypted packet headers for inspection. Whereas when packet headers are encapsulated into the packet payload and encryption is applied, we turn to the side channels of traffic flows to extract identifying information about traffic sources.

Side channel information has profitted a wide range of traffic analysis applications. For example, keystroke intervals are used for password guessing over SSH [75]; the sequence of bit rates is used to identify the streaming video encoded with variable bit rate schemes [68]; packet sizes and packet rates are used to reveal the presence of Skype traffic [12]; packet size, timing and direction are used to infer the application layer protocol [91].

In chapters 6 and 8, we use side channel information to identify websites accessed through low latency encrypted tunnels. Both website fingerprinting and flow watermarking techniques work on similar traffic models.

Existing website fingerprinting schemes are passive in the sense that they merely observe HTTP stream patterns. Most of them exploit packet size related features to generate fingerprint of a website, which is compared with website fingerprint profiles for identification. It is difficult to apply packet size based fingerprinting schemes on Tor, as Tor transmits data in multiples of a cell. We propose an active website fingerprinting model in Chapter 8 with evaluation on Tor.

Flow marking techniques do not require traffic themselves having rich identifying information. Instead, they embed a transparent watermark on the timing channel of a flow for traffic source verification. In contrast to existing website fingerprinting schemes, flow watermarking are active and applicable on Tor. Flow watermarking techniques require some control at both ends of a communication, while website fingerprinting only need some control at the user end to infer the visited remote website.

We review in this section the existing website fingerprinting schemes and countermeasures, as well as the flow watermarking schemes.

Website Fingerprinting Schemes

Several works [40, 78, 10, 50, 39] looked at the issue of using traffic analysis to identify websites accessed through certain encrypted tunnels, such as SSH tunnel or VPN. However, some of their assumptions have been invalidated with the changes in HTTP and web browser behaviors. For example, browsers were assumed not to reuse a TCP connection to download multiple embedded objects [40], and object requests were assumed non-pipelined [78]. Hence their approaches to determine an object size by accumulating the data received either through a TCP connection [40] or between adjacent HTTP requests [78] no longer work.

While Hintz [40] and Sun *et al.* [78] used features on objects sizes for website fingerprints, Bissias *et al.* [10], Liberatore *et al.* [50] and Herrmann *et al.* [39] used features on packets. Bissias *et al.* [10] used the features on distributions of packet sizes and inter-arrival times. It identified a website by finding the cross correlation between the test fingerprint and the fingerprint profiles. It obtained an accuracy of 23%, probably due to packet inter-arrival times vary with network condition and server load. Liberatore *et al.* [50] used the set composed of (direction, packet size) pairs as fingerprint, and applied Jaccards classifier and Naive Bayes classifier with kernel density estimation. Between which, Jaccards classifier gave better performance. It measures similarity as $\frac{|X \cap Y|}{|X \cup Y|}$, where X and Y are the sets representing a website profile and a test fingerprint, and $|X|$ denotes the size of set X. The scheme achieved an identification accuracy of 70% when examining

2,000 websites. Herrmann *et al.* [39] operated on the packet size frequency vectors, using Multinomial Naive Bayes (MNB) classifier. The transformation to relative frequencies yielded improved performance over Liberatore *et al.* [50]'s scheme. However, relying on relative frequencies made it difficult to incrementally add website profiles to the database, because the modification affects the overall frequency distribution. The identification accuracy is thus sensitive to the timeliness of website profiles. Fingerprinting schemes based on packet size related features [10, 50, 39] are not effective against Tor, as Tor transmits messages in fixed length cells. An attempt on Tor gave an accuracy of below 3% on identifying 775 URLs [39]. These schemes only utilized side channel features related to sizes but not ordering.

Coull *et al.* looked at a related problem which identifies websites from anonymized flow logs [20]. The scheme identifies individual servers that supply embedded objects of a webpage by applying kernel density estimation on the per flow size and the cumulative size of flows. From the sequence in which servers are contacted, it identifies the visited webpage. However, the technique does not transport easily to identifying proxied and encrypted communication, because servers are not consistently mapped to their pseudonyms and connection statistics are not preserved with proxy and encryption.

Countermeasures to Website Fingerprinting

Several variants of packet padding are proposed to defend against website fingerprinting. Padding every packet to the size of path Maximum Transmission Unit (MTU) can thwart size related traffic analysis, but the amount of overhead it causes is nearly 150% of the actual data [50]. "Mice-elephant" packet padding incurs relatively less overhead, at nearly 50% growth in the data transmitted. It pads packets to two sizes, either a small size for control packets, or to the path MTU for data packets. The large bandwidth overhead they cost leads to insufficient incentives for deployment.

Wright *et al.* [90] proposed traffic morphing as a bandwidth efficient alternative to packet padding. It transforms the source packet size distribution to mimic that of a target website, by splitting or padding the packets, while minimizing the bandwidth overhead. The morphing technique targets at fingerprinting schemes that only use information on packet size distribution. It considers limited or no packet ordering information.

The approximate total size of a webpage is not concealed even if packets are padded or the size distributions are changed. Dummy traffic can be used to cover up this feature if they are augmented indistinguishably from normal traffic. Clearly, packet padding and dummy traffic trade off bandwidth efficiency for anonymity.

Flow Watermarking Schemes

Communications with encryption and anonymizing network are perceived by many as both secure and anonymous. However, works by Wang *et al.* [87] and Yu *et al.* [94] demonstrated that low latency anonymizing networks are susceptible to timing attacks, and that watermarking techniques can be applied to UDP or TCP traffic alike, to track anonymous communications. Yet watermarks on multiple flows may interfere with one another if they are transmitted over common links.

Wang *et al.* presented a watermarking scheme to confirm the communicating parties of Skype VoIP calls [87]. Skype [80] encrypts data streams from end to end using 256-bit AES, and it uses VPN provided by `findnot.com` [29] as its anonymizing technology. The underlying peer-to-peer network of Skype is KaZaa [43], which transmits UDP messages. The watermarking scheme embeds a distinctive bit sequence into an encrypted VoIP flow by adjusting the interval of packets. The affected packets only need to be delayed by several milliseconds for an embedded watermark to be preserved across the anonymizing network, if sufficient redundancy is applied. The watermarking scheme requires direct control of a VoIP gateway located close to the caller.

Yu *et al.* proposed a flow watermarking technique based on Direct Sequence Spread Spectrum (DSSS) and evaluated it on Tor [94]. The technique embeds a watermark of pseudo-noise code by interfering with sender's traffic. Using interference eliminates the need to capture a flow for changing packet intervals, although the interferer needs to share the physical link with the traffic source. The delay between performing a traffic interference and when it is effected cannot be reliably predicted, due to the dynamic traffic rates of other flows on the link.

2.6 Attacks on Tor Anonymity

The main goal of attacks to Tor is to reveal the anonymous communication relationship, or to discover hidden services protected by Tor. Based on the features they exploit, attacks to Tor can be classified into attacks on circuit setup, attacks using timing information, and attacks based on traffic load.

Attack Targeting at Tor Circuit Setup

Bauer *et al.* proposed an attack that exploits Tor's preferential path selection strategy which favors nodes with high bandwidth capabilities [7]. As the resource claims by nodes are not verified, even a low resource adversary can make false claims and compromise a high percentage of circuit building requests of Tor. Srivatsa *et al.* proposed an attack that uses triangulation based timing analysis to infer the sender if the transmission route is set up by shortest path [76]. The attack potentially affects Tor because Tor has low latency, but Tor circuit is not set up using shortest path.

Timing Analysis

Many attacks [9, 87, 94, 95, 42, 44, 22, 52, 4] target at Tor's low latency property. They exploit timing related information to correlate flows or to confirm suspected sender and receiver relationship.

Passive flow correlation attacks include works by Zhu *et al.* [95], Hopper *et al.* [42] and the class of statistical disclosure attacks [9, 44, 22, 52]. The attack by Zhu *et al.* correlates the output link to an input link of a mix given the mix batching strategy [95]. Long term intersection attack [9] correlates times when senders and receivers are active. Disclosure attack [44] and statistical disclosure attack [22, 52] monitor messages sent by a user and messages received by a set of candidates in a series of intervals, so as to (statistically) establish the likely communication preference. Hopper *et al.* provided a quantitative analysis on the information leaked from network latency and network coordinate data [42]. It also mounted an attack for colluding websites to associate flows to the same initiator based on the web servers' local timing information.

Active traffic confirmation attacks [87, 94] perturb packet intervals to embed a unique "watermark" at sender for detection at the receiver, so as to confirm a suspected communication relationship. Difference between the attack schemes lies in their techniques to embed a watermark. A watermark can be embedded by a compromised Tor entry node [87], or through interference with the sender traffic applying direct sequence spread spectrum technique [94]. The attack by Abbott *et al.* [4] tricks a client web browser (e.g. by injecting JavaScript) into sending a distinctive signal which is logged with the help of the malicious Tor exit node. When the entry node becomes some malicious node due to circuit replacement, the signal is linked to the user and hence compromising the anonymity in web browsing. Yet interval based watermarking schemes, e.g. [94], are susceptible to multi-flow attack [46]. The attack combines multiple watermarked flows to detect the watermark presence, recover the secret parameters and remove the watermark.

Since the disclosure attacks and traffic confirmation attacks require simultaneous control over the entry and exit link of the Tor circuit, some proposal suggests diversifying the geographic location of Tor nodes selected for a connection [26]. Yet this does not prevent attacks that decouple the compromise of entry and exit nodes, e.g. [4]. Adaptive padding [86] is proposed to defend against both passive and active Tor traffic analysis which exploits timing. In adaptive padding, intermediate mixes inject dummy packets into statistically unlikely gaps in the packet flow, to destroy the timing "fingerprints".

Attacks Exploiting Traffic Load

Some attacks [70, 32, 56] use traffic load to correlate incoming and outgoing flows of a Tor node, so as to reveal the Tor path and to associate flows from the same initiator. Passive analysis [70] shows a connection can be traced through correlating the incoming and outgoing packet counts of a mix in an interval, and the propagation of traffic increase, which indicates a connection start, is observable by the packet counter. Active attacks [32, 56] are based on the observation that traffic volume of one flow affects the latency of other flows of a mix. Attacker injects packets and monitors the traffic latency to infer either the user traffic rate or the Tor path of a flow. A related attack [55] effects by causing server load variations, and remotely observes the clock skew through timestamps, so as to discover hidden services.

The attack we propose is substantially different from the existing Tor attacks. Firstly, existing attacks on Tor such as watermarking, active probing and disclosure attacks all focus on the subtle differences in packet timing or traffic load when they infer traffic route or communicating parties. We are the first in taking an active approach in utilizing inline object size features as "fingerprints" to identify websites accessed in Tor. Secondly, most end-to-end identification over Tor require some control over both the first link and the last link in the communication path. Whereas our model only requires controlling traffic on the entry link, the probability in fulfilling the attack condition is increased. Thirdly, Tor is believed to prevent adversaries from uncovering a communication relationship if adversaries have no ready suspects of the communicating pair. Yet our study will show that even for an adversary with poor apriori suspicion among a large set of candidate websites, anonymous web browsing can be identified with high accuracy.

2.7 Traffic Log Anonymization and Deanonymization

Traffic log anonymization has the objective of sanitizing sensitive data (e.g. hiding IP addresses) to protect user and server anonymity in releasing traffic log for research. Many attacks target at, and improvements are proposed upon (partial) prefix-preserving anonymization [92, 62].

Anonymization Techniques

`Tcpdpriv` [53] is the most well known anonymization tool. It operates on `tcpdump` traces, and provides different levels of restrictiveness in removing sensitive data, which includes prefix-preserving anonymization. Many tools wrap around tcpdpriv with slight extension, e.g. `ip2anonip` [65] and `ipsumdump` [47].

Xu *et al.* proposed a prefix-preserving IP address anonymization technique [92]. Prefix preserving requires if two unanonymized IP addresses share a k-bit prefix, so will their anonymized counterparts. The technique by Xu *et al.* applies a stateless cryptography algorithm, such that subnet structures are preserved and IP addresses are mapped consistently across anonymization sessions. The technique is implemented in a software named Crypto-PAn [83], and has been applied to anonymize Netflow data [71].

Pang *et al.* proposed a partial prefix-preserving anonymization system [62]. It guarantees that two IPs in the same unanonymized subnet will also be in the same anonymized subnet, but other prefix relationships among IPs are not preserved. It uses a pseudo-random permutation to anonymize subnet and host portions of IPs separately.

Koukis *et al.* designed a general framework that allows customized anonymization policy, including supports of application layer anonymization, such as to randomize the URL field of an HTTP request [48]. The framework is backed up by programming interfaces.

Deanonymization Attacks

King *et al.* presented a taxonomy of attacks to anonymized network log [45]. The taxonomy formalizes the attack pre-conditions into adversary knowledge set and adversarial capabilities, such that using first order predicate, they can express constraints of attack construction. The resulting high level attack classification is identical to that given by Slagell *et al* [72], which contains five attack classes: fingerprinting, structure recognition, known mapping attack, data injection and cryptographic attack.

Coull *et al.* proposed an attack feasible over NetFlow logs that deanonymizes servers through behavioral profiling [21]. It first finds heavy hitters using normalized entropy. If a few IP addresses occur much more frequently than others, the normalized entropy of the addresses will be low. Then it develops behavioral profiles of the heavy-hitters, such as what services they offer. Finally pseudonyms of servers can be deanonymized by comparing the behavioral profiles and public information on server popularity. Coull *et al.* [19] quantified the anonymity of a host by the entropy of probability distribution on the object's possible identities. The work also analyzed conditional anonymity which showed that the anonymity upon deanonymization of other hosts were significantly reduced.

Ribeiro *et al.* presented an efficient host deanonymization attack targeting (partial) prefix-preserving anonymized traces [67]. It found that network structural constraints led to IP disclosures. Ambiguous host identities were further disclosed by the optimal match (minimum cost) in relabeling host addresses in the binary tree that represented the network structure annotated with external information, e.g. host behaviors. Effectiveness of the attack was partly dependant on the completeness and accuracy on the external information. The work quantified in the worst case the amount of hosts deanonymized. It showed that generally partial prefix-preservation improved anonymity, but the sanitization that randomized subnets only sacrificed trace utility.

Brekne *et al.* proposed an attack that employs packet injection and frequency analysis to compromise individual addresses protected by prefix-preserving anonymization in multilinear time [13]. They also suggested modifications to strengthen the anonymization methods, such as to release the topology information gradually.

2.8 Summary

We laid the background knowledge on web traffic characteristics over low latency encrypted communication tunnels. We made two observations on VPN, SSH or TLS encrypted and proxied HTTP streams: (*i*) webpage download by HTTP is highly structured; and (*ii*) encryption and proxy do not severely alter the packet sizes, nor the packet ordering. Tor protects anonymity by layered encryption and multiple proxies. It is a reduced implementation of mix network, which sacrifices batching of packets for reduced delay to support interactive applications, and which eliminates dummy traffic for bandwidth efficiency. Tor operates under a security model where adversaries are assumed to be able to observe only a fraction of the Tor network and have control only on a fraction of the Tor nodes. Under Tor's security model, anonymity may still be compromised by exploiting Tor's low latency property or by correlating the traffic load. As the size of Tor cells is fixed, it poses a significant challenge to getting information about the underlying web traffic by the size channel. There is not yet any research on identifying websites over Tor by features of web object sizes.

We reviewed techniques to track traffic sources, including packet marking, website fingerprinting and flow watermarking. Packet marks are embedded into unencrypted packet headers for inspection. PPM (probabilistic packet marking) probabilistically embeds partial path information into packet headers. The path information thus spreads over multiple packets for reassembly. Existing schemes designed packet marks with sophisticated structure, most of which reserve a few bits to indicate the hop count from a marking router to the victim server. The differences between various DDoS packet marking schemes are subtle. There is not yet any metric for fair comparison across them because of their differences in model assumptions. As for website fingerprinting and flow watermarking, they have some similarities as well as differences. Both website fingerprinting and flow watermarking apply on encrypted and proxied communication channel. Existing website fingerprinting schemes are passive, while flow watermarking schemes are active. Existing website fingerprinting schemes use only side channel features related to sizes, and hence cannot withstand the countermeasure of traffic morphing. No website fingerprinting schemes have used packet ordering information, neither do they apply to Tor. Flow watermarking techniques are applicable to Tor, but they require monitoring both ends of the communication. The traffic analysis model of flow watermarking assumes candidates of a communication are available. Hence it verifies a pairwise communication relationship by embedding a watermark into the timing channel of the packet flow at one end of the communication, and verifying the watermark at the other end.

Chapter 3

Framework of Traffic Source Identification

We lay out our framework of traffic source identification, in which the domain of traffic source identification models is classified by the attributes of traffic model or investigator capability, and the criteria of scheme designs are discussed. The framework guides the construction of source identification schemes with respect to given traffic models.

3.1 Problem Statement

Traffic source identification is the investigation of network traffic so as to identify the communicating parties. When traffic obfuscation techniques are not used, traffic source can be easily identified from the IP addresses or message contents. However, when data obfuscation techniques (such as IP spoofing and data encryption) are used, it is no longer trivial to identify traffic source, and traffic analysis techniques are needed.

Traffic analysis is the process of monitoring the nature and behavior of traffic, rather than its content. We specify traffic contents as the application data being transmitted, and traffic behavior as the side channel information, such as timing, size and data structures.

There are two main traffic analysis approaches: active and passive. *Passive analysis* is an approach that merely observes the traffic in transmission to perform analysis. For example, passive website fingerprinting profiles and identifies a website from an encrypted and encapsulated HTTP stream, by using the observed side channel features from traffic as the website fingerprint. In contrast, *active analysis* is an approach that modifies the traffic behavior for analysis. For example, the acitve flow watermarking technique embeds an identification bit sequence into the timing channel of a traffic flow, and this identification bit sequence serves as a watermark to reveal the traffic source.

Our objective in this thesis is to study the relationship between traffic models and source identification approaches, and to design, evaluate and compare source identification schemes. Our traffic analysis for identifying traffic sources is closely related to web surfing. The traffic sources we are identifying can be web clients or web servers, depending on the application scenarios. For DDoS attack against web server, we monitor the traffic received by the server to identify the attackers; while for encrypted and tunneled web surfing, we monitor the traffic to identify the website being surfed by the client.

In this chapter, we present a taxonomy of traffic identification techniques based on the traffic conditions and the capabilities of investigator. In Section 3.2 and Section 3.3, we describe the components of a source identification model and the phases of operations in a source identification scheme, respectively. Built on the dependencies among model components, we present a decision tree based classification of traffic source identification models in Section 3.4. The classification shows the possible combinations of components that lead to source identification models, and hence guides the development of source identification schemes. We also describe in Section 3.4 three specific problem scenarios and the corresponding source identification models that we investigate. In Section 3.5, we examine the criteria for developing source identification schemes from models for the three problem scenarios presented in Section 3.4.

3.2 Components of the Source Identification Model

We consider that a traffic source identification model have the following four components: obfuscation techniques being applied to the source traffic, investigator's capability, features utilized for traffic source identification, and applications of the model. We summarize the model components and their details in Table 3.1, and present each component in detail in this section.

Obfuscation Techniques

Network traffic source can be obfuscated through forging IP addresses, encryption together with proxy, or pseudonymization, as described below.

With *IP spoofing*, the source IP address in the packet header is a spoofed address that is different from user's real IP. Forging of IP address in a packet header jeopardizes subsequent communication, hence it appears only in attacks for repelling responses and hiding attacker's identity. A forged IP can be a randomly generated bit sequence or an existing IP address in a victim network which the attacker wants to direct the responses to.

Table 3.1: Components of Source Identification Models

Component	Details
Obfuscation Technique	(a) IP spoofing (b) Encryption together with proxy (c) Pseudonymization
Investigator's Capability	(a) Passive (b) Active (b.1) Traffic stimulation (b.2) Mark embedding
Exploited Feature	(a) Timing (b) Size (c) Structure
Identification Application	(a) Attack forensics (b) Profiling user interests (c) Revealing network structure $\cdots$ *etc* $\cdots$

Encryption together with proxy creates an illusion that the communication endpoints cannot be identified, as proxy enables encapsulation of endpoint addresses and encryption protects confidentiality of the encapsulated data from an observer. Encryption and proxy are basis of many low-latency anonymous communication services, e.g. Anonymizer.com. VPN, SSH tunnel and SSL/TLS tunnel are also based on encryption and proxy. VPN multiplexes all connections between a source and destination pair into one.

Similar to IP spoofing, *pseudonymization* changes the source IP addresses in packet headers. Yet pseudonymization maps a traffic source to a (private) IP or a pseudonym usually consistently, and online communication with a pseudonymized traffic source can be supported. Pseudonymization is commonly used in sanitizing network packet traces or flow logs, before they are released for cross-organization research. It covers up the IP addresses and other sensitive data, while preserving the communication statistics. Prefix-preserving anonymization is currently the most recommended technique to perform pseudonymization. Pseudonyms are also used by servers to provide hidden services through Tor. An important objective of psedonymization is to defend against mapping of pseudonyms to individual hosts so as to protect privacy.

Tor provides by far the most reliable anonymous communication service. Tor network is an implementation of Chaum's design of "mixes" [14], but some operations in "mixes" are not included in Tor for efficiency concerns. In "mixes", besides performing layered encryption and using proxies, packets are repackaged into fixed size packets, so as to minimize the leakage of size related side channel information. In addition, packets are reordered to break the timing correlation, and dummy traffic are introduced to cover noises. However, packet reordering and dummy traffic are not implemented in Tor, because packet reordering introduces considerable network delay, and dummy traffic degrades the effective network throughput. Hence, layered encryption, multiple proxies and unified packet sizes are the anonymity protection mechanisms that Tor relies on.

Investigator's Capability

Traffic analysis can be carried out in a passive or active manner, constrained by the capability of traffic analysts. *Passive analysis* observes the traffic, where the operation is transparent to end users. It requires the least capability from a traffic analyst. *Active analysis* stimulates the traffic or embeds marks into packets or flows. It is able to obtain more information, but active analysis is not applicable to offline traffic analysis.

In passive traffic analysis, network communication are eavesdropped. Passive analysis requires gaining accesses to the desired network segments. Pervasive wireless communication medium and widely distributed bots make it easier to satisfy the requirement. Observation alone is sufficient for many applications. Most obfuscation techniques leak size or timing information, which renders them vulnerable to passive traffic analysis. For instance, encryption normally leaves message sizes unchanged.

As an approach to performing active traffic analysis, *traffic stimulation* changes the packet or flow behavior to reveal features of the source data. For example, to trace the attack paths of a flooding DDoS attack, controlled flooding exploits the fact that the packet rate of one flow affects the packet rates of other flows sharing a network link. Voluminous traffic is iteratively launched in suspected network links, for the victim server to determine if the network links are on attack paths. If a suspected network link is indeed on an attack path, then some amount of the attack traffic will be dropped during transmission, as a result of controlled flooding. A similar model has been developed to identify a Tor communication path, thus to link the otherwise unrelated flows to the same requester. Investigators manipulate the flow rates at certain Tor nodes. If the latency of a victim flow is found correlated to the manipulated traffic rate, then they are likely sharing (part of) a Tor circuit. These examples demonstrate that through stimulating the traffic rate, investigators are able to obtain additional information on the correlation between the manipulated flow and the monitored flow, which can then be exploited to uncover the communication path and identify traffic sources.

As another approach of active traffic analysis, *mark embedment* inserts marks into packets or flows for identification. For instance, in ICMP traceback, routers generate additional ICMP packets which duplicate part of the randomly sampled source flow packets and carry identifying marks of routers, for the purpose of tracking the source paths under DDoS attacks. Marks can be embedded into the side channel of flows when packets cannot be directly modified. Instead of changing some bits in packet headers, watermarks are embedded by modifying some flow characteristics, e.g. packet intervals. With sufficient redundancy, even slight perturbation on the timing of packets are observable across a network connection [94]. For instance, watermark embedded at the VoIP caller and extracted at the callee enables verification of the communication relationship by an investigator.

Exploited Features

Features exploited to identify traffic sources can either be extracted from the traffic itself, or injected by the investigator, and the choice of which is dependent on the underlying traffic model and investigator's capabilities.

Side channel features are related to traffic behavior instead of packet content. The side channel features exploited to perform traffic analysis are generally *size*, *timing* and *structure* related. These features can be properties of *packets*, *file objects*, or *protocols*. Packet features are less stable than features of file objects, which in turn are less stable than protocol features in general. The reasons lie in the different degrees of impact from network noise and labor required to modify these features. However, more stable features are harder to be extracted from obfuscated data. For example, to identify a webpage from VPN traces, the packet size information is available, while inline object sizes are difficult to extract because of interleaving object downloads, whereas structural information of order of the contacted third-party servers is concealed by encryption.

Besides side channel features being extracted from the traffic itself, features can also be injected by investigators. For example, in DDoS traceback, packet marks often contain router IPs and hop count from the marking router to the victim server. Sometimes they also include partial packet content and timestamp. There is much flexibility in designing the injected marks to packets for source identification. In contrast, flow marks should be designed to minimize interference, and hence has lower flexibility, and the marks are injected into and transmitted in the side channel of the traffic.

The selection of feature is limited by the source data and investigator's capability. Encryption together with proxy does not intentionally alter the size or timing information, yet not all features are observable after obfuscation, e.g. web object sizes. In comparison, dummy traffic, aggressive padding, and uniforming sizes intentionally reduce the distinguishability of size related features. Timing provides useful information in low latency communications, including web browsing, VoIP. Note that timing information is usually fuzzy, as it is affected by instant network conditions and the locality of data collection. Most website fingerprinting attacks observe the quantity and length of data packets incurred by surfing the website, but discard the timing information.

Applications of Traffic Source Identification

Traffic analysis to identify traffic sources or communication paths has a wide range of applications, including attack forensics, building user profiles, and identifying network structures. Some of the applications are for the surveillance of online security, while others demonstrate that the privacy protection provided by existing obfuscation techniques are less reliable than we once thought.

Identifying DDoS attack sources is an example of forensic application. Another forensic example is to identify the accomplice websites in XSS (cross site scripting) attacks, which spread malicious codes to facilitate harmful activities, such as stealing private information from clients. Yet another forensic application is to verify a communication relationship in lawsuits. Besides the content in communication, the communication relationship itself is also sensitive information.

Profiles of user interests on web browsing can be built from either packet traces or flow logs, by identifying the websites they visit, even if the source data are encrypted or anonymized. Packet traces presents some packet features and possibly features of web objects. Flow logs preserve features and statistics of flows and possibly object features, too.

Identifying subnet structures and key servers of an organization provides useful insights for the onset of an attack. Such information can be extracted from anonymized flow logs using traffic analysis techniques. It is a threat to business secrets and service availability.

3.3 Phases of Operations

A traffic source identification model in general contains two phases of operations, which are data collection and data analysis.

Data collection refers to the processes of sampling network traffic and extracting from them the features desired for analysis. For instance, website fingerprinting attacks build a database of website fingerprints and capture the victim HTTP streams in the data collection phase. Note that it is not necessary to build website fingerprint profiles before the testing HTTP streams are captured. For active traffic analysis, *data collection* also includes the process of traffic stimulation or mark injection. The medium that stores the injected data can be packet headers or some side channels of packet flows.

Data analysis determines the traffic sources from extracted features by comparing the features with known source profiles. A profile can be a distribution of packet marks associated with a router, or a unique mark attached to a packet flow, or some side channel features forming the traffic fingerprint of a website. Data analysis may need to perform fuzzy matching and evaluate the similarity between the profiled and the extracted testing features, due to various noises. The sources of noise can be variations in the instant network conditions, or difference in the localities where the profiling and testing traffic are collected.

3.4 Classification of Source Identification Models

We have presented in Section 3.2 the components of source identification models, here we use the decision tree representation in Figure 3.1 to illustrate the classification of source identification models by these components. Our framework is general enough that existing traffic source identification techniques are encompassed, and future techniques can potentially be incorporated. It also assists in deriving new traffic source identification models.

Figure 3.1 illustrates the classification of traffic source identification models. In the top level, source identification models are classified by the source obfuscation techniques, which can be (1) spoofing IP addresses, (2) applying encryption together with proxy, e.g. VPN or Tor, or (3) mapping source IP addresses to pseudonyms, such as in traffic log anonymization or protecting hidden servers. Under each type of source obfuscation, analysts can develop passive or active source identification approaches depending on their capability. The approaches include (1) eavesdropping the communication, (2) stimulating the communication or (3) embedding marks into packets or flows. The source identification models can be classified in finer grain by the traffic features they exploit.

Among the models shown in Figure 3.1, passive traffic analysis is adopted by (a) traffic logging for DDoS traceback, (d) passive website fingerprinting, (e) statistical disclosure that identifies long term communication probability over Tor, and (i) traffic logs deanonymization. Active approaches are taken by (b) controlled flooding and (c) packet marking for IP traceback, (j) hidden service discovery over Tor, and (g) flow correlation through stimulating the traffic rate or (h) embedding watermarks. In addition, we propose (f) an active website fingerprinting model that stimulates HTTP traffic to expose additional features to identify websites.

Each path from root of the decision tree to a leaf node represents the combination of components of a traffic source identification model. For example, in the model of (b) controlled flooding, the obfuscation technique used by attackers is (1) IP spoofing, the investigator is able to (1.2) modify the traffic rate of certain flows, and monitor the feature on the (1.2.1) traffic rate correlation with the attack flows, so as to identify the attack paths taken. For (h) flow watermarking, the model handles source obfuscation techniques of (2) encryption and proxy, where the investigator (2.3) embeds some secret code as a watermark in one end of the traffic flow for verification across the anonymous network, and the side channel feature selected to carry watermarks is (2.3.1) packet interval. We

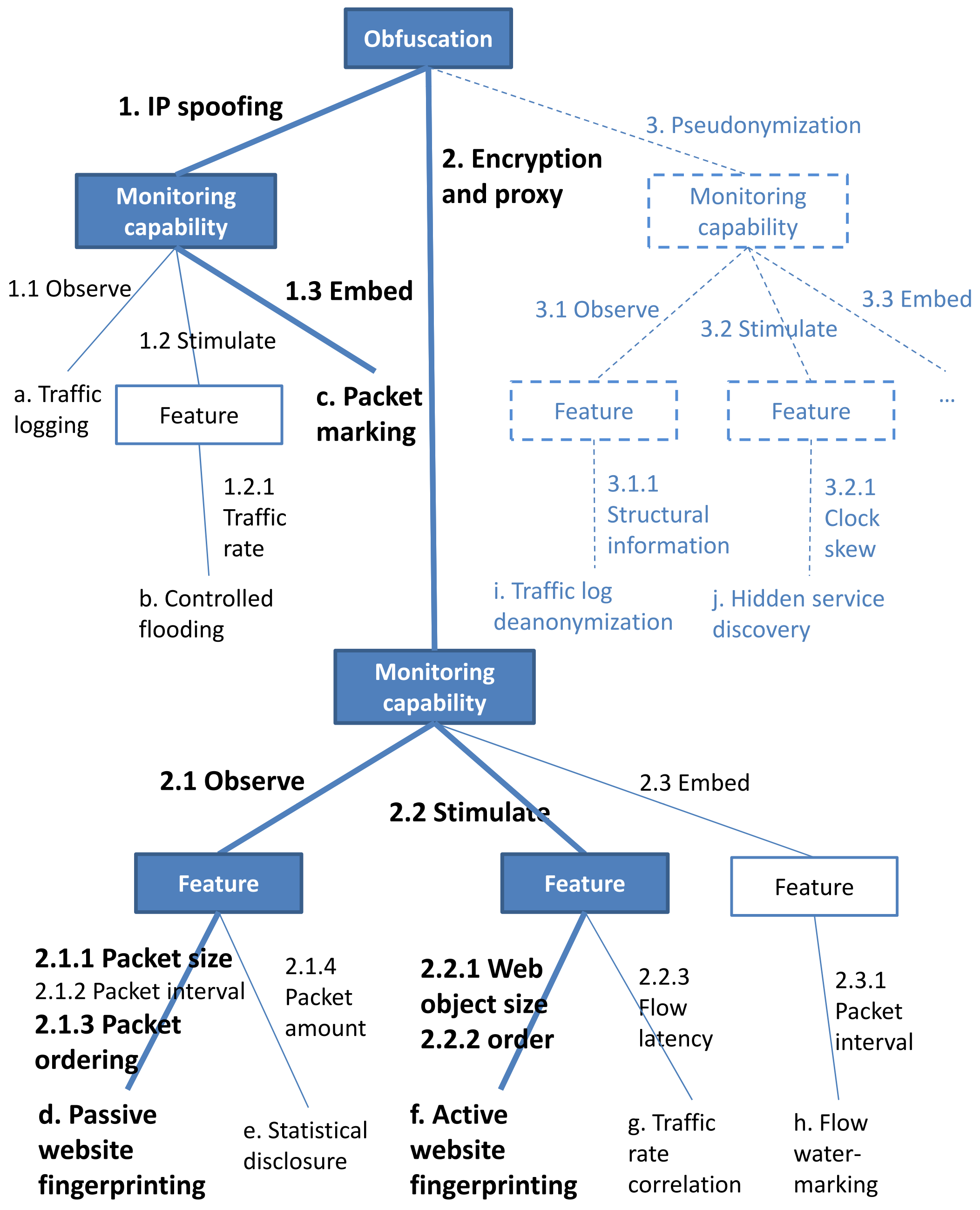


Figure 3.1: Classification of traffic source identification techniques
(The dotted branch on models to identify pseudonymized traffic sources is not our focus in this thesis)

develop our model of (f) active website fingerprinting through forming novel combinations of the component values. While all existing website fingerprinting approaches are passive, we use (2.2) an active stimulation approach to obtain (2.2.1) inline web object features so as to identify a website.

In this thesis, we investigate in detail three traffic source identification models applicable to different constraints and traffic models, namely, *packet marking*, *passive website fingerprinting*, and *active website fingerprinting*. The combination of their model components are highlighted in Figure 3.1. Below we give an overview on the traffic characteristics of these applications and the corresponding source identification approaches we take. As we are not looking into details of source identification models for pseudonymized traffic, the corresponding branch in the decision tree is faded.

Packet marking Packet marking applies when the source IP is spoofed and the packet header is not encapsulated or encrypted. PPM (probabilistic packet marking) probabilistically embeds partial path information into packet headers. The path information thus spreads over multiple packets for reassembly. We assume routers are cooperative in embedding packet marks and the packet stream for source identification contains sufficiently many packets. The scheme designed following this general approach is presented in Chapter 5.

Passive Fingerprinting When the source packet headers are encapsulated into packet payload and encryption is applied on the packet payload, packet marks embedded into the source packet headers can no longer be inspected. We turn to the side channels of traffic flows to extract identifying information about traffic sources. Passive traffic analysis is applicable to fingerprint websites when the HTTP streams reveal sufficient information about the sources.

Website fingerprinting aims to identify the website accessed through some encrypted tunnels. Website “fingerprints” are profiled on side channel features observed from the HTTP streams, where an HTTP stream is a stream of packets sent and received because of accessing a website. Existing website fingerprinting schemes target size related features, relying on the fact that encryption does not obfuscate the amount of data transmitted. Our passive website fingerprinting scheme exploits packet ordering information in addition to the usual size related information. It is described in Chapter 6.

Active Fingerprinting Existing website fingerprinting models are passive such that the investigator does not modify the traffic but only eavesdrops. However, when the web browsing traffic is not only encrypted and sent through proxies, but also the data transmission unit is fixed, the packet level characteristics are largely concealed.

Hence passive fingerprinting may not be sufficient to identify a source website, given the obfuscation techniques of encryption, proxy and uniform data transmission unit, such as in Tor.

We propose an active website fingerprinting model for the above traffic model, in which the investigator is capable of modifying the traffic behavior to expose object level characteristics of the source. In contrast to packet marking or flow watermarking which embeds selected marks into a packet flow, the features monitored in active fingerprinting originate from the source.

In the active fingerprinting scheme we develop, the selected features to website fingerprinting are web object sizes and order. Interleaved data transmission of several objects makes it difficult to extract the selected features. We delay the HTTP requests so as to reveal the inline object sizes and order of the source website. Our active website fingerprinting scheme is presented in Chapter 8.

3.5 Source Identification Scheme Design Criteria

In the previous section, we described three specific problem scenarios and the corresponding source identification models. In this section, we examine the criteria for developing concrete source identification schemes from these models. There are a number of technical designs being involved, which include packet mark design in DDoS traceback, fingerprint feature selection, and fingerprint similarity comparison in website fingerprinting.

Packet Mark Design Criteria

Packet marking schemes have a constraint that there are only limited number of bits available (typically 16 bits) in the packet header for marking. The number of routers or sources we need to identify exceeds the number of unique values the limited number of bits can represent. As we need to identify each individual router and collisions of packet marks cannot be avoided, each router is associated with a distribution of packet marks.

The packet marks should be designed such that it is easy to identify the contributing markers accurately and efficiently from a combination of packet mark distributions. Packet marks should be designed to minimize collisions, which translates to minimizing the collisions in packet mark distributions from different markers. Further analysis and evaluation on the design of packet marks are presented in Chapter 4.

Fingerprint Feature Selection Criteria

The essential criteria for selecting fingerprint features are fingerprint *consistency* and *distinguishability.* Consistency provides the basis for comparison between a fingerprint and its profile, while distinguishability provides contrast between a fingerprint and the profiles of others.

Consistency means the fingerprints of the same traffic source sampled at different time, at different network locations, and at different traffic rate of other flows sharing the communication path are sufficiently close. It ensures the testing fingerprints are comparable to their profiles. It is easy to understand that the intersection of feature values from different fingerprints satisfy consistency. However, as the sample size increases and multiple settings are considered, the number of exact matches in feature values reduces. The extreme is null intersection, which means there is no feature value left in the fingerprint. We need more flexible methods that tolerate some inconsistency in the selected feature values when developing a source fingerprint profile. The consistency of feature values is dependant on the traffic characteristics. For instance, if packets are padded with a randomized amount of bytes each time, then packet sizes are not consistent and hence not suitable to be fingerprint features.

Distinguishability requires the fingerprints of a traffic source be distinguishable from fingerprints of others. The less expected feature values among all fingerprints give more identifying information. We note that sometimes feature values may be not very inconsistent but if they provide additional distinguishability of fingerprints, the features should be selected. For example, in website fingerprinting, the embedded object sizes expose lots of information of the webpage, most of which enables the source identification. Although they are less consistent compared to the request sizes due to possibly object updates, the feature on object sizes should be selected.

It is desirable that only a small sample is needed to build a stable profile and the time to build a profile be short. In applications like website identification, long sample collection and profiling time reduces the identification accuracy because website contents can be updated during the data processing.

Similarity Measurement Criteria

Certain prior knowledge of the traffic sources are required for identification, which are in the form of their associated packet marks, flow marks, or fingerprints. During testing, we extract feature values from the traffic and evaluate their similarity with profiles of candidate sources. There are three main criteria of similarity measurement.

Firstly, the dependency and relationship of feature values should be captured in the comparison. In the website fingerprinting schemes we propose, we find the ordering information of featured packets or of HTTP object sizes adds valuable information to website identification. If we employ SVM, Jaccard's coefficient or Bayes network for fingerprint comparison, then the ordering information would be lost.

Secondly, the accumulation of errors from misjudgement should be minimized. In the analysis of a large amount of data, one may attempt an iterative process that isolates a group of related data to make a judgement on them, then remove these data from the total collection before repeating the process of data isolation and judgement. However, due to the ambiguity in the grouping of data, e.g. collisions of packet marks associated with different routers, the removal of data in some previous iterations affects the accuracy of judgement in the later iterations. The iterative approach is applicable if the grouping of data is certain. Otherwise, from the point of accuracy, it is desirable to always analyze the data from the total collection.

Lastly, it is desirable that the similarity measurement method be both memory and computation efficient. We would like to locate the attack sources the soonest we can, in order to deploy remedy policies and to catch the suspects in time. The comparison method should be efficient enough to scale well with the number of potential traffic sources. Specially, it is desirable that the time and memory used to identify a traffic source be linear to the number of profiles.

Scheme Evaluation Criteria

Effectiveness is the ultimate evaluation criterion of a traffic source identification scheme. Currently, it is quantitatively measured using identification accuracy in a classification problem; and it is evaluated using false positive rate (FPR) and false negative rate (FNR) in a detection problem. However, these metrics of identification accuracy, FPR and FNR are sensitive to the dataset used for evaluation. Better numerical results in one dataset may not represent a superior performance in another. It leaves the relative quality of different schemes indeterministic. Fair comparison on the quality of packet marks, flow marks or source fingerprints may require better evaluation metrics.

3.6 Summary

We present a framework of traffic source identification models. The framework considers each source identification model has components of obfuscation techniques, investigator's capabilities, features, and applications. The model components affect one another. User applications specify the privacy requirement and the data obfuscation method. Whereas data obfuscation method inspires the traffic monitoring approach as well as the selection of features for analysis in a traffic source identification application. The framework captures that generally a traffic source identification scheme contains data collection and analysis these two phases of operations.

Traffic source identification models are classified in the framework by their components. The models to apply in different problem scenarios are dependent on the obfuscation techniques and investigator's ability. We can opt for modifying the packet headers to embed marks, if the packet header is not encapsulated or encrypted. We can also opt for performing passive or active traffic analysis on the side channel characteristics of the exchanged packets between a pair of communication endpoints, where the usual side channel features utilized are packet sizes and intervals. Passive analysis suffices if substantial side channel features are observable, otherwise active analysis techniques are required to expose additional information.

Under the framework, we highlight the source identification models that we investigate, and examine the criteria in designing the source identification schemes, in problem scenarios of DDoS traceback, passive website fingerprinting over VPN and active website fingerprinting over Tor.

Chapter 4

A General Probabilistic Packet Marking Model

We consider a traffic model of DDoS flooding attacks with IP spoofing. We present a general packet marking model of DDoS traceback schemes that captures and compares their common phases of operations, namely, packet marking and path reconstruction. We propose using entropy as a fair evaluation metric of packet marks quality across the schemes to predict the traceback accuracy.

4.1 Overview

In Distributed Denial of Service (DDoS) attacks, many compromised hosts flood the victim with an overwhelming amount of traffic. The victim's resources are exhausted and services to users become unavailable. DDoS attacks paralyzed high-profile web sites, including Yahoo, CNN, Amazon and Microsoft, for hours to days [33]. DDoS attacks were launched against and brought down several root DNS servers, lasting for hours [2].

During a DDoS attack, attack nodes often perform address spoofing to avoid detection. An IP traceback mechanism aims to overcome address spoofing and uncover the attack paths or sources. While traceback is motivated by DDoS attacks, it also benefits analysis of legitimate traffic. Potential applications of traceback include traffic accounting and network bottleneck identification. In Probabilistic Packet Marking (PPM), routers probabilistically mark the packets they transmit, so that the victim can trace the attack paths up to their sources, based on the packets it received [69]. A packet is marked by writing to the reusable bits in the IP header. We call the strings written as *tags*.

In this chapter, we present a general model for PPM schemes by formulating it as an identification problem, in which each node (or edge) marks packets probabilistically according to an associated distribution of tags. Based on a collection of received packets, the victim attempts to recover the markers' identities. Since there are multiple markers, this collection is made up of samples from a mixture of the markers' associated distributions. Hence, traceback is essentially identification of each individual distribution contributing to the mixture. By viewing each distribution as a point in a high dimensional space, we can see that this model is closely related to the studies of collusion-resistant codes and fingerprinting [17, 82]. Thus, one may choose a known collusion-resistant code to assign codewords to different markers. The main difference between traceback and other applications of collusion-resistant fingerprinting is the scale of the problem. The number of markers can be more than a thousand, which is much larger than typical applications of collusion-resistant fingerprinting.

4.2 Probabilistic Packet Marking (PPM) Model

4.2.1 Problem Formulation

During a DDoS attack, a victim V receives an overwhelming amount of packets transmitted over multiple paths, each at a packet rate greater than R_{att}. A router along an attack path can embed information of its identity into the packet headers. We call such router a marker. Alternatively, the router can embed information of its identity and the next hop identity into the header. In this case, we treat the edge as the marker. Let $\mathcal{U}$ be the set of all possible markers, and $\mathcal{M}$ be the set of markers along the attack paths. Our goal is to identify $\mathcal{M}$ among $\mathcal{U}$. Let $m = |\mathcal{M}|$, and $n = |\mathcal{U}|$. Each marker is allowed to mark L bits in a packet header[1]. The PPM problem is interesting when 2^L is smaller than n. The problem thus becomes how to use multiple L-bit tags to identify elements of $\mathcal{M}$.

We measure the performance of a marking scheme by the false negatives ratio α, which is ratio of the number of markers not correctly identified over m, and the false positives ratio β, which is the ratio of the number of markers wrongly declared as on attack paths over m.

4.2.2 Components of PPM

The operations of a PPM traceback scheme can generally be divided into the following components: marking of packets by the routers, choice of tags used and reconstruction using information from marked packets by the victim. In the following, for each of these components, we first present the general idea and then highlight the design of RPM.

[1]Typically, as indicated in [69, 74, 24, 35, 93], the 16-bit packet identification field in the IP header is used. The packet identification field is used in less than 0.25% of the time to re-assemble fragmented packets [77].

Marking by Routers

In our model, each marker is associated with a distribution D on the L-bit tags. Such associations are pre-assigned and fixed throughout the marking and identification process. Consider a marker with identity i and its assigned distribution D_i. When it receives a packet, the marker chooses with probability ϵ, an L-bit tag s according to the distribution D_i and mark the packet with s.

The probability ϵ is a parameter that is the same for every marker. It is possible that some packets arrive at the victim without being marked. We assume that the bits in those unmarked packets are random and are uniformly distributed.

Since the marking process needs to be very efficient, sampling from the distribution D_i must be a simple operation. Thus, in RPM and other related work, only uniform distribution on a finite set is considered. Essentially, the marker just randomly and uniformly picks a tag s from a pre-assigned set. Let us write the probability density function of the distributions assigned to the marker with identity i as D_i. That is, $D_i(x)$ is the probability that the tag x appears in a packet marked by i. Since we assume that the distribution is uniform on a finite set of tags, $D_i(x) = 0$ or c for some constants. Thus, WLOG, we can also represent D_i as a subset of L-bit tags. Let us write this set as X_i where $x \in X_i$ iff $D_i(x) > 0$.

Choice of Marking

Consider a set of markers P. The collection of tags received by the victim follows a distribution which is a mixture of the distribution associated to the markers in P. Deriving the mixture distribution D_P from P is not straightforward due to the effect of the probability ϵ. Suppose that the only markers are $P = \{i_1, i_2\}$, where i_1, i_2 are along the same path and i_1 is nearer to the victim, then

$$D_P(x) = \epsilon D_{i_1}(x) + (\epsilon - \epsilon^2) D_{i_2}(x) + (1 - 2\epsilon + \epsilon^2) 2^{-L}$$

for every tag x.

Consider two sets of markers, P and Q. Let D_P and D_Q be the distribution of the tags received from P and Q respectively. If $D_P = D_Q$ then the victim is unable to distinguish whether the samples is from P or Q. If D_P is close to, but different from D_Q, an unreasonable large number of packets may be required to distinguish them. Hence, as an approximation, we take the mixture distribution of D_P as:

$$D_P(x) \approx \epsilon \sum_{i \in P} D_i(x), \quad \text{for all} \quad x$$

Thus, if

$$\sum_{i\in P} D_i(x) = \sum_{j\in Q} D_j(x), \quad \text{for all} \quad x$$

then it is difficult to distinguish packets from P and Q. In such case, we say that a collision has occurred.

In general, the associated D_i should be chosen so as to minimize collisions, for example, using a collision-resistant code. However, due to the size of the problem, RPM employs random codes as it is more practical. For each marker i, the D_i is generated from a random function (for example, SHA) with i as input. We will show in later section that use of random codes provides sufficiently good results.

Note that given the network topology, certain sets of markers are more likely to appear compare to others. Ideally, the D_i can be chosen by considering the network topology to avoid collisions. However, in practice, individual marker lacks the global topology information. Hence, a reasonable approach, as used in RPM, is to assign the D_i randomly.

Reconstruction Algorithm

Due to the large problem size, a PPM scheme needs to address both the choice of D_i and the identification algorithm at the same time. Enforcing certain relationship among the tags in D_i for each i can aid identification. For example, in RnL [35], all tags from a marker contain the same "cord". That is, a substring (the cord) of every tag from a marker is the same. An example given by [35] invests 15 bits for the cord when $L = 25$. Based on the cord, the received packets can be easily divided into smaller groups. Next, packets in different groups are identified independently. Since the number of packets in each group is smaller than the total number of packets, the task of identification become easier.

RPM does not exploit special structure in the tags for reconstruction. Instead, it assumes prior knowledge of neighborhood nodes and uses a hop-by-hop reconstruction. Such assumption is also made in FMS, AMS, and FIT.

Marking Structures of Existing Schemes

Existing schemes often divide the L-bit tags into multiple components to include structures that contains path information or aid reconstruction. If the number of unique tags generated by a marker is h, i.e. $|X_i| = h$, there is a $\lceil \log_2 h \rceil$-bit component which labels the tags. Let us call it the *hash index*. For example, if $|X_i| = 2$, then the hash index starts from 00, 01, 10 to 11. In addition, there is a component that is determined from the hash index and the marker's identity. Let us call this the *hash value*. A few schemes, for example, FMS [69], AMS[74], and FIT[93], allocate 8 bits for the hash value and 2-3

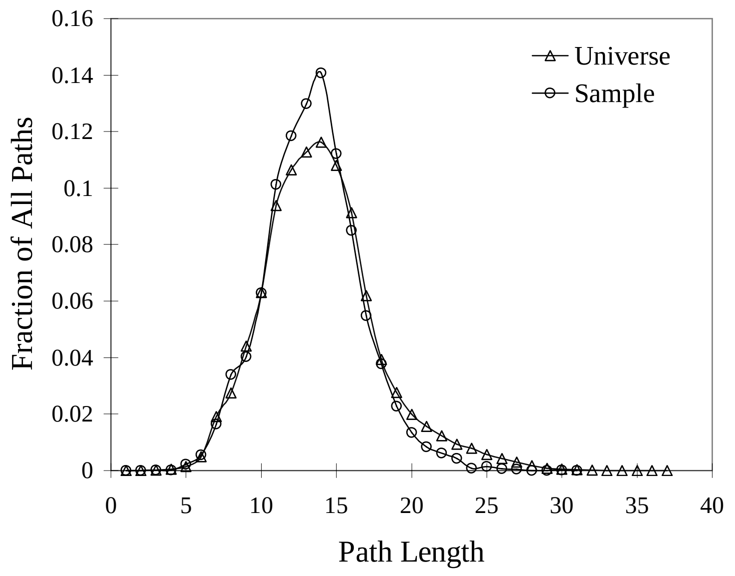


Figure 4.1: Path Length Distribution

bits for the hash index, and employ different hash functions to compute the hash value. RnL [35] also reserves some bits for the hash index and the hash value.

4.3 Analysis

4.3.1 Entropy of Packet Marks

One measure of the quality of a chosen D_i is the entropy of the mixture of distributions received by the victim. Intuitively, uniformly random packet marks deliver highest entropy. Higher entropy carries more bits of information, which can reduce the false negatives ratio α and the false positives ratio β. Entropy measure provides a means to evaluate the performance of PPM schemes. A good marking scheme should strive to achieve high entropy packet marks.

Some existing schemes trade off some entropy in the tags for easy path reconstruction; but they underperform in effectiveness. In particular, we will show that the use of hop count to indicate distance from markers to the victim results in lower entropy than uniformly random bits.

The design of FMS, AMS and FIT are very similar. They allocate 8 or 13 bits for hash value, 2-3 bits to indicate the hash index, and 5-6 bits to keep the hop count from the marker to the server. AMS slightly improves over FMS on the traceback accuracy because it uses a better hash function. FIT in turn outperforms AMS by reducing hash collisions. It introduces longer hash outputs, and encodes node information instead of edge, since there are fewer nodes than edges. However, for FIT, $L = 21$ instead of 16. It

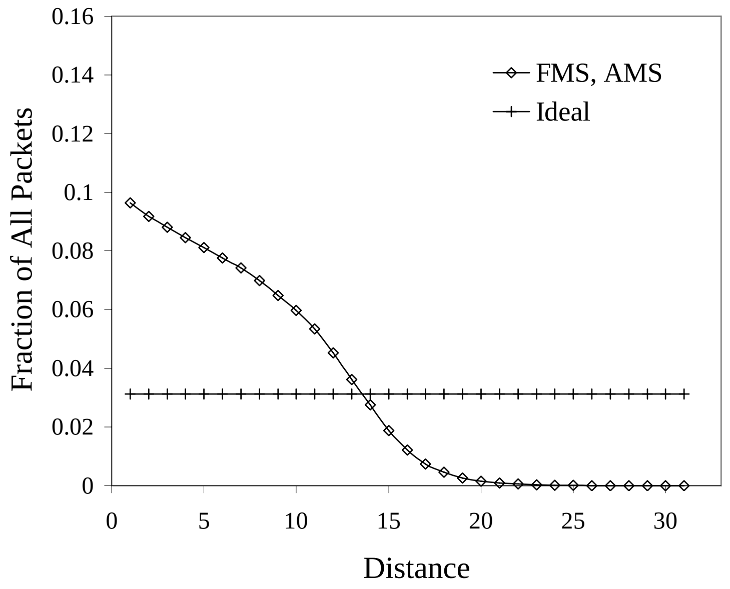


Figure 4.2: Distribution of Distance Values from Packet Marks

updates 5 bits of the Time-To-Live (TTL) field, besides marking over the 16-bit packet ID field. This changes the semantics of TTL. For instance, an intermediate router can now prolong the lifetime of a packet by enlarging its TTL value.

FMS and AMS utilize 5 bits of distance information to aid in the hop-by-hop path reconstruction. The entropy of such distance information is low, resulting in inefficient use of the marking bits. The reasons are two-fold. First, as shown in Figure 4.1, the path lengths between any router and a server are averaged at around 16 hops. This result is obtained using the topology with 225,415 edges and derived from Internet mapping project. While 5 bits can represent a distance of up to 31 hops, the upper half of the distance ($>$ 15 hops) values occur with low probability and are under-utilized. Second, the schemes allow routers to reset the distance information. Distance information embedded by remote routers are likely to be overwritten by routers closer to the victim. Again, as shown in Figure 4.2, the distance field ends up more frequently with small values. In sum, the distance field has lower entropy than uniformly random bits. The marking bits are not efficiently utilized by incorporating the distance information.

Now we measure the performance of different PPM schemes with respect to their tags' entropy. For comparison, we include marking schemes of AMS and RnL. RnL or randomize-and-link, uses large checksum cords to facilitate fast identification. RnL is selected because its packet marks are designed to have high randomness. The drawback of RnL is that its path reconstruction process is not scalable. The scheme RPM will be presented in next chapter. The tags RPM generates have an entropy very close to that of RnL, but RPM has a scalable path reconstruction.

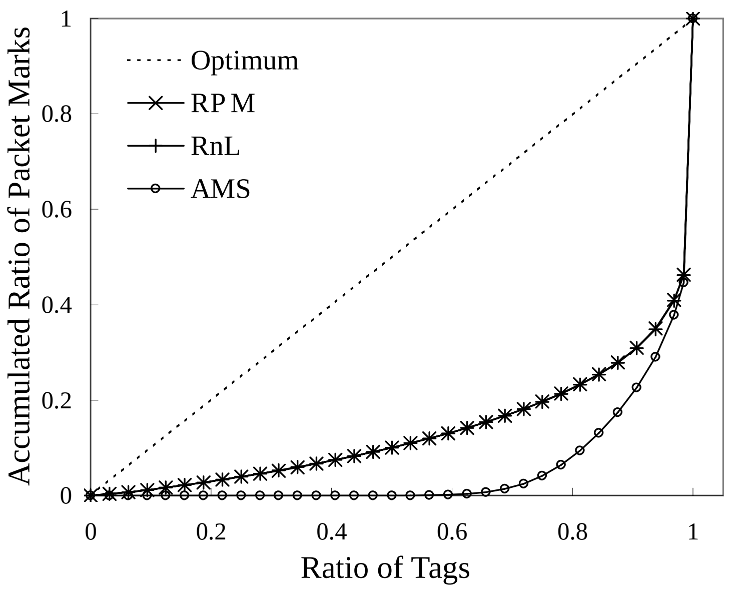


Figure 4.3: Packet mark value distributions

For a fair comparison, both RnL and AMS generate 16 16-bit packet marks for each edge and have a marking probability of $p = 1/16$. RnL uses 10 bits for the checksum and 2 bits for a hash fragment; AMS uses 5 bits for the distance and 7 bits for the hash value. The simulation uses the same topology as the previous experiment. Every edge in the topology generates 1,000 packet marks according to the marking schemes.

Figure 4.3 shows how the accumulated ratio of packet mark varies with ratio of tags received by the victim. Each point in the plot corresponds to a bin size of at least 1,000 distinct tags. In the ideal case, the packet marks distribution should be uniform and is shown for comparison purpose. The plot clearly shows that the distribution generated for AMS and RnL are skewed and much worse than the uniform distribution. For AMS, up to 70% of all tags are carried by a small number of packets ($< 1\%$). RnL performs better, where about 80% of all tags are carried by 20% of the packets. Another interesting observation is that a large number of packets ($> 50\%$) carry small portion ($< 2\%$) of all distinct tags for both AMS and RnL.

Given that the packet marks generated locally by RnL is fairly uniformly and randomly distributed, it may be surprising that the tags received by the victim is so clustered. This clustering effect can be attributed to the fact that markings are performed independently by each router. As a result, tags generated by routers close to the victim tend to be received by the victim while the routers further away tend to have their tags overwritten. One approach to increase the randomness of the tags received is to allow routers further away to mark the packets more often. However, such approach will require additional coordination and global message exchanges which are not considered in this work.

Nevertheless, Figure 4.3 clearly shows that tags collected from RnL are much more uniformly distributed than AMS. The average entropy computed for RnL tags is 12.62 bits (out of 16) and tags from AMS has an entropy of 11.99. The entropy and distribution of tags from RPM are very close to RnL. Hence, it is expected that RnL should have much higher traceback accuracy than AMS. RPM's traceback accuracy is expected to match RnL, but its path reconstruction process is more efficient.

To verify the utility of using entropy as a measure of marking performance, 1,000 attack paths, or equivalent to over 4,650 attack edges generate tags for identification, using the parametrization above. The performance of RPM will be presented in next chapter. All edges in the network are tested for malice which makes the result independent of the reconstruction scheme. RnL has a false positives ratio β of 0.19, while the β for AMS is 5.56. The result clearly shows that schemes with higher entropy can achieve much better performance (lower β).

4.3.2 Identification and Reconstruction Effort

From marked packets, the victim wants to reconstruct the attack paths. This can be done by first identifying the markers, and then deriving the paths. Exhaustive enumeration of all subsets of markers, and estimating their respective packet rate is infeasible. To aid identification, structured information can be embedded into the tags. This information facilitates association between packet marks and the marker, or links different packet marks generated by the same marker. There are generally two assumptions made in the reconstruction, no network topology information available or available of (partial) topology information.

Without Topology Information

RnL is designed for a marker to transmit a message, which can be the marker's identity. In RnL, the reconstruction of the message, or identification can be carried out without prior knowledge of the network. It associates fragments of a path information message using checksum cord. A fragment has v (3 to 6) bits, but 8 to 11 bits out of 17 is allocated to the checksum, so as to reduce the likelihood that different markers producing the same cord in the whole network. Identification uses checksum as an associative address of message fragments, as it is invariant for a message. The verification of valid fragment combinations is expensive, particularly when many markers have the same checksum. By using combinatorial search to reconstruct multiple attack paths, the scalability of RnL is limited.

The algebraic approach [24] proposes an interesting way to reconstruct the attack path. In this approach, only a small number of packets is needed for reconstruction. The scheme has flexible parameterizations, with the length of a packet mark ranges from 18 to 25. It uses s bits to instantiate a random variable, and use router IP addresses as the coefficients to construct a polynomial. v bits are allocated to store the evaluated polynomial. h bits are assigned to keep track of the number of participating routers, which is translated into the degree of the polynomial. The values of v and h are tunable depending on the total number of bits allocated to a tag. The entropy of each component is high. However, the encoding and path reconstruction processes are expensive. Even though BCH decoding can be employed to find a high degree polynomial among samples, no known algorithm can efficiently identify multiple polynomials among samples. That is, its path reconstruction does not scale with multiple attack paths.

With Topology Information

An example scheme that uses topology information is AMS. AMS assumes that the upstream router map is available. It subdivides the path information to track the hop count from the marker to the victim. The distance information binds different fragments of a message. It reduces the path reconstruction complexity by limiting the combinatorial search of messages to each distance. However, as shown in Section 4.3.1, storing distance information reduces entropy of tags.

In general, reconstruction without any topology information is expensive. Hence, reconstruction with topology information is more practical. For effectiveness, it is important that the information associating tags to markers have high entropy. At the same time, using more bits to limit the search space can increase the reconstruction efficiency. In the next chapter, we will present our algorithm RPM that takes all these considerations into account.

4.4 Discussions on Practical Limitations

Topology

Effectiveness of the identification of attack paths is conditioned on the correctness of topology information. Network topology could be obtained through probing with trace route utility, or from available results of network structure analysis, e.g. Internet mapping project. Note that trace route can also be exploited by penetration testers to gather information about network infrastructure and IP ranges around a given host. So supplying detailed information about internal pathways is restricted for privacy and security reasons in many organizations. However, a topology map with only gateway information could already be helpful to identify attack paths, and such information is relatively easy to obtain. The network topology we obtain may have slight variations from the topology during attack. Timeliness of the release of network topology thus affects the correct identification of attack paths. However, our attack path reconstruction is not very sensitive to the accuracy of topology. We need only the adjacency relationship among routers to progress the path reconstruction hop by hop upstream.

Router cooperation

PPM requires router cooperation across autonomous systems (ASes). It may impact the practicality of PPM, because individual AS makes independent policies. Incentives for performing packet marking could be low for upstream routers. However, ASes that contribute to protect servers in other ASes should receive help and cooperation in reciprocal. And identifying the downstream routers of attack paths is urgently needed by packet throttling type of defences. We note that PPM does not require contiguous routers performing the task. Selective cooperative routers scattered over the network would still make it effective. Suppose only gateways participate in the marking, not only the internal structures of ASes have better privacy, but also the scale of the identification problem is reduced. We are potential in more accurately identifying the ASes where attackers originate, then the ASes can identify and handle the attack sources locally.

4.5 Summary

IP traceback has been an actively researched DDoS defence. For attack packets with spoofed source addresses, IP traceback traces the paths they traverse up to the sources. Traceback also benefits traffic accounting applications, such as tracking clients' bandwidth utilization, or locating the bottleneck links in the network.

In this chapter, we present a general model for PPM schemes. The general model provides a platform for PPM schemes comparison and helps to identify the appropriate system parameters. We also show that entropy is a good predictor of traceback accuracy and the use of hop count information in tags reduces the entropy.

Chapter 5

Random Packet Marking for Traceback

From the analysis we made on the general packet marking model in the previous chapter, we instantiate a probabilistic packet marking scheme named Random Packet Marking *for IP traceback. Our scheme improves the packet marks quality through increasing their randomness, and hence significantly improves the accuracy of identifying DDoS attack paths.*

5.1 Overview

Guided by the general probabilistic packet marking model presented in Chapter 4, we design a PPM scheme, called Random Packet Marking (RPM). The marking process is very simple, and it is a direct implementation of the model. No sophisticated structure or relationship is required for the tags. For example, many existing schemes divide the allocated bits into groups and different groups have different functionalities, like hop-count, hash value etc. In contrast, RPM treats all bits equally. For the reconstruction of marker identities, RPM employs a hop-by-hop reconstruction similar to AMS [74]. Hence, some prior knowledge of network topology is required. Simulation results show that RPM significantly outperforms AMS in acquiring higher traceback accuracy. Compared to schemes based on algebraic coding [24], RPM has much lower reconstruction cost and achieves higher scalability for the same number of attackers and packet markers.

The effectiveness of RPM demonstrates that, with knowledge of the neighboring nodes, it is undesirable to enforce structures in tags, since the structures impose constraint on the choice of tags. Without the constraint introduced by structures, each packet can carry more information and the chance of collision (that is, false positive) can be reduced.

5.2 Random Packet Marking (RPM) Scheme

As mentioned in Section 4.3.2, RnL is designed for scenarios where knowledge of the network topology is not available during identification. Without network topology, it is computationally difficult to identify large number of markers, although sufficient information is hidden in the tags. On the other hand, some PPM schemes facilitate easy identification but sacrifice some randomness in the tags, reducing the amount of information tags carry. For example, AMS exploits topology information and encodes hop count to facilitate efficient reconstruction; but the entropy per tag is low.

In this section, we present RPM that naturally follows from our model. RPM achieves both high entropy tags and efficient reconstruction. It does not divide a packet mark into components. All available bits are allocated to tags associated with markers. The effectiveness of RPM is influenced by the collision probability of the distributions D_i. Yet RPM is general enough to support any functions in assigning D_i. RPM caters for high entropy in the mixture of packet mark distributions. With a reasonable D_i assignment function and a proper setting of system parameters, RPM suffices in obtaining high traceback accuracy. Its path reconstruction is lightweight and scalable, aided by the topology information. RPM improves over AMS in that its marking scheme is simple, fast, and the marks have high entropy. At the same time, RPM simplifies RnL and extends it to identify large number of markers with topology information.

5.2.1 Packet Marking

We choose to employ edges as the markers. RPM works on graph structured networks. Edge encoding facilitates a hop-by-hop reconstruction of the attack graph.

In this scheme, the size of the set of tags X_i (recall that $x \in X_i$ iff $D_i(x) > 0$) is the same for every marker i. Let

$$h = |X_i|.$$

The association of X_i to an edge i is obtained using a known hash function, for example, taking the first L bits from the output of SHA. For an edge with IP addresses IP_1 and IP_2, let us represent the identity of this edge as $i = IP_1 \| IP_2$ where $\|$ is concatenation. The set X_i assigned to i is

$$\{ \; H(r \| IP_1 \| IP_2) \;\; | \;\; r = 1, 2, \ldots, h \; \}$$

where H is the hash function whose output is a L-bit sequence.

For clarity, let us recall the marking process. For the marker i with the associated X_i, when a packet is received, the followings are carried out:

- With probability ϵ, randomly and uniformly picks a x from X_i, and write x into the packet header.

There are two important parameters for us to determine, ϵ and h, which will be discussed in Section 5.3.1. An example in our experiment takes $\epsilon = \frac{1}{16}$, $L = 16$ and $h = 2^4$.

Note that the markers make decisions independently and they mark packets by overwriting existing values. It evades the potential domino effect from malicious packet mark manipulation. Even if there are compromised routers in the network, the attack paths reconstructed are reliable up to the nearest attacking routers. It improves the survivability of traceback, compared to marking schemes that rely on existing packet marks [5, 24].

5.2.2 Path Reconstruction

Path reconstruction finds the likely attack edges based on a collection of packet marks and connects the edges to reconstruct the attack paths. Packet marks are generated on attack and benign paths alike. Those from benign paths can be treated as noise in the reconstruction. RPM works on graph structured networks. Edge encoding facilitates a hop by hop reconstruction of the attack graph. It is assumed that the victim has an upstream router map. This is a reasonable assumption that is also made in [74, 93].

We now describe two methods. Both methods follow the hop-by-hop approach. They differ in the choice of an evaluation function that decides whether a given edge is a marker. The first evaluation function employs a Bloom filter, which is fast to compute but some information is discarded. Specifically, it only keeps track of the unique L-bit tags that the victim receives, but not the number of packets for each tag. The second evaluation function counts the number of occurrences per tag, and estimates the likelihood of an edge contributing to the tags received by the victim.

In a noiseless path reconstruction, the victim is assumed to be capable of differentiating malicious packets from benign ones. For example, in TCP SYN floods, malicious packets with spoofed addresses never completely establish a network connection; whereas benign packets abide by the network protocol and their sources are responsive. The victim can process only packet marks from non-responsive sources for path reconstruction. A Bloom filter [11] is used to store packet marks received from the attackers. With sufficient packets received, the victim performs a breadth-first search on the topology to reconstruct the attack paths edge by edge. It starts by testing if the edges composed of an immediate upstream router and itself belong to the attack graph. Recall that a router's marking behavior is fully determined by its IP address and the random identifier. The victim checks all neighboring edges, and computes for each edge the resulting packet marks. Benign edges are ruled out if not all packet marks encoding the edge are in the Bloom filter, else the edge is determined as residing on an attack path. The search continues at the identified edges until the obtained attack graph cannot be further extended.

In a noisy reconstruction, the victim does not pre-process the packets to discard benign packet marks. All packet marks are supplied to the path reconcentration procedure.

In the presence of noise, it is more appropriate for the victim to evaluate the likelihood of an attack edge based on the occurrence frequencies of its packet marks rather than simply based on their existence. A counting Bloom filter is used to store the occurrence frequency of each packet mark. The frequency of packet marks generated by an edge reflects the packet rate the edge experiences. If the likelihood exceeds some threshold, the edge is identified as attacking. Note that the threshold is distance dependent. Edges approaching the victim have gradually higher thresholds, because their packet marks experience lower probability of being overwritten and appear more frequently.

Equation 5.1 shows the function used to evaluate the relative likelihood of an attack edge,

$$f(e) = \frac{1}{h}\sum_{i=1}^{h} \#H_{e,i} \tag{5.1}$$

where $\#H_{e,i}$ measures the number of packets having mark v_i of marker e. This equation computes, for a marker, the algebraic mean of the occurrence frequencies of its packet marks. Because occurrence frequency of packet marks is proportional to the packet rate, and packet rate is the differentiation criteria between benign users and attackers, the value from Equation 5.1 gives the relative measure of confidence in determining attackers.

The pseudocodes below summarize the path reconstruction procedure.

RPM Reconstruction

```
Start at the victim's immediate edges
Compute $mean$(edge mark frequencies)
(Equation 5.1)
if mean $\geq$ threshold then
  Add this edge to the attack graph
  Move on to upstream edges
end if
```

It is assumed attackers are sending at much higher rate than non-attackers. Larger traffic rate thus provides higher confidence in identifying an attack edge. Assuming that the victim is interested in paths with packet rate greater than R_{att}, the packet mark threshold for edges at distance t from the victim can be computed as

$$\frac{1}{h}\ \epsilon(1-\epsilon)^{(t-1)}\ R_{att} + Z$$

where Z is an estimated noise level. This can be estimated using the average of a few randomly chosen sets of h tags.

RPM and AMS have similar reconstruction cost. Both schemes use topology information to perform a breadth-first search.

5.3 Evaluation

5.3.1 System Parameters

There are two parameters in the model we presented, ϵ and h. Both of them affect the identification accuracy. The number of tags h each marker should generate plays a very important role in determining a scheme's effectiveness.

Choice of ϵ

Assuming the marking probability ϵ is uniform across all markers, there are two factors affecting the choice of the ϵ value. One is that the mixture of mark distributions must retain portions of contribution from each upstream marker, as the mixture is the basis for identifying markers. Markers d hops away from the victim has a probability of $1-(1-\epsilon)^{d-1}$ of its packet marks being overwritten. This probability drops with decreasing ϵ; larger portions of upstream markers' contribution can be retained, especially for the farthest markers. The other factor for choosing ϵ concerns with the marking workload. Smaller ϵ means lighter workload for each marker. Therefore, small values are preferred for ϵ. Generally, the values between 0.03 to 0.06 provide a good tradeoff. If ϵ is too small, the victim is required to handle a large amount of traffic before it can gather a converged mixture of packet mark distributions.

Choice of h

In this section, we analyze the effect of different h on the false positives, when Bloom filter is used for reconstruction. Recall from Section 4.3.1 that high entropy tags facilitates high traceback accuracy. However, it is not always true that the higher the packet mark entropy, the better the traceback accuracy. Consider an extreme case where every marker holds 2^L distinct tags and uniformly randomly selects them to mark packets. Entropy of the collection of tags is close to the maximum, L, but the ability to differentiate markers is lost. Bloom filter has its fill factor increases quickly with markers, if each marker has

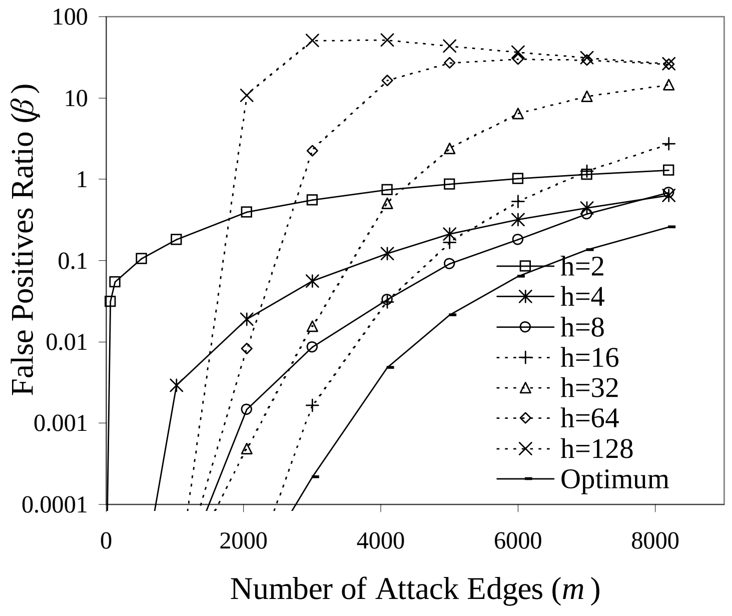


Figure 5.1: Effect of marking component lengths

many distinct tags; so is the false positive probability. Hence the choice of h is crucial to the scheme's effectiveness.

Consider a non-attack node i. It will be wrongly classified as an attack node if all of its associated tags are received by the victim. Let $\hat{\beta}$ be the probability that the node is wrongly classified. Since the reconstruction is carried out hop-by-hop, not all nodes in the network will be evaluated. Hence, it is not easy to relate $\hat{\beta}$ to the false positive of the overall scheme. Nevertheless, the analysis of $\hat{\beta}$ provides some insights on the choice of h. We do not consider false negative since the bloom filter will not miss any attacker.

We assume that the tags assigned to the markers are random. Since there are m attack nodes, and each node is assigned h tags, there are a total of mh tags chosen (there are likely to have repetitions within the mh tags). We can approximate this assignment as the random assignment of mh balls into 2^L boxes[1], where each box correspond to a tag. WLOG, let us assume that the first h boxes are assigned to the non-attack node i. Hence $\hat{\beta}$ can be approximated as the probability that all the first h boxes are filled, which can be approximated by

$$\hat{\beta} \approx \left(1 - (1 - \frac{1}{2^L})^{mh}\right)^h \approx \left(1 - e^{-\frac{mh}{2^L}}\right)^h \tag{5.2}$$

where e is the based of natural logarithm.

We perform the following simulation to analyze the role of h. The same edges data

[1]The two processes are not equivalent since each marker has exactly h distinct tags.

Table 5.1: Comparison in bit allocation of PPM Schemes

Scheme	#Bits (L)	Path Info (x) (ID Info, Dist)	Hash Index ($\log_2 h$)
FMS	16	$(8, 5)$	3
AMS	16	$(8, 5)$	3
FIT	16+5	$(13, 1 + 5)$	2
RPM	16	$(16, 0)$	0

set as in Section 4.3.1 is used. L is set to 16. Each edge generates 1,000 packets with h randomly chosen tags. For each h, 1 to over 8,000 edges are randomly chosen as malicious. The false positives ratio β is measured as the ratio of falsely accused edges to attack edges. Figure 5.1 illustrates how traceback accuracy varies with number of attack edges for different h. From Figure 5.1, when $m = 4,000$, $h = 8$ or 16 has the lowest false positive ratio. From Equation 5.2, β is minimized when $h = 12$. When $m = 8,000$, β attains minimum at $h = 5$. The optimal β, computed using Equations 5.2, achieved for various number of attackers is also shown in Figure 5.1. The simulation result largely agrees with the analytical model.

When $h = 2$, β is high even with relatively small number of attack edges. However, β increases slowly as the number of attack edges increases. From Equation 5.2, $h = 1$ is the optimal setting if there are about 20,000 attack edges, with β approaches 1.66. As h increases to 2^4, false positive tends to be lower for number of attack edges less than 4,000. Beyond 2^4, β increases quickly for large values of h. For number of attack edges from 4,000 to 8,000, $h = 2^3$ is an optimal configuration.

In general, the result shows that there is an optimal number of unique tags each marker should generate given the number of attackers. However, as the number of attackers cannot be predicted, h has to be set to a value that is appropriate for the maximum attackers anticipated.

Based on the analysis and simulation, it is now useful to compare the configuration of PPM schemes using our analysis on the choice of h. Table 5.1 summarizes the bit allocations for packet marks of several PPM schemes. These schemes are Fragment Marking Scheme(FMS) [69], Advanced Marking Scheme (AMS) [74], Fast Internet Traceback (FIT) [93] and our proposed scheme, Random Packet Marking (RPM). Randomize-and-Link (RnL) [35] and the algebraic encoding approach [24] both exploit more than 16 bits as L, and have flexible settings in lengths of hash index and path information (Section 4.3.2). Nevertheless, it is interesting to note that most of the schemes, including RnL and the algebraic encoding, use 2 or 3 bits for hash index, thus h is 4 or 8. For a large number of attack edges, $h = 2^2$ or 2^3 suffices. When $h = 2^3$, the schemes are optimized for 4,000 to 8,000 attack edges. For a smaller number of attack edges, $h = 2^4$ is a better choice.

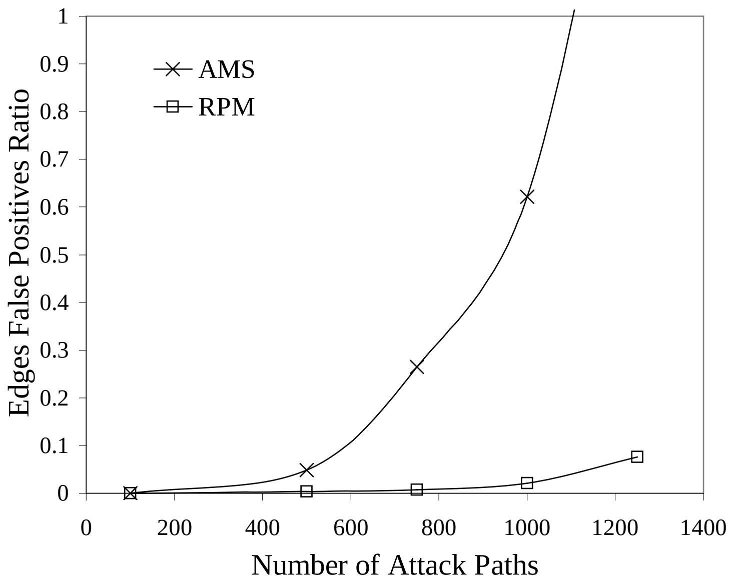


Figure 5.2: False positives of AMS and RPM (noiseless)

5.3.2 Performance

For evaluation, we compare RPM to AMS. AMS is selected as it is the most scalable among existing schemes that use 16-bit packet marks. The network topology used is drawn from the Internet Mapping Project [1] data captured on January 2006. The data set contains the route information from a source to 111,342 ($\sim 2^{16.8}$) destinations. There are 260,386 ($\sim 2^{18.0}$) unique edges or 209,582 ($\sim 2^{17.7}$) distinct nodes. The single source is used as the victim. From 100 to 1,250 nodes are randomly chosen as the sources of attack packets. They may reside anywhere in the network. The attack path packet rate is set to 1,000 packets per second, and the benign path packet rate is set to $\frac{1}{10}$ of an attack path. Each attack is simulated for 10 seconds. By then, the relative amount of different packet marks are stable, and the collection is enough to reconstruct the attack graph.

In the experiments, the parameters for RPM are $\epsilon = \frac{1}{16}$ and $h = 16$. The setting for AMS is as stated in [69], and referenced in Table 5.1. Simulations show that both AMS and RPM have negligible false negatives, hence their α lines are omitted from the figures. Their performances are compared only in terms of β, the false positives ratio in identifying edges.

Figure 5.2 shows how β varies with number of attack paths in the noiseless case where only packets from attackers are considered. It can be clearly observed that AMS has exponentially increasing β with the increasing number of attack paths. When there are 1,000 attack paths, or roughly 4,650 attack edges, RPM and AMS have β values of 0.02 and 0.63 respectively.

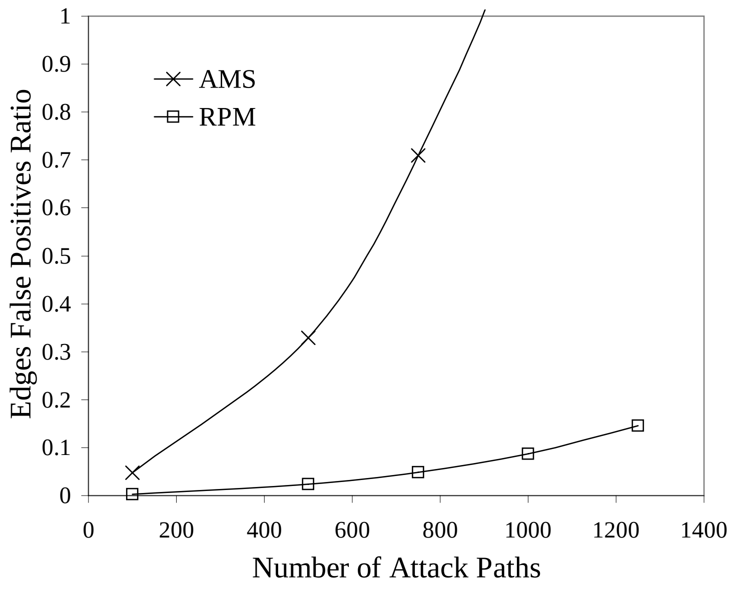


Figure 5.3: False positives of AMS and RPM (noisy)

Figure 5.3 shows the case where packet marks from both user and attacker paths are supplied to the path reconstruction procedure. 200 user paths and varying number of attack paths are simulated. The result is similar, though both AMS and RPM have higher false positives. With 1,000 attack paths, RPM and AMS have β values of 0.09 and 1.28 respectively.

The improvement can be explained by comparing the packet marks entropy shown in Figure 4.3. It shows that the packet entropy of RPM is very close to that of RnL, and is much higher than AMS.

Figure 5.4 shows the number of routers falsely identified at each distance. The number of attack nodes at each distance is also shown in the figure as a reference. In the simulation, there are 1,000 attack paths and 200 user paths. At distances of 9-14 hops away, where there are many routers, AMS generates many false positives. This is because AMS cannot resolve packet mark collisions of routers at the same distance. On the other hand, RPM has small amount of false positives at all distances.

Finally, Figure 5.5 shows β for RPM under different scenarios. The 'RPM noisy (200 users)' case maintains 200 normal users, varying the number of attack paths from 100 to 1,250. In the figure, the x-axis represents the number of attack paths for this case. The 'RPM noisy (200 attackers)' case keeps a constant number of 200 attack paths, but varies the number of users from 100 to 1,250. The x-axis denotes the number of users in this case. It can be clearly seen that even in the presence of noise, RPM still outperforms AMS (noiseless) by a significant amount.

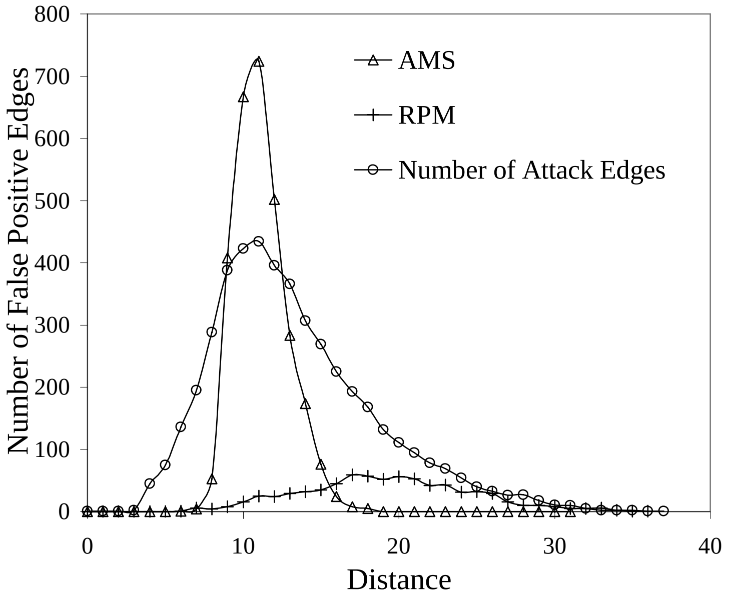


Figure 5.4: False positives of AMS and RPM by distance

5.3.3 Gossib Attack and RPM's Survivability

Gossib (Groups of Strongly SImilar Birthdays) attack [85] was proposed by Waldvogel. It can be used to obtain effects similar to a birthday attack on PPM traceback schemes. Gossib increases the state space for the victim to search, by randomly inserting edge fragments into the packet marks. Simultaneously, it inserts misleading edges into the attack graph. In addition, it optimizes the number of packets needed to fake the edges or edge fragments.

Gossib attack appears severe to PPM traceback, for a diligent attacker can forge as many packet marks as the amount of traffic it transmits, in comparison to a PPM marker who performs the marking routine at around 1/20 of the time.

However, routers can co-operate to proactively identify and filter forged packet marks, using an approach inspired by route-based distributed packet filtering (DPF) [63]. Park and Lee proposed DPF as a defense against address spoofing. A gateway router verifies the incoming addresses with respect to the topology. When a source address is invalid against an interface, the gateway filters the packet. A number of spoofed packets can survive even with the gateway's filtering check, which is when the spoofed addresses indeed are expected from the particular interface, but are either behind or in front of the attacker. DPF limits the number of allowed address spoofing. The paper claims with 20% of all gateway routers performing the check, the effect of address spoofing becomes contained.

For RPM to defend against the Gossib attack, routers at strategic points are forced

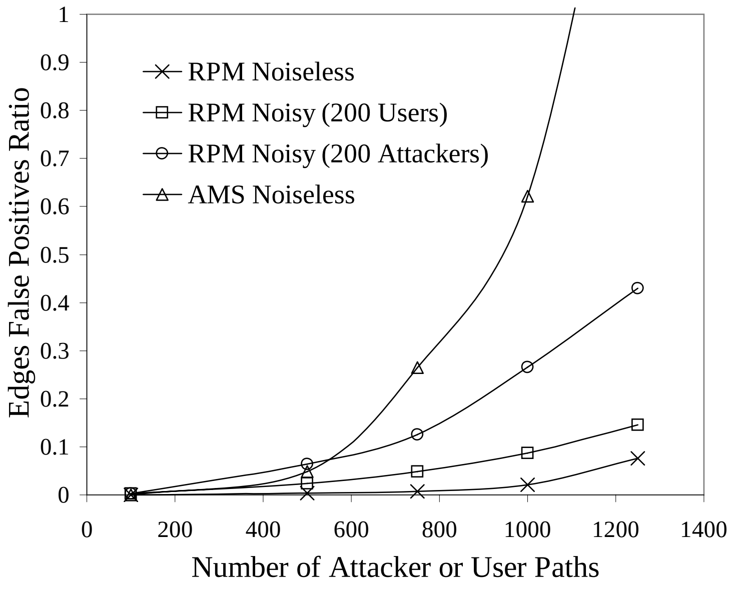


Figure 5.5: False positives of RPM

to validate if the packet marks are expected to their interfaces. When a mismatch occurs, the packet marks can be safely eliminated, by restoring them to all 0's. The strategic points can be based on topology and distance between routers.

For example, a checking router can be placed at about 3 to 5 hops away from another checker. So each checker keeps track of all the possible packet marks of its upstream markers within 3 to 5 hops. Using the same bit-allocation of packet marks as in Section 5.3.2, each checker keeps about 16 packet marks for each nearby upstream router. The coverage effect by 20% routers can be achieved with this simple proposal, and the result of DPF can apply.

By placing checkers at strategic points, the bogus packet marks can be eliminated early. More importantly, the number of allowed packet marks each checker needs to keep track of can be minimized. Hence strengthening the differentiability of packet mark validity. The exact placement of the checkers are topology dependent. In general, the rule of thumb is to place them before the bottleneck links, where multiple branches in the network graph merge.

As defending against the Gossib attack is not the focus of our paper, we do not reproduce the evaluations similar to the DPF research. It is sufficient to note that it is possible for RPM (and PPM traceback schemes in general) to survive diligent Gossib attackers.

5.4 Summary

We present a PPM scheme called RPM that has good traceback accuracy and efficient path reconstruction. Simulations show improved scalability and traceback accuracy over prior works. For example, a thousand attack paths induce 63% of false positives in terms of edges identification by using AMS. RPM lowers the false positives to 2%. The effectiveness of RPM demonstrated that imposing sophisticated structures on tags is not necessary. If imposing the structure reduces the randomness in tags, it should be avoided. The improvement of RPM over prior schemes is mainly the result of increasing the information carried by packet marks.

Chapter 6

Website Fingerprinting over VPN

We consider a traffic model of encrypted and proxied web browsing traffic, such as through VPN, SSH and SSL/TLS encrypted tunnels. We passively analyze the web traffic to extract side channel information to identify "fingerprints" of websites. We develop a scheme that introduces packet ordering information into website fingerprints in addition to the commonly used packet sizes.

6.1 Overview

Website fingerprinting aims to identify the website accessed in some low latency, encrypted tunnels. Website "fingerprints" are profiled on side channel features observed from the HTTP streams. By HTTP stream, we mean the stream of packets sent and received when web pages are accessed. From the perspective of web clients, website fingerprinting raises the privacy concern and indicates the need for further anonymization. On the other hand, such techniques aid the legitimate warden to track web accesses over encrypted channel.

Existing website fingerprinting schemes naturally fingerprint websites using packet size related features [40, 78, 50], since neither the IP address nor domain name of the website is available due to encryption and proxies. Due to network delay variations and the HTTP pipelining mechanism adopted by modern browsers, the ordering of packets varies in different accesses to the same website, previous works [40, 78, 50] have chosen to discard the information on packet ordering. For example, Liberatore and Levine's scheme [50] represents each website fingerprint as a set of packet sizes and uses a variant of set-difference to measure the similarity of two fingerprints.

In this chapter, we propose a website fingerprinting scheme that exploits packet ordering information, as well as information on packet sizes. We argue that although packets can be significantly reordered by HTTP pipelining and networks effects, there is sufficient residual information in packet ordering to facilitate fingerprinting, and thus improving the website identification accuracy.

Our scheme is evaluated on a classification problem that identifies an HTTP stream among a list of profiled websites, as well as a detection problem in which it is uncertain whether the test website belongs to the profiled set.

We tested our scheme on datasets containing up to 2,000 websites, with traces collected over two months. For the classification problem, we test our scheme with full length OpenVPN test streams, i.e., 30-second long encrypted HTTP streams tunneled through VPN proxy server, and the website identification accuracy is 97% among 1,000 websites. We also test with partial SSH traces that capture only the first few seconds of encrypted communication, the accuracy is 81% among 2,000 websites, still over 10% better than the previous scheme. For the detection problem, our scheme presents an equal error rate at 7%, which is significantly better than the previous scheme at 20%. Hence, for both types of evaluations, our scheme yields superior identification accuracy than previous schemes.

In addition, we analyze the consistency of our website fingerprints, with respect to static and dynamic websites, and different HTTP pipelining configurations. We find that only 6% of the websites need reprofiling after a month. The results verify that our feature selection generates consistent fingerprints.

6.2 Traffic Analysis Model

We use passive traffic analysis to fingerprint websites. There are two important observations that enable us to adopt this model. The observations are: (*i*) webpage download by HTTP is highly structured; and (*ii*) encryption and proxy do not severely alter the packet sizes, nor the packet ordering. The advantage of passive traffic analysis is that the presence of a warden is completely transparent.

The HTTP streams for analysis are captured over the protected tunnel between client and the VPN or SSH server, as shown in Figure 6.1. As in previous works [40, 78, 50, 10], we assume browser caching is disabled. Website fingerprints are extracted from the HTTP streams. Several fingerprint instances form the profile of a website. A testing stream is compared to each profile for identification.

Traffic Model

Our fingerprinting model supports most browser configurations and the following connection patterns are accommodated (*i*) multiple TCP connections can be opened to download in parallel different embedded objects, (*ii*) each TCP connection can be reused to download multiple objects, (*iii*) HTTP requests can be pipelined in each connection, and (*iv*)

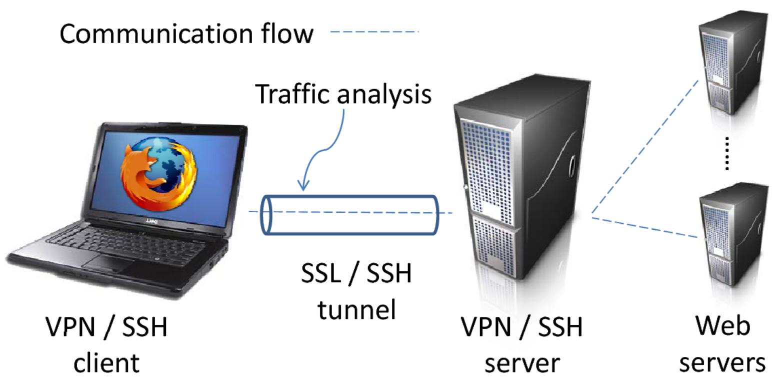


Figure 6.1: Illustration of traffic analysis setup

all sessions are multiplexed onto a single port.

The connection patterns speed up the webpage download, but they increase the difficulty to fingerprint websites. HTTP pipelining means a client is allowed to make multiple requests without waiting for each response. Because of multiple TCP connections and HTTP pipelining, data packets of different embedded objects can interleave, thus object sizes cannot be determined by accumulating the amount of data received between consecutive HTTP requests. Multiplexing communication sessions hides the number of TCP connections opened and the number of objects in a webpage.

Problem Scenarios

We tackle two problem scenarios, classification and detection. In both scenarios, a set D of, say 1000, websites are profiled.

- **Classification** Given a test stream which is known to be a visit to a website in D, identify the website. This is the same problem addressed in previous work.
- **Detection** Given a test stream, determine whether it is a visit to a website in D, and identify the website if it is. We examine the false positive rate (FPR) and false negative rate (FNR) in identification. This problem has not been addressed in previous work.

The classification scenario requires the fingerprints of a website be sufficiently similar. While the detection scenario further requires fingerprints of different websites be sufficiently dissimilar.

Noise Model

There are a few sources of noises that degrade the consistency of HTTP streams, even if the website contents have not changed. Connection multiplexing and browser configuration in HTTP pipelining are the main sources affecting the ordering of objects. In addition, the object request order may be browser specific, and the dynamics in network condition causes some random noise.

6.3 Website Fingerprinting Scheme

The fingerprint profile of a website is built by accessing the website through the encrypted tunnel and extracting the fingerprints from captured HTTP streams. Testing streams are captured in a similar way, from which fingerprints are extracted and compared to the fingerprint profiles for identification. We present the details on fingerprint feature selection and extraction, and fingerprint similarity comparison in this section.

6.3.1 Fingerprint Feature Selection

A straightforward approach to represent the fingerprint of an HTTP stream is by using the sequence of all packet sizes and directions, i.e. fingerprint $F = \langle(s_1, d_1), (s_2, d_2), ...\rangle$ where s_i is i-th packet size, and d_i is its direction. Clearly, the approach makes fingerprint comparison inefficient, since each sequence easily contains over a thousand elements. Yet more importantly, the identification accuracy will be affected, because the ordering of some packets often changes due to various noises. Hence, we apply domain knowledge to select features.

Firstly, packets with sizes smaller than a threshold are considered control packets and discarded from fingerprints, as control packets do not represent the webpage content and potentially vary across traces.

Secondly, we keep only the non-MTU (Maximum Transmission Unit) downloading packets as features of HTTP responses. The reasons why we exclude the MTU downloading packets are explained below.

Ideally, we would like to monitor object sizes rather than packet sizes, as inline objects are less variable and more characteristic to a website. However, data of different objects can interleave in transmission due to multiple TCP connections, HTTP pipelining and connection multiplexing. It is difficult to associate the data blocks to their respective objects. Fortunately, HTTP transmission is not random. Majority of web servers transfer data in chunks. In each data chunk, all packets are sized as path MTU, except the last packet transferring the remaining data. Therefore, although we cannot estimate object sizes, we can leverage the last packet size of a data chunk, as it is specific to an object component. As for packet ordering, packets that change their orders tend to be the intermediate packets of different objects. By filtering the intermediate packets, we reduce the probability of fingerprint inconsistency.

Thirdly, we design the fingerprint of an HTTP stream to be two sequences, representing the request and response features respectively. Although a request must precede its corresponding response, the order of other requests to this response is not deterministic, but sensitive to the pipelining configuration. Hence the request and response features are not merged into a single sequence.

Therefore, to fingerprint a website over encrypted tunnels, the side channel features we select are (*i*) **sequence of HTTP request sizes** and (*ii*) **sequence of HTTP response sizes** (except MTU packets). Note that *the number of objects* and *the number of components in object responses* are two implicit features represented by the lengths of these two sequences.

The features we select closely adhere to the webpage content and layout. The request sequence reveals the relative locations of the embedded objects in a webpage and their URL lengths. The sequence is consistent since webpages tend to have stable layout, and browsers issue the object requests in the order they parse the object references. The response sequence indicates the download completion order of the object components and their last packet sizes. The sequence is stable since web servers serve requests in the order they are received, and the response chunks are ordered by the object components. Hence our website fingerprints are very consistent across traces, if network dynamics are disregarded and the browser configuration is fixed.

We identify the HTTP requests and responses by packet sizes. As client sends packets to the server either as acknowledgements or to make requests, any packet in the uploading direction (from client to server) with size greater than a threshold is considered a request. For the downloading traffic, we can broadly classify the packets into three types, which we call 0-size packets, MTU-size packets and non-MTU-size packets. The *0-size packets* carry no object data, but are control packets added by OpenVPN, HTTP or TCP for communication management. The *MTU-size packets* transmit the continuous object data, with sizes close to the path MTU. The rest packets belong to the non-MTU-size category. Since TLS requires the sender to flush each record [41], we use non-MTU-size packet as the signal for completion of a message chunk, and hence as a signature for an HTTP response.

Note that DNS queries and replies are encapsulated and encrypted into TCP packets. They may contribute to the fingerprint sequences depending on their packet lengths after encryption. In any case, the order and sizes of the DNS queries and replies are consistent since they occur when new TCP connections are about to be established for object retrieval from a third party server, and their order and lengths represent the domain name length of the referenced server and the approximate location of the reference in the webpage.

Estimate of Number of Unique Fingerprints

Based on the possible number of feature values and ordering, we can estimate the number of unique fingerprints and hence the potential number of distinguishable websites. From the dataset containing the fingerprints of the homepages of about 1,000 popular websites, the average number of object requests and response components are 66.5 and 135.9 respectively. Since each non-MTU-size packet can take over 1,000 possible values and repetitions are allowed, the number of unique fingerprints is estimated to be over $1,000^{202.4} \approx 2^{2024}$. This number is an underestimation because it assumes website fingerprints are of the same length. Yet not all websites can be uniquely identified. For pure HTML webpages without embedded objects, approximately 1M unique fingerprints are available, but the number of such webpages is larger. Fortunately, popular websites of interest for identification usually have large number embedded objects, e.g. graphics and advertisements. Websites that have relatively long fingerprint sequences naturally have higher probability of being uniquely identifiable.

6.3.2 Fingerprint Similarity Measurement

Edit distance measures the number of edit operations to make two strings identical. *Levenshtein distance* is a form of edit distance that allows insertion, deletion or substitution of a single character. We use Levenshtein distance to measure the similarity between two fingerprints. It is commonly computed using the Wagner-Fischer algorithm [84].

In our application, we set the cost of insertion, deletion and substitution all equal to 1. If network dynamics are disregarded, each operation reflects a modification to an embedded object, and we do not consider any operation changes the webpage more.

We normalize the edit distance by $\max(|x|, |y|)$, where $|x|$ denotes the length of sequence x. We define $similarity = 1 - distance$, to convert the distance measurement into similarity. Given two fingerprints, two similarity values measured on the object request sequences and on the non-MTU response sequences are combined as follows:

$$\alpha \cdot sim_{HTTPget} + (1 - \alpha) \cdot sim_{nonMTUpkts}$$

where α is a tunable parameter, indicating the degree of reliability of the similarity values. We set $\alpha = 0.6$ in our experiments, as the sequence of object requests is more stable. The probability of a test stream coming from a particular website is determined as the maximum similarity between the fingerprint of this stream and the various fingerprints in the website profile.

Edit distance is chosen as our similarity measure because (i) fingerprint instances vary as a result of network dynamics and webpage updates, whose effects are shown as insertion, deletion or substitution in the sequences, (ii) it considers the order information, which differs from Jaccard's coefficient that treats the feature values as a set, and (*iii*) the length information of fingerprint sequences are encapsulated for evaluation.

Example

Given two fingerprints $F_A = (A_{req}, A_{res})$ and $F_B = (B_{req}, B_{res})$, where $A_{req} = \langle 436, 516, 500, 532\rangle$, $A_{res} = \langle 548, 628, 1348, 1188, 596, 372, 980\rangle$, $B_{req} = \langle 436, 516, 500, 253, 500\rangle$ and $B_{res} = \langle 548, 628, 1348, 1188, 436, 412, 1268\rangle$.

To make the request sequences A_{req} of F_A and B_{req} of F_B identical, we need two edit operations: delete the last 500 from the B_{req}, and change 253 in B_{req} to 532. Thus the edit distance between the request sequences A_{req} and B_{req} is 2. We then normalize it by dividing it by the longer length of the two sequences, the edit distance is 2/5 after normalization. So the similarity between A_{req} and B_{req} is $1 - 2/5 = 3/5$. Similarly, we can compute the similarity between the two response sequences A_{res} and B_{res}, which evaluates to 4/7. Hence when $\alpha = 0.6$, the overall similarity between F_A and F_B is $0.6 \times 3/5 + (1 - 0.6) \times 4/7 = 0.59$.

6.4 Evaluation

In this section, we present the performance and analysis of our fingerprinting scheme. The experiment settings and data collections are described in Section 6.4.1. We evaluate the performance of our scheme in both classification and detection scenarios and present the identification accuracies in Section 6.4.2. The consistency of our fingerprints with respect to HTTP pipelining configurations, number of profiling samples and website updates is analyzed in Section 6.4.3.

6.4.1 Experiment Setup and Data Collection

We prepared three datasets on OpenVPN for evaluation, namely, *static50, dynamic150*, and *OpenVPN1000*. The dataset *static50* contains the trace data of 50 websites that are rarely updated (less than once in a few months); the dataset *dynamic150* contains the traces of 150 websites that are frequently update (daily); and the dataset *OpenVPN1000* contains the traces of 1000 popular websites.

To demonstrate that our scheme also works on SSH, we use the publicly available dataset *OpenSSH2000* [50], which contains the traces of 2000 most visited websites accessed through OpenSSH, captured once every 6 hours for 2 months. This trace captures only packet headers in the first 6 seconds of a webpage download.

We prepare the datasets of OpenVPN by visiting the websites[1] and capturing the

[1]Currently we identify a website by fingerprinting its homepage. If the internal pages of a website are

packet traces daily for 3 months. The traces are captured using TCPDump [79] between the client and the VPN server. We used Firefox for the capture, HTTP pipelining is enabled and its default value is 4 (at most 4 HTTP requests can be served concurrently). Cache is cleared after each access. Flash player is installed, since many websites use flash in their homepages. Each web access is monitored for 30 seconds, to allow the operations of object request and response to fully complete.

The topology, size and content of high level cache is not within our control for evaluation. In our experiments, the profiled websites are accessed regularly, so we expect most objects are consistently cached close to the VPN or SSH server.

We use Liberatore and Levine's scheme [50] for performance comparison, and refer to it as the *reference scheme*. Our scheme that performs preprocessing is referred to as the *improved scheme*, otherwise it is called the *basic scheme* or simply our scheme. For each website, 15 traces are used to generate the fingerprint profiles, and 10 other traces are used for testing. The fingerprint sequence of object requests are composed of packet sizes that are larger than 300 bytes in the uploading direction. The fingerprint sequence of non-MTU responses consists of packet sizes between 300 to 1,450 bytes in the downloading direction. These settings are used in our experiments, unless otherwise specified.

6.4.2 Fingerprint Identification Accuracy

In the following, we show the fingerprint identification accuracies for classification and detection scenarios. As described in Section 6.2, a test stream is known to be from the profile websites in the classification scenario, but it is unknown whether a test stream is from the profile websites in the detection scenario.

Accuracy of Classification Scenario

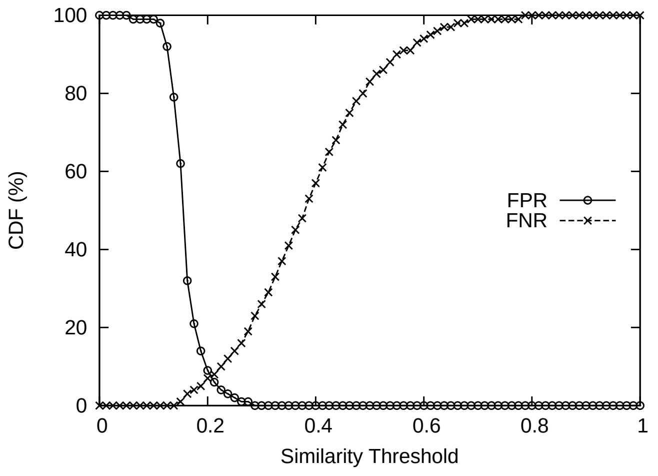
The fingerprint identification accuracies on the OpenVPN datasets *static50*, *dynamic150* and *OpenVPN1000* are shown in Table 6.1. Note that for all three datasets, the accuracy of our scheme is close to 100%. In comparison, the reference scheme yields an identification accuracy of 90% when it is run on *OpenVPN1000*.

Dataset	Static50	Dynamic150	OpenVPN1000
Accuracy	99%	97%	97%

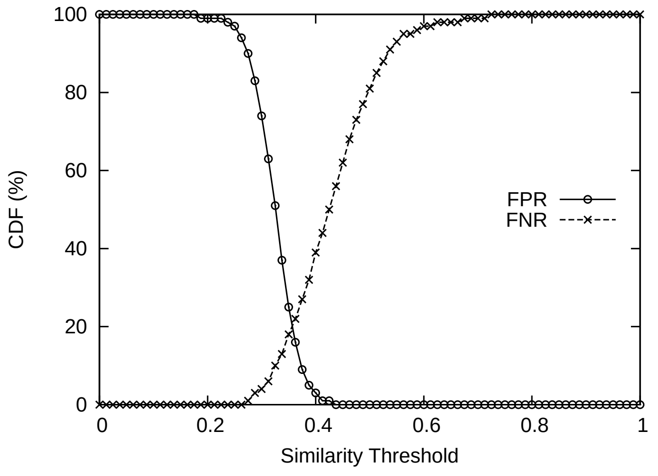
Table 6.1: Fingerprint identification accuracy for various datasets

To demonstrate that our scheme works well for identifying websites accessed over other tunnel, we tested it on the dataset *OpenSSH2000*, with parameters fully complying with

also profiled, we can use the fingerprints of individual pages as well as their linkage information to identify more accurately and confidently the website being surfed.



(a) Our scheme



(b) Reference scheme

Figure 6.2: False positive and false negative rates with respect to similarity thresholds

the settings in [50]. The accuracy of our scheme reaches 81%, outperforming the reference scheme by 11%. The improvement is due to the fact that our scheme is able to identify websites that have similar sets of packet sizes but different packet orders.

Accuracy of Detection Scenario

It is unknown in this scenario whether a test stream is from the profiled websites. Thus a test stream not from the profiled websites may be identified wrongly as from one of the website in the profile (false positive); and a test from the profiled websites may be identified wrongly as not from the profiled websites (false negative). We evaluate the fingerprint identification accuracy in terms of false positive rate (FPR) and false negative rate (FNR) for the detection scenario.

Profile	Accuracy
Multiple pipeline degree	99.7%
Single pipeline degree (pipeline degree = 4)	99.3%

Table 6.2: Fingerprint identification accuracy with respect to different pipeline configurations

From the dataset *OpenVPN1000*, we randomly pick 200 websites to profile. Traces from both these 200 websites and the remaining 800 are used to test the fingerprint identification. The accuracy of our method is shown in Figure 6.2a, and performance of the reference scheme is shown in Figure 6.2b.

Comparing Figure 6.2a with Figure 6.2b, the FPR and FNR curves for our scheme are more separated from each other than that for the reference scheme. The equal error rate (EER) for our scheme occurs when the similarity threshold is 0.21, at which both FPR and FNR are 7%. While for the reference scheme, at the similarity threshold of 0.36, EER occurs to be 20%, almost three times of ours. If we minimize the total error rate, i.e. sum of FPR and FNR, the optimal similarity threshold for our method will be 0.22, with only 6% FPR and 8% FNR. While for the reference scheme, the optimal similarity threshold will be 0.37, at which its FPR and FNR are 9% and 27% respectively. The results show that our scheme is much stronger for differentiating websites.

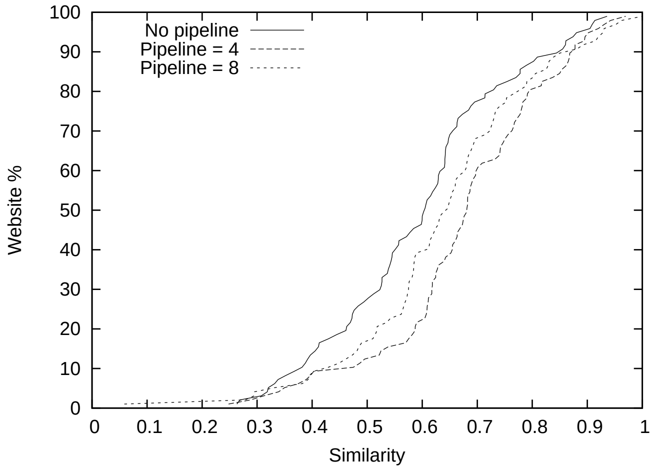
6.4.3 Consistency of Fingerprints

Website fingerprints can be influenced by several factors even if the webpage contents have not changed. These factors include browser pipeline configuration (Section 6.4.3) and network dynamics (Section 6.4.3) that cause packet loss, reordering and retransmission. Note that the randomness in the amount of control packets does not affect our fingerprint at all. Through experiments and analysis in this section, we show that these factors do not seriously affect the consistency of our fingerprints.

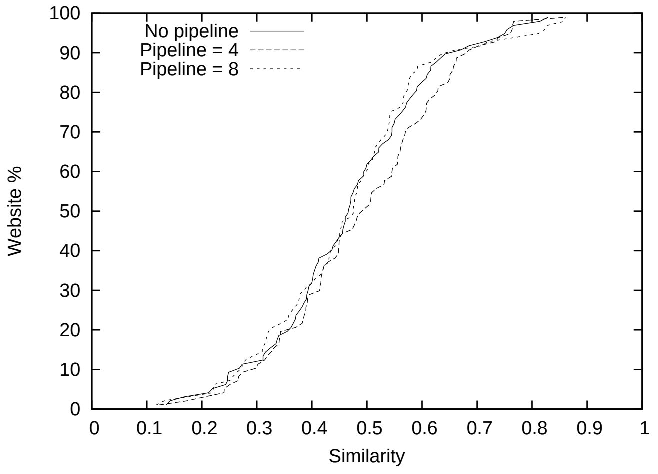
Effect of HTTP Pipelining

To illustrate the effect of HTTP pipelining on the fingerprint sequences, we vary the pipeline degree from 1 (non-pipelined) to 8 (the maximum supported by Firefox as to date). We capture OpenVPN traces of 100 websites. For each website, fingerprints derived from 15 traces with varying pipeline degrees are kept as profiles, and 15 test streams per website of varying pipeline degrees are compared with their own website profiles.

Figure 6.3 shows that the fingerprint similarities of a website is high. 70% of the websites have similarity in the object request sequences higher than 0.5, and 50% of the non-MTU-size response packet sequences have similarity larger than 0.5. It is also shown



(a) Sequence of HTTP requests



(b) Sequence of non-MTU responses

Figure 6.3: Effect of pipelining on the fingerprint sequences

the similarities of a website's fingerprints have similar values at different pipeline degrees. Test streams of a pipeline degree of 4 is most similar to their profiles, because 4 is the median of pipeline degrees in each profile. Because the object request and response order is tightly associated with the webpage layout and object sizes, HTTP pipelining does not dramatically change the fingerprint sequences.

To demonstrate that websites can be profiled using a single pipeline degree, we performed an experiment in which the profiled fingerprints are of a single pipeline degree of 4, in comparison to the case where the profiles are composed of fingerprints of various pipeline degrees, while the test streams are always of various pipeline settings. 100 websites are profiled, each with 15 traces, and 15 other traces are used as test streams. The identification accuracies are shown in Table 6.2. With either profile, the identification

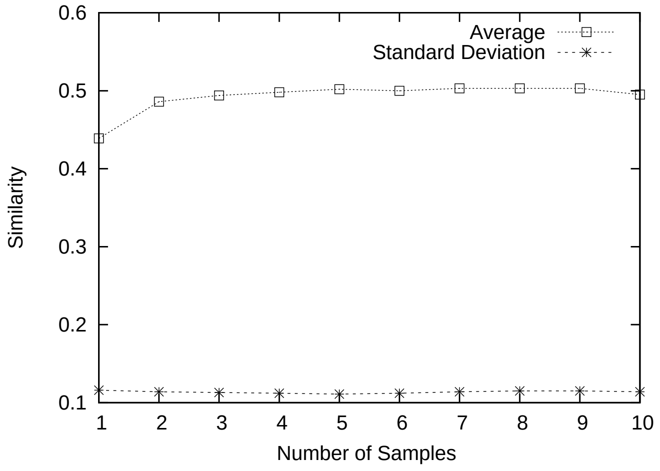


Figure 6.4: Effect of the number of learning samples

accuracy exceeds 99%. The results show that our fingerprints are not sensitive to changes in the pipeline settings. We can identify websites with high accuracy even when they are profiled in a single pipeline degree but tested with various pipeline degrees.

Number of Profiling Samples per Website

The number of samples needed to profile a website is related to the stability of the network and the website properties such as variations in the dynamic contents. Yet these factors are connection and website specific and may evolve over time. In order to isolate the effect of network dynamics from webpage updates, we perform the experiment on the websites that update infrequently, i.e., using the dataset *static50*. The number of profiling traces varies from 1 to 10, and 5 test streams per website are used to test the similarity to their own website profile. Figure 6.4 shows the average and standard deviation of similarity of all websites versus the number of profiling samples. The figure shows that under normal network condition, we only need a couple of traces to profile a static website. Increasing the number of samples further does not increase the similarity on average. Although network dynamics affects the object download completion order, the experiment shows that our fingerprints are not seriously affected by it. The reason is that we only monitor the last packets in message chunks instead of all data packets, so the probability of the monitored packets being affected is reduced.

Effect of Website Update

To evaluate how well the profiles represent websites over time, we measure the fingerprint identification accuracy weeks after constructing the profiles. The evaluation helps to decide the update frequency of website profiles. We perform an experiment on the datasets

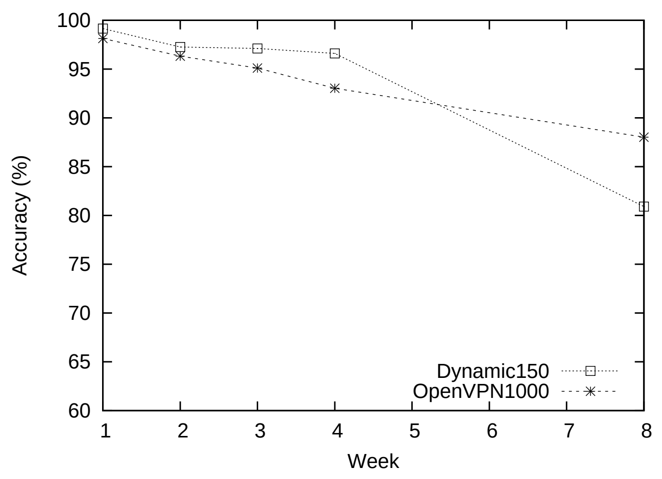


Figure 6.5: Effect of time on accuracy

OpenVPN1000 and *dynamic150* to study the impact of (frequent) content updates on the fingerprint identification. For each website, we randomly pick 5 traces captured within week 0 to generate the website profile, then the traces of 4 randomly chosen days in each of week 1, 2, 3, 4 and 8, are tested against the website profiles.

The results are shown in Figure 6.5. For dataset *OpenVPN1000*, the identification accuracy gradually decreases over time, which remains high at 93% a month later and becomes 88% after two months. For majority of the websites, their profiles need not be regenerated, only about 6% of the most frequently updated websites need to change their fingerprint profiles in the first month, and another 6% in the second month. For the dataset *dynamic150*, the decrease in identification accuracy is more obvious over the two-month period, because their website contents are frequently updated, and the changes accumulate over time. Nevertheless, the fingerprint identification accuracy is 96% one month later, and 81% two months later.

6.4.4 Computation Efficiency

The time taken to identify a website varies with the fingeprint length. It is on average about 5 seconds (measured on a PC with 2.53GHz CPU and 2GB of RAM) when identifying the fingerprint among 1,000 website profiles each containing 15 fingerprints. The evaluation can be sped up with parallel computation on multi-core microprocessor. The storage required is very small, and the website profiles containing 15,000 fingerprints (written in ascii format) only take 12.4 MB. The computation cost of our scheme is low, and the storage overhead is small, so our approach is scalable.

6.5 Discussions

We discuss several issues concerning the practicality of website fingerprinting.

Associating packets to HTTP streams.

Real life Internet traffic are more complex than what we analyze in the model. One can use Skype, IM concurrently with web browsing, and can open multiple websites at the same time. Previous works [91, 12] have shown that application layer protocols can be fairly reliably identified in encrypted traffic. Previous work has also experimented with using time gap to separate traffic of webpages from tabbed browsing. It is based on the assumption that the victim requests webpages sequentially due to think times, and hence there are no overlapping transmissions. Yet the result shows it may be difficult to accurately associate network packets to different webpages. We plan to derive new methods in future work to extract individual web sessions from network traces.

Browser caching.

Some privacy guidelines state that browser caching should be disabled. Most internet banking sites always remind users to clear off their browser cache after each Internet banking session. If browser cache is enabled, one can determine if a user has accessed some sensitive website using the attack in [27]. The victim is issued requests on some objects of the sensitive website. From the response time, we can determine if the objects are cached and hence whether the user has visited the website.

Browser cache is often enabled in reality for efficiency. However, the experiment by Herrmann *et al.* [39] on their packet size frequency based website fingerprinting shows that the performance of their scheme is only moderately affected if browser cache is enabled. Though the underlying reason is not thoroughly analyzed, it suggests that caching may not be affecting website fingerprinting as much as one might think, even when the cache size is very large (2GB) such that practically cached contents are nearly never deleted.

Information from internal pages and user interaction.

Internal pages of a website and user interaction provide additional sources of information to website identification. If internal pages of a website are profiled, we can use fingerprints of individual pages as well as their linkage information to identify more accurately and confidently the website being surfed [16].

From encrypted Internet traffic, HMM can be used to classify application layer protocols, e.g. IM, web browsing, P2P sharing. Our work focuses on the web browsing traffic, to identify the website being accessed. Assuming it is known which web application is being used, one can model the series of interactive requests and the finite number of possible responses as state transitions, to infer the choices a user indicates [15]. User interactions are very dynamic in general, but careful modeling and examination will help to not only identify the website, but potentially reveal more targeted sensitive information, e.g. operations performed in Internet banking. We believe these mechanisms complement one another. We also note that the encrypted packets associated with internal pages or user interactions are more difficult to isolate compared to identifying traffic for the homepage.

6.6 Summary

In this chapter, we developed a robust website fingerprinting scheme over low latency encrypted tunnels. We make use of the seemingly noisy packet ordering information, rather than just the distribution of packet sizes as in previous work. The ordering of a selection of packets is consistent due to the behaviorial characteristics of protocols and browsers, and such information is preserved and exposed regardless of proxy and encryption.

We tested our scheme extensively on datasets containing up to 2,000 websites, with traces collected over two months. Our scheme identifies websites with high accuracy on both SSH and SSL tunnels, even for websites with dynamic contents. We moved the evaluation model beyond closed world identification (classification problem), such that a testing stream may be outside the set of profiled websites (detection problem). For both classification and detection problems, our scheme yields superior identification accuracy than the previous scheme.

We also analyzed the consistency of our website fingerprints, with respect to static and dynamic websites, and different HTTP pipelining configurations. The results verify that our feature selection generates consistent fingerprints.

Chapter 7

Resistance of Website Fingerprinting to Traffic Morphing

We investigate the robustness of our scheme proposed in the last chapter against traffic morphing, which is a bandwidth efficient countermeasure against website fingerprinting that changes the packet size distribution of the source web traffic to imitate another website.

7.1 Overview

We proposed in last chapter a website fingerprinting scheme that takes advantage of packet ordering information. The scheme not only improves the fingerprint identification accuracy upon previous results, but also withstands traffic morphing that minimizes the bandwidth overhead while transforming the packet size distribution to defend against website fingerprinting. We show in this chapter that our website fingerprinting scheme can differentiate between morphed traffic and its mimicked target, and that source websites can be identified with high accuracy even with morphing.

Simple defenses to website fingerprinting have been proposed in previous works, including variations of packet padding, e.g. maximum padding and mice-elephant packet padding. Maximum padding pads every packet to the size of path MTU. It can thwart size related traffic analysis, but the amount of overhead it causes is nearly 150% of the actual data [50]. Mice-elephant is less aggressive. It pads packets to two sizes, either to a small size for control packets, or to the path MTU for data packets. Thus the information revealed is limited to the approximate total size of object requests and the total size of webpage contents, including embedded objects. Yet the bandwidth overhead it causes is 50% of the data size. The large bandwidth overhead they cost leads to insufficient incentives for deployment.

As reviewed in Section 2.5, "traffic morphing" [90] is designed as a scheme to evade detection by warden, and at the same time, incurs small bandwidth overhead. Traffic morphing alters the packet size distribution of an HTTP stream, to make it appear as if it is from another website. A morphing matrix is pre-computed that transforms the packet size distributions and minimizes the total increase in packet sizes. The morphing technique targets at website fingerprinting schemes that utilize size related features but not the ordering information. Unigram traffic morphing changes the distribution of single packet size; bigram traffic morphing changes the distribution of sizes of two consecutive packets. Limited or no ordering information is considered in bigram or unigram morphing.

The website fingerprinting scheme we propose is able to withstand traffic morphing that is constrained on bandwidth overhead minimization. Although traffic morphing is effective in misleading or confusing the website identification by fingerprinting schemes that exploit only size related features, our scheme is robust to the traffic morphing technique, as we are able to differentiate among websites that have similar packet size distributions but dissimilar packet ordering.

The limitation of traffic morphing lies in that it cannot simultaneously satisfy low bandwidth overhead and preserve packet ordering. The reason is that in practice, for a given website, there are very limited choices for the size of the n-th packet once the sizes of the previous $n-1$ packets are fixed. We point out that if the traffic morphing scheme is extended to n-gram ($n \geq 2$) morphing (morphing the distribution of sizes of n consecutive packets), its bandwidth efficiency will become worse than some simple packet padding scheme.

Experiment results verify that our scheme is resilient to the traffic morphing technique [90]. When traffic is morphed according to unigram (i.e. single packet size) distribution, our scheme distinguishes the morphed traffic from the mimicked target with a success rate of 99%; while the previous scheme distinguishes only 25%. When traffic is morphed according to bigram (i.e. two adjacent packet sizes) distribution, our scheme distinguishes 98.7%, versus 22.5% for the previous scheme. When bigram morphed traffic are mixed with traces of 2,000 other websites, our scheme correctly identifies 78% of the morphed traffic, while the previous scheme identifies 52%.

Our analysis suggests that both sizes and ordering features should be removed in countermeasures to website fingerprinting. We suggest some countermeasure exploiting randomization in packets or HTTP requests and responses. The countermeasure provides some communication privacy and is low in bandwidth overhead, though it may trade off the response time.

7.2 Website Fingerprinting under Traffic Morphing

Our scheme is able to differentiate website fingerprints even under traffic morphing [90]. For the ease of discussion, let us call the source website W_S, and its morphed traffic S'. Website W_S mimics a target website W_T. The traffic of W_T is T. From S', we want to identify W_S.

We differentiate W_S and W_T fingerprints by packet ordering. Assume that we know website W_S morphs its traffic, while website W_T does not, a large edit distance between T and test stream S' indicates that S' is not from W_T, because the fingerprints of W_T should be fairly consistent without morphing. Hence, test stream S' is from website W_S.

If there are k websites imitating the same target W_T, source W_S is likely uniquely identifiable. The reason roots at the morphing constraint to minimize bandwidth overhead. A morph algorithm maps each source packet to some minimally different sizes in the target distribution. Because of the correlation in packet sizes before and after morphing, and the consistency in the ordering of unmorphed packets, morphed traffic also have reasonably consistent order. It leaves us a loophole to differentiate among the k websites.

In evaluations, we examine the identifiability of k morphed websites amid other websites not related by morphing. We preprocess to narrow down the candidate websites by packet size distributions, e.g. using $L1$ distance measurement.[1] On one hand, the preprocessing safeguards that the morphing websites are not filtered prematurely. Morphed streams tend to have larger variations in packet ordering, because a source packet can map to a few target sizes and the choice is probabilistic for each morphing instance. On the other hand, preprocessing eliminates noise websites that the test stream may incidentally share similar size sequence with, but are dissimilar in size distributions.

Since we evaluate the similarity of fingerprints based on the whole sequence of feature values, our scheme can recover the identity of the morphed traffic, as long as morphing does not handle the distribution of n-gram for n close to the sequence length.

7.3 Tradeoffs in Morphing N-Gram Distribution

We now consider the tradeoffs if traffic morphing preserves some packet ordering as in morphing bigram (two adjacent packet sizes) or higher-gram distributions.

While unigram morphing changes the distribution of single packet size, bigram morphing changes the distribution of two consecutive packet sizes, to make the traffic appear

[1] $L1$ distance between two website fingerprints is computed as the sum of absolute differences in their packet size distributions.

as if from some target website. The problem of finding the morph matrix is formulated as an optimization problem, in which the goal is to minimize the bandwidth overhead, and the constraints are the source and target packet size distributions. The morphing matrix is pre-computed and it transforms the source to target packet size distribution with minimum overhead. It is easy to understand that when n-gram distribution is satisfied, m-gram distribution for any $m > n$ is not necessarily satisfied.

Traffic morphing has an important objective of bandwidth efficiency. However, if traffic morphing is extended to n-gram ($n \geq 2$) to consider some packet ordering information, its bandwidth usage becomes increasingly inefficient. The reason is that higher-gram morphing has to concurrently satisfy all the lower-gram distributions, so there are very limited choices for the size of the n-th packet once the previous $n - 1$ packet sizes are fixed. Source packet sizes have to map to the target distribution even though the size differences are large. For some value of n, the bandwidth efficiency of traffic morphing drops below packet padding schemes.

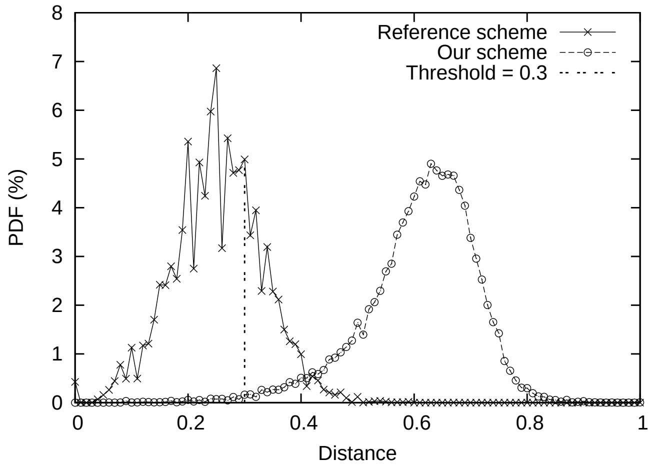
N-gram morphing has a considerable computation cost which grows exponentially with the number of grams considered. The computation is performed to find the morphing matrix that minimizes the bandwidth overhead. In the worst case, n-gram morphing needs to optimize $x^{2(n-1)} + x^{2(n-2)} + ...x^2 + 1$ matrixes per source-target pair of websites, with each matrix containing x^2 elements. Here x represents the number of distinct lengths of packet payload. Its maximum is 1461 for Ethernet packets. If we consider unigram morphing trades off computation for bandwidth, then when n increases, all the computation, storage and bandwidth costs of n-gram morphing increase.

7.4 Evaluation

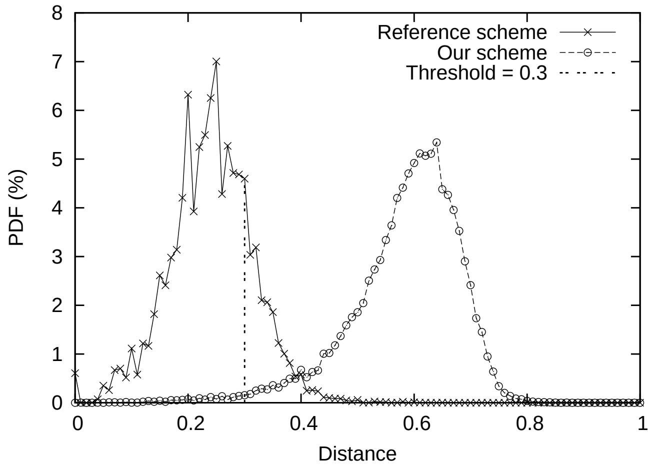
In this section, we empirically show that our scheme can differentiate fingerprints of websites morphed by packet sizes with limited ordering information, and we verify that there is significant bandwidth overhead for n-gram ($n \geq 2$) morphing.

7.4.1 Fingerprint Differentiation under Traffic Morphing

We perform three sets of experiments to evaluate the fingerprint differentiation ability of our scheme under traffic morphing. We use Liberatore and Levine's scheme [50] for performance comparison, and refer to it as the *reference scheme*. Our scheme that performs preprocessing is referred to as the *improved scheme*, otherwise it is called the *basic scheme* or simply our scheme.



(a) Unigram morphing



(b) Bigram morphing

Figure 7.1: Distribution of distance between morphed traffic and the mimicked target

Differentiating Morphed Traffic

We take 2,000 websites as the mimicked targets, and generate the morphed traffic such that the packet size distributions are within $L1$ distance of 0.3 between a morphed traffic and its target. The same $L1$ distance threshold is used in the traffic morphing scheme [90]. For each target website, we generate 4 variants of the morphed distributions, and draw 5 instances from each variant. The distance in our scheme is measured using edit distance; while that in the reference scheme [50] is $1 - S$, where S is Jaccard's coefficient. The distribution of the distances between the morphed traffic and the target traffic in unigram morphing and bigram morphing are shown in Figure 7.1a and Figure 7.1b, respectively.

For unigram morphing as shown in Figure 7.1a, compared to the reference scheme, our scheme shifts rightwards the range of distances from 95% in the interval [0.1, 0.4] to 95%

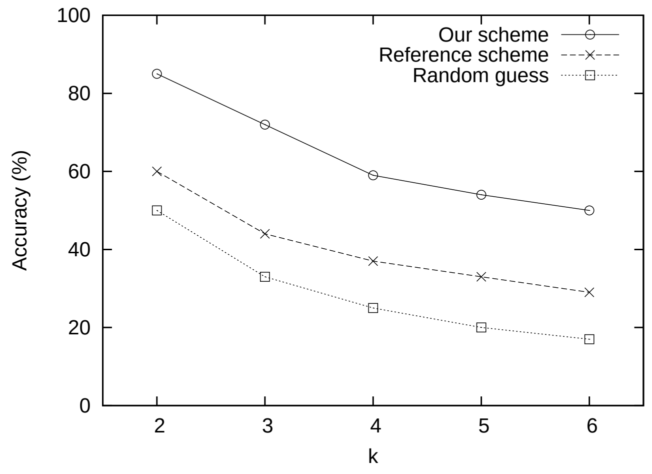


Figure 7.2: Identification accuracy of k websites that morph to the same target

in [0.4, 0.8]. The larger distance indicates that our scheme differentiates more clearly the morphed traffic and the target. The vertical bar at 0.3 indicates the distance threshold, and a test returning a value below the threshold means that the distance evaluation cannot separate the morphed instance and the mimicked target. For the reference scheme, 75% of the tests fail; while for our method, there is only 1.0% error cases, i.e., our scheme differentiates 99% of the morphing traffic. As shown in Figure 7.1b, the experiment on bigram morphing yields similar results. For the reference scheme, 78% of the tests fail; while for our method, there is only 1.3% error cases. The experiment result shows that although traffic morphing is effective against the reference scheme, our scheme is highly resistant to it.

Identifying Multiple Websites Morphed to the Same Target

We extend the evaluation to multiple websites morphing to the same target, and compare the distinguishability using our scheme, the reference scheme and random guess. We take 20 disjoint sets of k websites, where the first $(k-1)$ websites bigram morph towards the k-th website. Each morphing website generates 6 instances, 3 as learning samples and the rest 3 as test cases. We vary k from 2 to 6, and measure the identification accuracy of morphed traces in each set. The average identification accuracy of all sets at each k is presented in Figure 7.2.

When our scheme differentiates between $k = 2$ websites, i.e. the source and target of morphing, it yields a high identification accuracy of 85%. As k increases, more websites share the same morphing target, the accuracy gradually decreases. When $k = 6$ candidates, which means five source websites morph to one target, our scheme has an accuracy of 50%. In contrast, accuracy of the reference scheme is 29% when $k = 6$, while random

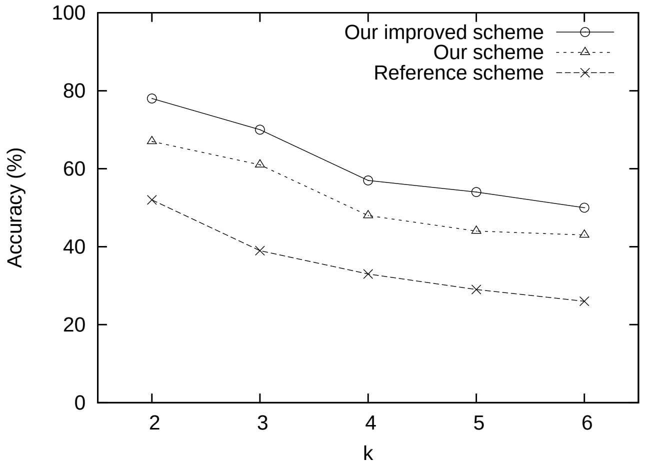


Figure 7.3: Identification accuracy of k morphed websites among 2,000 other websites

guess gives a probability of 17%. The reference scheme performs not much better than random guess. It shows that the reference scheme cannot reliably distinguish websites that share similar packet size distribution. Our scheme is much stronger at differentiating among multiple morphing websites and their mimicked target.

Identifying Morphed Traffic Mixed with Other Websites

Previous experiments examine the distinguishability within morphing sources and their target. Next we evaluate the identifiability of 20 sets of k morphing websites when they are mixed with 2,000 other websites. Morphing websites are profiled based on their morphed traces. We set the $L1$ distance threshold to be 0.4 in screening websites by packet size distributions. The threshold is slightly larger than in morph algorithm computation to accommodate some incompliance between the computed and generated packet size distributions. The incompliance is caused by packet splitting which generates new packets that are not accounted for in the computation of morph algorithm. Yet the threshold cannot be too large, so as to avoid keeping as candidates of websites not in the morphing relationship. Large threshold in $L1$ distance will add noise to the identification.

The identification accuracies are shown in Figure 7.3. Comparing Figure 7.3 and Figure 7.2, our improved scheme performs equally well when it identifies k morphing websites among a large set of unrelated websites and when it differentiates within the morphing websites. When k varies from 2 to 6, our improved scheme has an identification accuracy ranges from 78% to 50%, while the reference scheme has a corresponding range of 52% to 26%. Even when $k = 6$, meaning each target website has 5 imitators, our identification accuracy of the imitators is still 50%. The performance of our basic scheme in Figure 7.3 degrades compared to in Figure 7.2, because some morphed traces incidentally map to websites unrelated by morphing. However, its performance is still better than the reference scheme. Our basic scheme identifies the morphing websites with an accuracy of 61% at $k = 3$, the reference scheme correctly identifies 39%.

We run our improved scheme on 2,000 non-morphing websites. It gives the same identification accuracy of 81% as the basic scheme in this case. It is compatible to website fingerprinting at the absence of morphing, since non-morphing websites have HTTP streams that are consistent in both packet size distributions and ordering.

7.4.2 Bandwidth Overhead of N-Gram ($N \geq 2$) Morphing

We perform an experiment to verify that there are few choices of packet sizes for the n-th packet once the previous $n - 1$ packet sizes are fixed. We retrieve 6-second traffic from 2,000 most popular websites, and analyze the distribution of the sizes of n consecutive packets for each website, then average the results. The experiment results are presented in Figure 7.4.

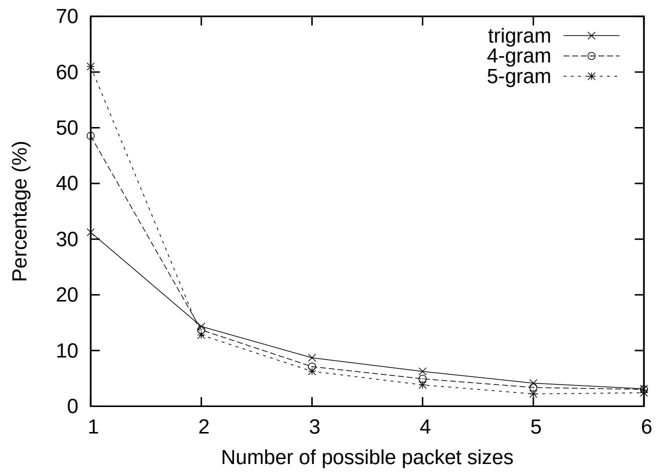


Figure 7.4: Distribution of the number of possible packet sizes for n-gram morphing

In Figure 7.4, for trigram, 4-gram and 5-gram morphing, once the sizes of the previous 2, 3 and 4 packets are fixed, there is only one possible packet size for the third, fourth and fifth packet with probability 31.2%, 48.5% and 61.0%, respectively. Similarly, there are no more than three possible packet sizes for the n-th packet with probability 54.2%, 69.4% and 77.2% for trigram, 4-gram and 5-gram morphing, respectively. The experiment results show that with high chance, there are very limited parameters that can be adjusted to reduce the bandwidth overhead. Thus when morphing considers packet ordering for $n \geq 3$, it becomes very poor in bandwidth efficiency.

We further present the numerical evaluation on the bandwidth overhead of bigram morphing ($n = 2$), in comparison to mice-elephant packet padding. Though Wright *et al.* have evaluated n-gram morphing ($n = 2, 3$) on other applications, the bandwidth overhead on defending against website fingerprinting they presented was only for unigram morphing ($n = 1$), in comparison to padding packets to the path MTU [90]. Mice-elephant packet padding is a simple and effective padding scheme against website fingerprinting, and it causes less bandwidth overhead compared to maximum packet padding.

We use traces of 100 websites from the *SSH2000* dataset. The i-th website is morphed to the $(i + 1)$-th website. Packet sizes can be increased through padding and decreased through fragmentatioin. Each HTTP stream is morphed 5 times. The average overhead is presented in Figure 7.5.

Figure 7.5a shows the percentage of added packets due to traffic morphing. The added packets are caused by packet splitting, which are not captured in but in fact change the source packet size distribution. The pre-computed morphing matrix does not apply to optimize the transformation of the dynamically added packets. The target sizes of these packets are sampled from the target size distribution to ensure they too follow the distribution after morphing. Transmission of extra packets incurs overhead in bandwidth and delay. From our data, more than 7 times of packets are added due to morphing for some website. Creation of extra packets is most severe when the source packets are mostly large while the target packets are mostly small. The source packets are fragmented several times iteratively.

Figure 7.5b shows the average added bytes per packet by morphing and mice-elephant. We count only the bytes padded due to morphing a small packet to a large one, but not the header bytes added due to packet splitting. A packet may be split one or multiple times. Only the initial split is guided by the morphing matrix, subsequent morphing operations depend on the sampling of target sizes. The randomness can affect the amount of added header bytes observed. Yet we note that if added packet headers are accounted, the average added bytes per packet will increase. Using mice-elephant, 80% websites have 150 to 300 bytes of overhead per packet; but using traffic morphing, about 90% websites need to send 300 to 500 extra bytes for each packet, which is much higher than mice-elephant. When source distribution contains a large percentage of small packets, while target distribution contains many large packets, then the padded bytes per packet is expected to be high.

Figure 7.5c shows the overall bandwidth overhead, comparing bigram morphing and mice elephant packet padding. Similar to the measurement of added bytes per packet, we do not count the bytes of added packet headers. From the figure, we can see that 20% websites have bandwidth overhead more than 100% of the actual data, using bigram morphing. The overall bandwidth overhead can be 4 times of the original source traffic. While using mice-elephant packet padding, 75% websites have the bandwidth overhead less than 50% of the total source packet sizes, and only 5% websites have bandwidth overhead more than the total source packet sizes.

We have discounted the packets needed to make the amount of data similar between the source and target websites, in the overhead measurement of morphing. If a target website has more data than the source website and the morphing scheme takes that into account, then the overhead of traffic morphing in terms of additional packets, extra bandwidth and padded bytes per packet may increase from the data we present here. In short, our experiment shows that n-gram morphing is less bandwidth efficient than a simple mice-elephant packet padding even when $n = 2$.

7.5 Countermeasures

Existing proposals of countermeasures to website fingerprinting cover up traffic features by modifying the packet sizes. The operations range from padding packets, repackaging packets to a single size and adding dummy traffic. Yet most countermeasures (e.g. packet padding and dummy traffic) are not deployed in practise due to the large bandwidth overhead they incur.

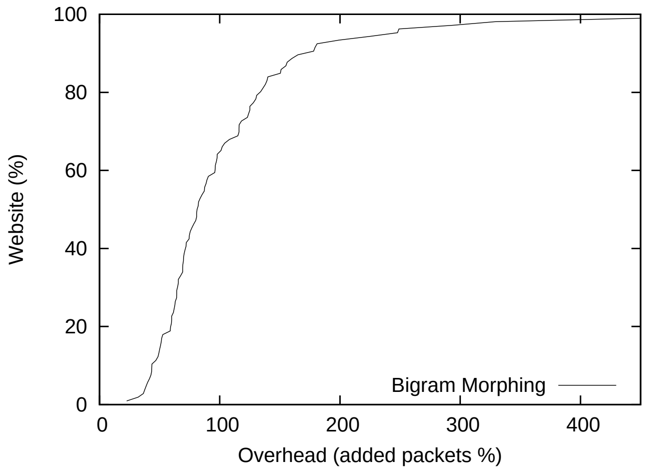
We suggest an alternative countermeasure that uses randomness to weaken the ability of website fingerprinting. The countermeasure randomizes on both packet sizes and request ordering, as information from size and timing channels can be exploited by website fingerprinting schemes. To muddle the ordering of object downloads, we have the browser to randomize the object request order. To conceal the request and object sizes, we have the server and the browser to package data into randomly sized packets. The countermeasure incurs low much bandwidth overhead.

Randomizing the Ordering of Object Requests

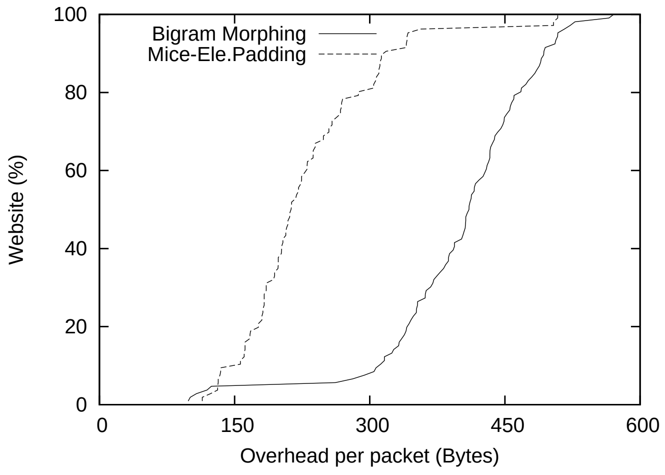
The website fingerprints we extract are consistent because of the predictable behaviors of HTTP and browser. As part of a countermeasure, we propose randomizing the ordering of object requests by browser. This functionality can be easily incorporated by implementing it as a browser plug-in. With the plug-in, browser buffers x HTTP requests of embedded objects, and sends them in a random order, where x can be a random number dynamically generated at each web access. The cost of it lies in buffering the object requests which may delay the presentation of a webpage slightly.

Randomizing the Packet Sizes

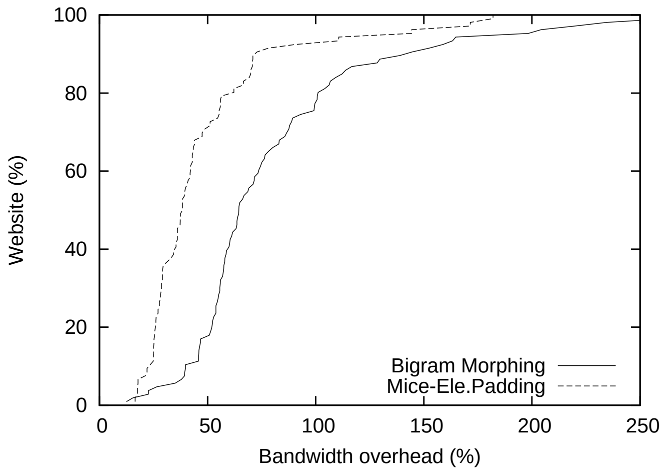
To make the sizes and ordering of packets untraceable, browser and web server each generates a sequence of random numbers for a web access, and buffers data to send in



(a) Added packets



(b) Extra bytes per packet



(c) Bandwidth overhead

Figure 7.5: Comparison of bandwidth overhead between bigram morphing and mice-elephant packet padding

packets of the random sizes. The operations include splitting the packet data and merging data from multiple packets.

Web server generates a random number when there is data to transmit. If the data have fewer bytes than the random number, server waits for more data. This incurs some buffering delay. If no more data are to be sent, or the waiting time expires, the server simply releases the packet. If the amount of data is more than the randomly generated target size, server splits the packet, and processes the remainder of the packet in the same way. Random delay can be added to conceal which packet sizes have been touched up in the HTTP stream. Although data buffering causes delay, the countermeasure significantly increases the difficulty of website fingerprinting.

We remark that neither packet padding nor the countermeasure we propose is perfect. They only hide most of the consistent features available for fingerprint. An HTTP stream still leaks information on the approximate total size of a website. In addition, packet padding cannot change the number of object requests. Hence dummy traffic is needed to cover up these features. The dummy traffic should be augmented without distinguishable patterns in sizes or timing, in order to prevent filtering or statistical traffic analysis. It presents a tradeoff between communication anonymity and communication efficiency in time and bandwidth.

7.6 Summary

Our website fingerprinting scheme is designed to withstand traffic morphing. Essentially, our scheme is effective even under traffic morphing because traffic morphing cannot handle packet ordering and satisfy low bandwidth overhead simultaneously, whereas we are able to identify and exploit the seemingly noisy packet ordering information for website fingerprinting. If morphing considers packet ordering information, its computation cost, storage cost and bandwidth overhead all increase. Its bandwidth efficiency is worse than mice-elephant packet padding, even in bigram morphing. Our scheme is able to identify websites from morphed traffic with significantly higher accuracy than the previous scheme.

Most existing countermeasures to website fingerprinting are not deployed in practice for bandwidth efficiency concerns. We propose some countermeasure with almost zero bandwidth overhead. To defend against website fingerprinting, we randomize the packet sizes and the order in which embedded objects of a webpage are fetched. Effectiveness of the countermeasures depends on the aggressiveness in applying them, though at a correspondingly heavy cost of performance and hence user satisfaction. While padding trades off network bandwidth to conceal the packet sizes; our proposed countermeasure trades off the delay.

Chapter 8

Active Website Fingerprinting over Tor

We consider the traffic model of web browsing over Tor, where the HTTP stream is encrypted, tunneled and the data are transmitted in fixed sized units. Building on the passive website fingerprinting systems over SSH and SSL tunnels, we propose an active website fingerprinting model over Tor. By spacing out the object requests, the active approach obtains the sizes and order of web objects as website fingerprints.

8.1 Overview

Tor [25] is a popular circuit-based low-latency anonymous communication network. It provides anonymity protection to users over TCP applications, e.g. web browsing. The web browsing traffic flows through a Tor circuit in fixed length cells, encrypted and decrypted by Tor routers in an onion-like fashion. A Tor circuit typically consists of three routers, which are called entry node, middle node and exit node respectively. The Tor routers chosen to form a circuit can be user specified or randomly selected. Tor sets up three connections along a circuit, such that it can switch connections to transmit data if the current connection is not stable.

Some research works have investigated attacks that degrade the anonymity provided by Tor, notably traffic confirmation attacks [87, 94] and statistical disclosure attacks [9, 44, 22, 52]. Traffic confirmation attack embeds some flow watermark into sender's outbound traffic and verify the watermark at the suspected receiver to confirm a communicating pair. Statistical disclosure attack correlates sender's outbound traffic and receiver's inbound traffic in terms of timing or volume to statistically disclose the likely communication relationship. Note that these attacks require an adversary to be able to monitor the traffic at both ends of a communication.

We consider a website fingerprinting attack against Tor whose objective is to link a user with the website the user accesses through Tor. The attack allows an adversary to profile user interests and their web browsing habits. Existing website fingerprinting schemes [40, 78, 10, 50] target at certain encrypted communication tunnels, such as SSH tunnel or VPN. Yet they are not applicable to Tor. Fingerprinting schemes based on packet size related features [10, 50] do not work on Tor, as Tor transmits messages in fixed length cells. Existing schemes that use web object sizes as fingerprint features [40, 78] hold simplified assumptions about traffic patterns and hence cannot extract the desired features realistically. The size of a web object was determined by accumulating the data received through a TCP connection, assuming TCP connections were not reused to download multiple embedded objects [40]. Alternatively, it was determined by accumulating the data received between adjacent HTTP requests, assuming object requests were not pipelined [78]. However, HTTP supports pipelining and TCP connections are reused. A key difficulty that prevents passive fingerprinting models from identifying features of web objects lies in the ambiguity in distributing data packets back into their respective objects. Message chunks cannot be differentiated by packet sizes in Tor.

We propose an active website fingerprinting model that extracts web object sizes and order as the fingerprint features. The scheme achieves a promising accuracy when it is used against Tor. In our active approach, the adversary holds each HTTP object request for a couple of seconds before releasing it into the network, to ensure that web objects are downloaded one at a time in each TCP connection. The adversary measures for each outgoing request the number of corresponding incoming Tor packets, which is counted as a web object size. The adversary extracts the fingerprint of a website from its web object sizes and order when the website is fetched via Tor. A testing fingerprint is matched against the fingerprint database to determine the website a user surfs.

While anonymous email systems are designed to resist attacks in which all communication links are being monitored, Tor aims to protect the anonymity of its users only from non-global adversaries. Tor assumes an adversary has the ability to observe and control part of the network, but not its totality. Even though it is controversial whether Tor's restriction on attacker's capability is realistic, our attack demonstrates that an adversary can extract vital information about the anonymously accessed websites for identification, without stepping outside the threat model considered by Tor.

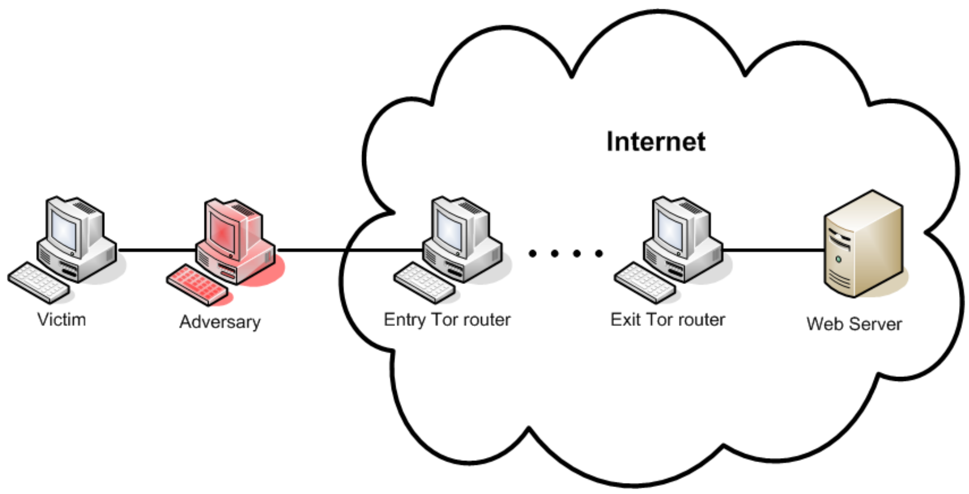


Figure 8.1: Traffic intervention and analysis setup

8.2 Active Website Fingerprinting Model

Our active website fingerprinting model is designed for Tor, yet it applies to website fingerprinting over other proxy plus encryption technologies, including SSH or TLS tunnels and VPN. Tor traffic is comparatively the most difficult to fingerprint, as it leaks the least amount of size related information. Whereas other technologies do not intensionally cover up the size related features.

8.2.1 Traffic Analysis Setup

We propose an *active* website fingerprinting model. The setup for traffic analysis over Tor is illustrated in Figure 8.1. The victim host is connected to the adversary, which in turn connects to the Tor entry node, and the exit Tor node connects to the web server to fetch the user desired webpage. The adversary is a man-in-the-middle between the victim and the Tor entry node, and hence able to tamper with victim's traffic. When victim's packets arrive at the adversary, they are held for a couple of seconds before being forwarded to the entry Tor router. In real life, such adversaries can possibly be company network administrators monitoring web accesses of its employees, or ISPs gathering web access statistics or profiling user interests.

Comparing Figure 8.1 and Figure 6.1, we can see the traffic analysis setups for website fingerprinting over different channels are similar. The difference is that the adversary not only eavesdrops but also intervenes with the transmission of an HTTP stream over Tor. Note that a Tor circuit is set up with three routers by default, which eliminates any single point from knowing both the source and destination of a session. It is arguably stronger in providing anonymity in web browsing. Yet in our website fingerprinting models, it does not matter how many proxies are lying behind the adversary. The same effect is achieved that the HTTP stream is encapsulated and encrypted. The difference between an anonymous channel through Tor and through an SSH or SSL tunnel is that Tor sends messages in fixed length cells. This difference urges us to use an active fingerprinting approach, as any size related features are hard to be extracted with a passive model over Tor.

Our active model exploits the side channel information about web objects as features of website fingerprints, while supporting the common traffic patterns in HTTP streams. The traffic patterns our model accommodates include multiple TCP connections within a session and HTTP pipelining. As several objects can be concurrently downloaded in different connections, we keep track of connections individually. HTTP pipelining allows multiple objects to be downloaded in a pipelined fashion within a connection, so data of different objects can interleave. We extract per web object features by intervening with the transmission of HTTP requests, which undoes pipelining to the effect.

Like other web fingerprinting schemes, our attack involves the adversary to build a fingerprint database which is applied to match the victim's traffic. To build the fingerprint database, the adversary surfs to a large set of frequently accessed websites, and apply the active fingerprinting technique to extract the fingerprints of each website. Fingerprint from the victim's HTTP stream is extracted similarly, which is then matched with the fingerprint database for website identification.

Our website fingerprinting model is within Tor's security model. Tor's security model allows a non-global observer and allows control over part of the Tor network. We monitor the encrypted and proxied traffic only at the client end of a web browsing session. The requirement on adversary's capability is even easier to satisfy than traffic confirmation attacks.

8.2.2 Features for Fingerprint

We select the features for website fingerprinting by examining the process that a web browser fetches a webpage. Typical communication between a client browser and a web server for downloading a webpage is illustrated in Figure 2.1. Given a website URL, web browser fetches the webpage by first issuing an HTTP GET request. In response, the web browser receives an HTML object which may contain references to other web objects, such as graphics files or style sheets. The web browser then parses the HTML file, and issues an HTTP GET request for each of the embedded web objects. The GET requests are usually sent in parallel on multiple TCP connections, so as to speed up the process and prevent a single GET failure from delaying the rest of the webpage download. Thus, downloading a webpage actually results in sending a (fixed or variable) number of GET requests in some order. We observed that for the same web browser (eg. Firefox), the order in which the GET requests are sent remains fairly stable.

We use web object sizes and their ordering as features of a website fingerprint. Object based features are more persistent and more characteristic of a website than packet based features. The selected features represent properties of web objects and reflect the layout of a webpage. The consistent ordering of embedded objects in webpage retrievals roots from that the layout of a webpage is infrequently changed, and that a web browser uses the same strategy to schedule requests of the embedded objects. Passive techniques are insufficient to extract object sizes and their associated ordering from Tor, thus we propose using an active approach that delays the subsequent HTTP GET requests in order to determine the size of the currently being downloaded object. We describe details of our fingerprint extraction method in Section 8.3.1.

We could track the size sequence of all embedded objects, but it incurs substantial delay in the webpage presentation. Hence we choose to monitor only the first n objects in a website download. Our experiments show that with only 10 objects monitored per website, the fingerprint identification already achieves high accuracy. We note that the identification accuracy will increase as the number of objects being monitored increases, yet at the same time, so will the incurred delay be increased.

In addition to web object sizes and ordering, we keep track of the total size in object requests and that in object responses, as they are helpful to differentiate websites especially when the websites' partial object size sequences are similar.

In short, the fingerprint F_A of website A is $F_A = (q_A, t_A)$, where $q_A = \langle s_1, s_2, s_3, ...s_n \rangle$ is a sequence representing the first n object sizes, and $t_A = (x_A, y_A)$ where x_A and y_A are the total request and response sizes respectively.

8.2.3 Similarity Comparison

To match the fingerprint extracted from a testing packet stream against the website fingerprints in the database, we need some similarity measurement mechanism. The measurement should take into consideration the ordering of feature values and be efficient. We extend edit distance and apply it to the similarity comparison of website fingerprints over Tor.

We adapt edit distance to measure the distance between two integer sequences, where each sequence represents order and sizes of the downloaded objects of a webpage. Edit distance is often used to measure the dissimilarity of two strings. The common edit operations it allows include insertion, deletion and substitution of a single character. We propose supplementing *split*, *merge*, and *split and merge* to the list of allowed edit operations on integer sequences. Given integer sequences $S_A = \langle c \rangle$ and $S_B = \langle a, b \rangle$, S_A can be transformed into S_B using a single *split* operation, if a and b sum to c. *Merge* is the reverse operation to split. It combines two neighboring integer values into one.

Given $S_B = \langle a, b \rangle$ and $S_C = \langle c, d \rangle$, a *split and merge* operation can change S_B to S_C if $a + b = c + d$. It accounts for the effect that part of b is split and grouped with a, or part of a is split and clustered with b. The same modification can be effected with two substitutions, but the associated costs are different.

The reason we extend edit distance with the above operations is to handle differences in website fingerprints due to variations in network delay and HTTP pipelining configuration. If pipelining is enabled, an Ethernet packet may contain more than one short object requests. Tor proxy located at the client host sends out the Ethernet packet part by part in sizes of Tor's transmission unit. Two object requests could be released within the same Tor packet, if the last part of the first request and the whole of the second request are contained in one Tor packet. Thus data of the two responses may interleave and cluster in time, despite we artificially space out object requests by Tor packets. Pipelining configuration and browser processing delay determine if multiple requests are packed into one Ethernet packet and hence affect the observable sequence of object sizes. They can vary across traces even if the website has not updated. *Merge* and *split* correspond to clustering two object sizes or not in fingerprint comparisons. Network delays also affect the clustering of object data, as our identification of object boundaries is based on timing heuristics. Prolonged delay of some intermediate response packets can lead to a false identification of object boundary and hence the remaining data of the current object be accumulated with data of the next object. *Split and merge* caters for regrouping partial object data in similarity measurement.

8.3 Website Fingerprinting Scheme

In this section, we describe how we implement the active website fingerprinting model proposed.

8.3.1 Determining Object Sizes and Order

We take an active approach to determine web object sizes and ordering with some heuristics. To handle multiple connections set up through a Tor circuit, we monitor each connection individually. We maintain state variables of each connection and swap them in and out as required by parsing the Tor packet stream. In addition to multiple connections and HTTP pipelining, Tor obfuscates website features against passive observation using techniques of encapsulation, encryption and unifying packet sizes. HTTP pipelining allows several object requests to be concurrently served, so data of different objects can interleave within a connection. Web server address is encapsulated into the packet payload. Encryption protects the data confidentiality. Uniform data transmission unit destroys most size related features about packets. It is important to distinguish data

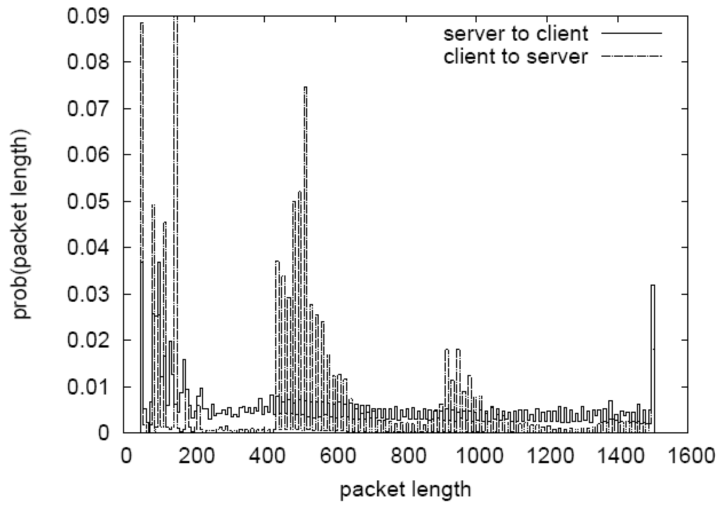


Figure 8.2: Distribution of per-trace unique packet sizes of 2,000 popular websites [50]

packets from control packets, and to associate data packets to their respective objects for object size identification.

Identifying Object Responses with Heuristics on Sizes

A Tor packet cannot be distinguished individually whether it belongs to data packets or control packets, but we can classify a packet depending on whether it forms a response alone or with several other packets. To distinguish response data packets from control packets, we use a heuristic that for efficient transmission, control packets tend to be smaller than 1KB, or equivalent to 2 Tor packets (512 bytes each) or less, hence responses containing more than 2 Tor packets are likely transmitting object data. Although websites often have objects that are smaller than 1KB, we do not use sizes of these objects as website features, because small objects mix with control packets in sizes, and it is the larger objects revealing more distinctive information about websites after all.

We remark that we cannot reliably distinguish each request size. Most object requests are 400 to 600 bytes or 900 to 1000 bytes long (request size statistics of 2,000 websites are shown in Figure 8.2). It is possible that a request spans over more than one Tor packet, thus an outgoing packet yielding no responses could be a control packet as well as part of an incomplete request. Fortunately, our goal is to obtain individual object sizes in a sequence, which is sufficient for fingerprinting websites with good accuracy, the request sizes only provide additional information if available.

Determining Object Boundaries by Request Holding and Heuristics on Timing

To cluster response packets that belong to one object and hence determining the object size, we artificially space out the object requests. We release victim's outgoing Tor packets one at a time, and hold any subsequent outgoing Tor packets until all the response data to the currently released Tor packet have been received. Thus a response size can be determined by accumulating the amount of data received in correspondence to a request.

Because of encryption and uniform packet sizes, we cannot distinguish by clear markers where an object's data end. Instead, we use a heuristic based on timing to determine object boundaries. We assume object data comes in a stream such that the packet intervals are reasonably small. If we wait for an interval, and there is no incoming packets, we consider the object download is completed. The threshold of packet interval must be greater than the round trip time of the connection for the first data packet, otherwise an object boundary will be falsely determined before the first data packet comes back. Yet once the first packet comes back, the incoming packet intervals are caused only by processing delays of web server and Tor routers, and affected by the network connection conditions. So the interval threshold for subsequent packets can be smaller. In our experiments, we set the packet interval threshold to be $T1 = 5$ seconds before the first response packet arrives, and $T2 = 2$ seconds for subsequent packets. Compared to using $T1$ alone, having two timers $T1$ and $T2$ speeds up the downloading of a webpage, for otherwise there would be some unnecessary waiting time after the last data packet of an object is received.

8.3.2 Fingerprint Similarity Comparison

Edit distance measures the number of edit operations to make two strings identical. The common edit operations considered are insertion, deletion and substitution of a single character.

We extend edit distance to match object size sequences of website fingerprints. Since the observed object sizes may split or merge due to HTTP pipelining or delay variations, we extend edit distance to allow split and merge operations. Pseudocodes to compute our extended edit distance is shown in Appendix B.

The extended edit distance can be computed using Dijkstra's shortest path algorithm. We search for the minimal distance from $n(0, 0)$ to $n(|q_A|, |q_B|)$, where $n(i, j)$ is a node corresponding to the i-th element of q_A and the j-th element of q_B, and $|q|$ is the length of sequence q. The nodes are connected by edit operations with associated costs.

The costs of edit operations are tunable. We consider insert, delete and substitute should have a higher cost than a split or merge. This is because in website fingerprints, split or merge of an object size is transient and due to HTTP pipelining or delay variations, but insert, delete or substitute reflects the change of embedded objects of websites.

We normalize edit distance by an upper bound of the distance between sequences q_A

and q_B, as shown in Formula 8.1.

$$Ed(q_A, q_B) = \frac{DijkstraDistance(q_A, q_B)}{\max(|q_A|, |q_B|) \cdot c} \tag{8.1}$$

where $|q|$ denotes the length of sequence q, and c is a cost factor which we set as the deletion cost.

Difference in the total sizes t_A and t_B of websites A and B are measured by summing their absolute differences in request and response sizes, and normalized by their total size, as shown in Formula 8.2.

$$L1(t_A, t_B) = \frac{|x_A - x_B| + |y_A - y_B|}{x_A + x_B + y_A + y_B} \tag{8.2}$$

where x_I and y_I are the total request and response sizes of website I respectively, and $|x_A - x_B|$ is the absolute difference between x_A and x_B.

The similarity measure between website fingerprints F_A and F_B combines the information on object sequences and total sizes. It is computed as Formula 8.3.

$$Sim(F_A, F_B) = 1 - \frac{Ed(q_A, q_B) \cdot w_1 + L1(t_A, t_B) \cdot w_2}{w_1 + w_2} \tag{8.3}$$

where w_1 and w_2 are adjustable parameters, representing the weights of similarity in object size sequences and total sizes respectively. We set $w_1 = \max(|q_A|, |q_B|)$ and $w_2 = 2$, so that the weights correspond to the number of feature values considered. (1 - normalized distance) converts the measurement from distance to similarity. Identity of a test stream is given by the most similar fingerprint of the profiled websites.

Example

Suppose we are given two sequences q_A and q_B, where $q_A = \langle 123, 780, 465, 324, 230, 235\rangle$ and $q_B = \langle 123, 789, 345, 120\rangle$. We evaluate the edit distance between q_A and q_B with consideration of our extended set of edit operations.

To make q_A identical to q_B, the edit operations we allow include insertion, deletion and substitution of a single element, and in addition, merging two adjacent elements, splitting an element into two, and redistributing two adjacent elements by merging them and then followed by a split. Figure 8.3 illustrates these edit operations. Node $n(i, j)$ corresponds to the i-th element of q_A and the j-th element of q_B, the edges correspond to the edit operations with associated costs. $q_A = \langle 123, ...\rangle$ can be changed to $q_B = \langle 123, 789, ...\rangle$ by inserting 789 from node $n(1, 1)$. Similarly, $q_A = \langle 123, 780, ...\rangle$ can be modified to $q_B = \langle 123, ...\rangle$ by deleting 780 from node $n(1, 1)$, or $q_A = \langle 123, 780, ...\rangle$ can be updated to $q_B = \langle 123, 789, ...\rangle$ by substituting 780 to 789 at node $n(1, 1)$. In addition,

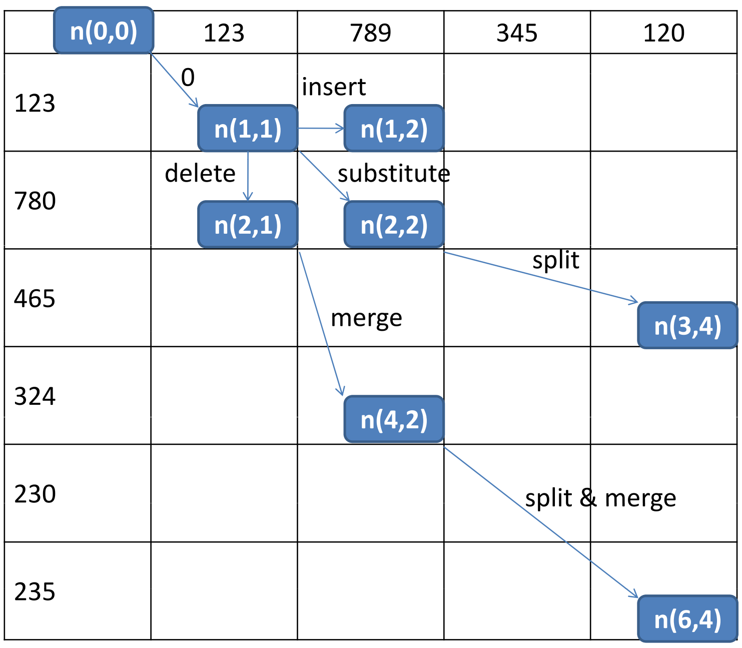


Figure 8.3: Example illustrating edit operations between integer sequences

$q_A = \langle ..., 465, ...\rangle$ can be changed to $q_B = \langle ..., 345, 120\rangle$ by splitting 465 to 345 and 120 at $n(2,2)$, or $q_A = \langle ..., 465, 324, ...\rangle$ can be modified to $q_B = \langle ..., 789, ...\rangle$ by merging 465 and 324 to 789 at $n(2,1)$. Finally, redistribution can be used in updating $q_A = \langle ..., 230, 235\rangle$ to $q_B = \langle ..., 345, 120\rangle$ from node $n(4,2)$. Note that insertion, deletion and substitution traverses the graph by one grid rightwards, downwards and diagonally respectively. In comparison, split traverses the graph two grids rightwards and one grid downwards, merge traverses the graph one grid rightwards and two grids downwards, and redistribution traverses the graph diagonally by two grids. We are amused at its resemblance to walking a chessman on a chessboard.

To measure the edit distance between q_A and q_B, we search for a path that traverses from node $n(0,0)$ to $n(6,4)$ with minimum cost. If all the edit operations have a unit cost, then the edit distance between q_A and q_B is 3. The path traversed is $n(0,0)$-$n(1,1)$-$n(2,1)$-$n(4,2)$-$n(6,4)$, and the edit operations performed are deleting 780 at $n(1,1)$, merging 465 and 324 to be 789 at $n(2,1)$, and finally redistributing 230 and 235 into 345 and 120 at $n(4,2)$, as illustrated in Figure 8.3.

8.4 Evaluation

We evaluate the effectiveness of our active website fingerprinting scheme by measuring the fingerprint identification accuracy. Our scheme should be considered effective if any non-negligible fraction of websites are correctly identified, as Tor is designed to protect anonymity of the communicating parties.

8.4.1 Data Collection

We collected traces over Tor of surfing to 200 unique websites daily in a few weeks. For each website, only its homepage is fetched. The 200 websites are chosen with the requirement that each of them generates sufficient traffic for fingerprinting to be feasible. With this requirement, our chosen websites all have images and none of them are purely text-based.

Using the traffic analysis setup illustrated in Figure 8.1, victim's outgoing packet arrives at the adversary machine. The adversary takes note of the packet contents and then drops the packet. After a few seconds, the adversary recreates the packet with the same content, and forwards it to the entry Tor router. This creates the effect that the adversary holds victim's packet for a couple of seconds.

We set the time limit on waiting for the initial data packet to be 5 seconds, and the subsequent data interval to be less than 2 seconds. Beyond the time limit, we consider the data transmission with respect to the current request has completed. The first 11 object requests are spaced out so that the first 10 object sizes are revealed. The rest requests are sent out without holding, to reduce the delay on website presentation.

Both the victim and the adversary were running on linux systems (Fedora Core 8). The Tor version used when the traces were collected is 0.1.2.17. The victim used Mozilla Firefox 2.0 to retrieve each URL via a SOCKS proxy (Privoxy). Privoxy then connects to the local Tor client on the victim's machine. The local Tor client chooses (by default) 3 Tor routers to form a circuit linking to the desired website. Communication between the victim and the website is tunneled through the circuit.

We configured Firefox not to attempt various extraneous connections, such as due to live bookmarks or automatic update checks. Browser cache is cleared before surfing to any website. Although disabling these features makes the resulting traffic somewhat less realistic, we believe it is a reasonable tradeoff to let us focus on the problem under investigation.

8.4.2 Identification Accuracy

In order to facilitate analysis and comparison, we classify the websites by their update frequencies, as either *static*, which updates once in several months, or *dynamic*, which updates at least daily. There are 50 static websites and 150 dynamic websites.

We examine the performance of our scheme and the effect of website updates. In particular, we evaluate the website identification accuracies when the test streams and the profiling streams are captured on alternate days, and two weeks apart. 4 training and 4 testing traces per website were captured on alternate days in the "next day test". In the "half-month test", 4 traces per website sampled in the first week were used to build the website fingerprint database, and 4 traces sampled two weeks later were used as testing traces for identification.

In evaluations, our search of minimal edit distance allows a feature value to be split, merged, or split and merged at most once, as the delay we impose to isolate the transmission of objects is sufficiently large that an object has low probability in being split into more than two parts. We tune the parameter values based on our dataset, and set the cost of a single split or merge to be 5, the cost of an insertion, deletion or substitution to be 6, and the cost of split and merge to be 7.

Dataset	Next Day Test	Half-Month Test
Static	73.5%	73.5%
Dynamic	70.5%	62.1%
Web200	67.5%	61.6%

Table 8.1: Website fingerprint identification accuracies.

The evaluation results are presented in Table 8.1. The identification accuracy of 200 websites is close to 70% in the next day test, and it is above 60% in the half-month test. As static websites update infrequently, their identification accuracy is not very sensitive to the age of website fingerprint profiles, and it remains at 73.5% for both tests. For dynamic websites, the identification accuracy drops over time as their contents change from the profiles. Yet even when the profiles were built half a month ago, their identification accuracy is still 62.1%. In comparison, if we randomly guess the website identity among dynamic websites, the probability of success is only 0.7%. Our scheme significantly reduces the anonymity provided by Tor.

8.5 Countermeasures

Applying aggressive padding to objects to conceal their sizes could be a countermeasure to active website fingerprinting, but it is not bandwidth friendly. Instead, we propose countermeasures that change the order of object requests and hide the sizes of individual objects. They incur little bandwidth overhead, and the buffering delay they cause tends to be insignificant.

(*i*) *Randomizing the object order.* Similar to the countermeasure against our passive website fingerprinting, browser buffers several object requests and sends them in different orders each time.

(*ii*) *Mixing data of different objects.* Browser packages data of different HTTP requests into a packet, such that a number of object requests become concurrent. Correspondingly, server interleaves data of different objects within packets. Thus object boundaries that previously align with packet boundaries are eliminated. The number of objects and which objects to request (serve) in each packet are decided with randomness by browser (server) in each web access, where the randomness helps to prevent fingerprinting. Random delay can be inserted between packets to further confuse the timing based identification of object boundaries.

8.6 Summary

We show that anonymous web accesses over Tor can be substantially hampered with active website fingerprinting, under Tor's somewhat restricted threat model, and without having to add external packet marks or flow marks.

In our proposed active website fingerprinting model, the adversary spreads HTTP GET requests issued by the victim client, so that only one GET request is served at a time. The adversary only releases the next GET request after all response packets are received for the current request. This helps to measure the corresponding web object size for each object request. A webpage download thus results in a sequence of web object sizes, where both the ordering and sizes of web objects are exploited when we match the website fingerprints.

Our study is the first in presenting such promising website fingerprinting accuracies over Tor. Tor is trusted by many in providing a low latency, anonymous communication channel, but this does not hold under our proposed attack.

Chapter 9

Conclusion and Future Work

We conclude this thesis by summarizing our findings from monitoring the models and analyzing the schemes of traffic source identification. We also underline the practical constraints that traffic source identification schemes have not handled thus far, and point out directions to enhancing the framework and improving the scheme designs.

Contributions

In this thesis, we investigated approaches to traffic source identification with respect to different traffic models and analyst's capabilities. We provided a framework of traffic source identification, in which we laid out interactions between model components and criteria for scheme designs. The principles are substantiated in our investigation under several specific problem scenarios, namely, probabilistic packet marking for DDoS traceback, passive website fingerprinting over VPN and active website fingerprinting over Tor. Our findings through the investigations are summarized below.

Traffic Source Identification Framework. We developed a framework of traffic source identification models. The framework captures the important components of traffic source identification design: obfuscation techniques applied on the source data and analyst's capability. Existing source identification schemes are classified in the framework by their component values. We deduced from the framework that it is possible to perform traffic analysis over Tor to identify the website accessed. We considered in the framework criteria to developing a traffic source identification scheme under several models. We considered minimizing collisions in packet marks for the PPM model of DDoS traceback, and that features in website fingerprinting are desired to present consistency for a website and maximum separation among websites. Overall, our framework is useful in classifying traffic source identification schemes, deriving novel source identification models, and guiding the construction of source identification schemes.

General Probabilistic Packet Marking (PPM) Model. Packet marking applies when the investigator is able to modify the packet header, which requires the packet headers are not encapsulated. We found an effective probabilistic packet marking scheme does not require sophisticated structure, if some network topology information is know. This is on the contrary to what previous works deliver. We also found that to fairly compare variants of packet marking schemes despite of their differences in assumptions, we can utilize the metric on the entropy of packet marks. Higher entropy indicates better utilization of the marking field and more information is transmitted, which then predicts a higher source identification accuracy assuming an optimal path reconstruction algorithm.

PPM Scheme: Random Packet Marking (RPM). Inspired by the findings from the general PPM model, we developed a packet marking scheme named Random Packet Marking (RPM). It is designed with high randomness in the packet marks generated, and hence it obtains significant performance improvement over prior work.

Passive Website Fingerprinting Scheme Exploiting Packet Ordering. Passive traffic fingerprinting applies when the traffic itself carries sufficient identifying information, such as the packet size distribution of a website. We found that besides packet sizes, packet ordering is fairly consistent for a webpage download. Such information is exposed regardless of encryption and proxy. Thus we developed a passive webstie fingerprinting scheme applicable to VPN, SSH or TLS encrypted tunnels, which utilizes the packet ordering information to enhance the identification accuracy of website fingerprints.

Analysis of Scheme Robustness against Traffic Morphing. Besides to enhance the website identification accuracy, an important motivation of incorporating the packet ordering information into website fingerprints is that the ordering information enables the website fingerprinting scheme to withstand traffic morphing [90]. Traffic morphing defends against website fingerprinting by transforming the packet size distribution of the source website to mimic a target website, while minimizing the bandwidth overhead. It targets at website fingerprinting schemes that exploit size related features only. We empirically verified that our scheme is robust against traffic morphing, as our scheme can differentiate websites having similar packet size distributions by packet ordering. We analyzed the traffic morphing scheme and showed that it cannot handle packet ordering while satisfying low bandwidth overhead. If some ordering information is considered, its bandwidth efficiency will be worse than some simple countermeasure of packet padding. Our analysis implies that an effective countermeasure to website fingerprinting should aggressively remove traffic features in both size and timing channels. We thus suggest randomizing both packet sizes and the ordering of HTTP requests to defend against website fingerprinting.

Active Website Fingerprinting Model and Scheme over Tor. Extending from the passive techniques, we proposed an active website fingerprinting model and scheme over Tor. The active approach helps to recover web object sizes when HTTP requests are pipelined. It works even when packet sizes are fixed. The fact that Tor transmits data in fixed length cells fails all website fingerprinting schemes that rely on *packet* sizes. We therefore use *object* sizes and ordering as website fingerprint features. In order to obtain web object sizes, we space out the download of web objects by holding HTTP GET requests, so that each object size can be determined by accumulating the amount of data received in correspondence to a request. Our active model and scheme demonstrated for the first time that it is possible to identify websites accessed through Tor from the size channel.

To perform active traffic source identification over Tor, investigators must have the capability of delaying Tor traffic, and the latency caused may be noticeable. Whereas the passive traffic analysis model on VPN requires only eavesdropping and is transparent to users. There is sufficient side channel information gathered from passive observations of VPN traffic to yield high website identification accuracy. Having said that, the model we designed for Tor is guaranteed to further improve the VPN fingerprinting accuracy, since more accurate and reliable features on objects are obtained.

Practical Considerations

We dealt with some of the methodological issues found in previous work that raise questions about the real-world performance of the techniques. In particular, our passive website fingerprinting model moves beyond the closed-world examination to a more realistic setting where a website must be detected from among a set of websites that the adversary may not even know about. We also evaluated the identification accuracy of website fingerprints with examinations of pipeline configurations, website variability, and the amount of training data.

However, there are still a couple of idealized assumptions in the models and analysis. In particular, probabilistic packet marking schemes for DDoS traceback all assume routers are cooperative. Because routers are administered under different autonomous systems whose management policies are largely independent, it is not easy to ensure routers along the attack paths are cooperative.

So far website fingerprinting proposals have not addressed the problem of concurrent web browsing sessions. All of them assumed that HTTP streams of different browsing sessions are separated correctly. In practise, it is not trivial to parse packets to their respective HTTP streams from encapsulated and encrypted traffic. Multiplexing connections over a single port complicates the matter and makes it more difficult.

Handling caching is another practical problem. The accuracy of website fingerprinting would be substantially deteriorated if caching is enabled. Browser caching affects the download of a webpage in that cached web objects are retrieved from the local cache and are removed from the traffic trace. Differences in the browser caches make the HTTP streams vary in different accesses to the same website. All proposals up to date have their performances heavily undermined by browser caches, which underlines the difficulty of the problem. The dynamics of cache contents makes the evaluation results probabilistic. Generally, evaluation with certain cache configurations presents limited analysis that tends to be biased. Further systematic and comprehensive modeling and measurement of cache as attributed to network buffer constraints or user behaviors is required. Fingerprinting models that take care of browser caching are yet to be proposed.

Partly because of the gap between the research settings and the practical traffic conditions, the threat to user privacy caused by the traffic source identification techniques is not arousing enough awareness it deserves. Even simple countermeasures are not deployed, for the currently proposed countermeasures all require paying the price of large bandwidth overhead or processing overhead.

However, when the technical assumptions are satisfied, traffic source identification presents a real threat to user privacy. Our proposals and experiments have made it clear that web clients' privacy can be compromised to a certain extend and the operations are easy to perform. In the performance centric design of Internet services, the idealistic defense proposals have insufficient deployment incentives. Therefore, not until more lightweight countermeasures are proposed, web users should remain alert of the capability of traffic source identification by legislative warden or an adversary.

Future Work

In future work for traceback schemes, we would find inspiration from flow marking techniques and reference schemes related to the identification of stepping stones to design a traceback model that does not require collaboration of routers across autonomous systems. Although it is commonly believed that routers are secured and trustworthy, from an analysis point of view, it would be interesting to investigate if some routers are malicious. We would analyze the trust management issues on packet marks and develop countermeasures to effectively isolate forged packet marks.

In future work for website fingerprinting, we would explore different methods to model the structural information of websites and evaluate their effectiveness in improving the fingerprint identification accuracy. We would compare the countermeasures, such as packet padding, dummy traffic, randomizing the packet sizes, and randomizing the order of object requests, so as to provide an in depth analysis of their robustness and tradeoffs. We would extend website fingerprinting to the scenario where concurrent sessions of web browsing are supported. We would design different heuristics and compare their accuracy in parsing concurrent browsing sessions from encapsulated and encrypted traffic. We would also extend the evaluation of website fingerprinting to the scenario where caching of web contents is enabled. We would construct a systematic model of web caching, and analyze the impact of caching on the effectiveness of website fingerprinting.

Bibliography

[1] Internet mapping project. Research, Lumeta,, Jan.

[2] Anomalous DNS activity. Current activity archive, US-CERT, Feb. 6, 2007.

[3] String-matching in Excel: VLookup() with fuzzy-matching to get a 'closest match' result. URL *http://hairyears.livejournal.com/115867.html*, May 2007.

[4] Abbott, T. G., Lai, K. J., Lieberman, M. R., and Price, E. C. Browser-based attacks on tor. In *proc. of Privacy Enhancing Technologies workshop (PET'07)* (Jun. 2007).

[5] Adler, M. Tradeoff in probabilistic packet marking for IP traceback. In *proc. of ACM Symposium on Theory of Computing (STOC)* (Nov. 2001).

[6] Allen, C., and Dierks, T. The TLS protocol version 1.0. RFC2246, Jan. 1999.

[7] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., and Sicker, D. Low-resource routing attacks agaisnt Tor. In *proc. of Workshop on Privacy in the Electronic Society (WPES'07)* (Oct. 2007).

[8] Bellovin, S., Leech, M., and Taylor, T. Icmp traceback messages. Internet draft, IETF, draft-ietf-itrace-01.txt, Oct. 2001.

[9] Berthold, O., and Langos, H. The disadvantages of free mix routes and how to overcome them. In *proc. of designing privacy enhancing technologies: workshop on design issues in anonymity and unobservability* (Jul. 2000).

[10] Bissias, G. D., Liberatore, M., Jensen, D., and Levine, B. N. Privacy vulnerabilities in encrypted HTTP streams. In *proc. of Privacy Enhancing Technologies workshop (PET'05)* (May. 2005), pp. 1–11.

[11] Bloom, B. Space/time trade-off in hash coding with allowable errors. *Communications of the Association for Computing Machinery 13*, 7 (1970), 422–426.

[12] Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., and Tofanelli, P. Revealing Skype traffic: When randomness plays with you. *ACM SIGCOMM computer communication review (SigComm'07) 37*, 4 (Oct. 2007), 37–48.

[13] Brekne, T., Arnes, A., and Oslebo, A. Anonymization of IP traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In *Proceedings of the Workshop on Privacy Enhancing Technologies (PETS'05)* (May 2005), pp. 179–196.

[14] Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 24*, 2 (Feb. 1981), 84–88.

[15] Chen, S., Wang, R., Wang, X., and Zhang, K. Side-channel leaks in web applications: a reality today, a challenge tomorrow.

[16] Cheng, H., and Avnur, R. Traffic analysis of SSL encrypted Web browsing. URL *http://citeseer.ist.psu.edu/656522.html*, 1998.

[17] Chor, B., Fiat, A., and Naor, M. Tracing traitors. In *proc. of CRYPTO* (Aug. 1994), pp. 257–270.

[18] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. Freenet: A distributed anonymous information storage and retrieval system. In *proc. of Workshop on Design Issues in Anonymity and Unobservability* (2000).

[19] Coull, S., Wright, C., Keromytis, A. D., Monrose, F., and Reiter, M. K. Taming the devil: Techniques for evaluating anonymized network data. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)* (Feb. 2008).

[20] Coull, S. E., Collins, M. P., Wright, C. V., Monrose, F., and Reiter, M. K. On web browsing privacy in anonymized netflows. In *Proceedings of the 16th USENIX Security Symposium (Sec'07)* (Aug. 2007), pp. 339–352.

[21] Coull, S. E., Wright, C. V., Monrose, F., Collins, M. P., and Reiter, M. K. Playing devils advocate: Inferring sensitive information from anonymized network traces. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS'07)* (Feb. 2007), pp. 35–47.

[22] Danezis, G. Statistical disclosure attacks: traffic confirmation in open environments. In *proc. of security and privacy in the age of uncertainty (SEC'03)* (May. 2003).

[23] Danezis, G., Dingledine, R., and Mathewson, N. Mixminion: Design of a type iii anonymous remailer protocol. In *proc. of IEEE Symposium on Security and Privacy (SP'03)* (May. 2003), pp. 2–15.

[24] Dean, D., Franklin, M., and Stubblefield, A. An algebraic approach to IP traceback. *ACM Transactions on Information and System Security 5*, 2 (May. 2002), 119–137.

[25] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *In Proceedings of the 13th USENIX Security Symposium* (2004), pp. 303–320.

[26] Edman, M., and Syverson, P. AS-awareness in Tor path selection. In *proc. of 16th ACM conference on Computer and communications security (CCS'09)* (Nov. 2009).

[27] Felten, E. W., and Schneider, M. A. Timing attacks on web privacy. In *proc. of ACM Conference on Computer and Communications Security (CCS'00)* (Nov. 2000).

[28] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. Hypertext transfer protocol – HTTP/1.1. URL *www.ietf.org/rfc/rfc2616.txt*, Jun. 1999.

[29] Findnot. `http://www.findnot.com`.

[30] Freedman, M. J., and Morris, R. Tarzan: A peer-to-peer anonymizing network layer. In *proc. of the 9th ACM Conference on Computer and Communications Security (CCS'02)* (Nov. 2002).

[31] Freier, A. O., Karlton, P., and Kocher, P. C. The SSL protocol version 3.0. URL *wp.netscape.com/eng/ssl3/ssl-toc.html*, Nov. 1996.

[32] Fu, X., Graham, B., Xuan, D., Bettati, R., and Zhao, W. Empirical and theoretical evaluation of active probing attacks and their countermeasures. In *proc. of 6th International Workshop on Information Hiding (IH'04)* (May. 2004).

[33] Garber, L. Denial-of-service attacks rip the Internet. *IEEE Computer 33*, 4 (Apr. 2000), 12–17.

[34] Goldschlag, D. M., Reed, M. G., and Syverson, P. F. Hiding routing information. In *proc. of the 1st international workshop on Information Hiding (IH'96)* (May 1996), pp. 137–150.

[35] Goodrich, M. Efficient packet marking for large-scale IP traceback. In *proc. of the 9th ACM conference on Computing and Communications Security (CCS'02)* (Nov. 2002), pp. 117–126.

[36] Gooskens, C., and Heeringa, W. Perceptive evaluation of Levenshtein dialect distance measurements using Norwegian dialect data. *Journal of Language Variation and Change 16*, 3 (Oct. 2004), 189–207.

[37] Gülcü, C., and Tsudik, G. Mixing e-mail with babel. In *proc. of the Network and Distributed Security Symposium (NDSS'96)* (Feb. 1996), pp. 2–16.

[38] Gusfield, D. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, 1997.

[39] Herrmann, D., Wendolsky, R., and Federrath, H. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naive-bayes classifier. In *proc. of 2009 ACM workshop on Cloud computing security (CCSW'09)* (Nov. 2009).

[40] Hintz, A. Fingerprinting websites using traffic analysis. In *proc. of Privacy Enhancing Technologies workshop (PET'02)* (Apr. 2002), pp. 229–233.

[41] Hollenbeck, S. Transport layer security protocol compression methods. URL *www.ietf.org/rfc/rfc3749.txt*, May 2004.

[42] Hopper, N., Vasserman, E. Y., and Chan-Tin, E. How much anonymity does network latency leak? In *proc. of 14th ACM conference on Computer and communications security (CCS'07)* (Oct. 2007).

[43] Kazaa. `http://www.kazaa.com`.

[44] Kesdogan, D., Agrawal, D., and Penz, S. Limits of anonymity in open environment. In *proc. of information hiding workshop (IH'02)* (Oct. 2002).

[45] King, J., Lakkaraju, K., and Slagell, A. A taxonomy and adversarial model for attacks against network log anonymization. In *Proceedings of the ACM Symposium on Applied Computing (SAC'09)* (Mar. 2009), pp. 1286–1293.

[46] Kiyavash, N., Houmansadr, A., and Borisov, N. Multi-flow attacks against network flow watermarking schemes. In *proc. of 17th Usenix conference on Security symposium (SEC'08)* (Jul. 2008).

[47] Kohler, E. ipsumdump. URL *www.cs.ucla.edu/ kohler/ipsumdump.*

[48] Koukis, D., Antonatos, S., Antoniades, D., Markatos, E., and Trimintzios, P. A generic anonymization framework for network traffic. In *Proceedings of the IEEE International Conference on Communications (ICC'06)* (Jun. 2006).

[49] Li, J., Sung, M., Xu, J., and Li, L. Large-scale IP traceback in high-speed Internet: Practical techniques and theoretical foundation. In *proc. of 2004 IEEE Symposium on Security and Privacy (SP'04)* (May 2004).

[50] Liberatore, M., and Levine, B. N. Inferring the source of encrypted HTTP connections. In *proc. of 13th ACM conference on Computer and Communications Security (CCS'06)* (Oct. 2006), pp. 255–263.

[51] Mankin, A., Massey, D., Wu, C.-L., Wu, S., and Zhang, L. On design and evaluation of intention-driven ICMP traceback. In *proc. of IEEE Computer Communications and Networks* (Oct. 2001).

[52] Mathewson, N., and Dingledine, R. Practical traffic analysis: extending and resisting statistical disclosure. In *proc. of Privacy Enhancing Technologies workshop (PET'04)* (May. 2004).

[53] Minshall, G. Tcpdpriv. URL *ita.ee.lbl.gov/html/contrib/tcpdpriv.html*.

[54] mixmaster. URL*http://mixmaster.sourceforge.net*.

[55] Murdoch, S. J. Hot or not: revealing hidden services by their clock skew. In *proc. of 13th ACM conference on Computer and communications security (CCS'06)* (Oct. 2006).

[56] Murdoch, S. J., and Danezis, G. Low-cost traffic analysis of tor. In *proc. of 2005 IEEE Symposium on Security and Privacy (SP'05)* (May. 2005).

[57] Navarro, G. A guided tour to approximate string matching. *Journal of ACM Computing Surveys 33*, 1 (2001), 31–88.

[58] Needleman, S. B., and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology 48* (1970), 443–453.

[59] Norvig, P. How to write a spelling corrector. Available at *www.norvig.com*.

[60] OpenVPN. `http://openvpn.net`.

[61] Oram, A., Ed. *Peer-to-peer: Harnessing the Benefits of a Disruptive Technology.* OReilly & Associates, Mar. 2001, ch. 7, pp. 89–93.

[62] Pang, R., Allman, M., Paxson, V., and Lee, J. The devil and packet trace anonymization. *SIGCOMM Computer Communication Review 36*, 1 (Jan. 2006), 29–38.

[63] Park, K., and Lee, H. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. In *proc. of annual conference of ACM Special Interest Group on Data Communication (SIGCOMM'01)* (Aug. 2001), pp. 15–26.

[64] Pfitzmann, A., Pfitzmann, B., and M.Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *proc. of GI/ITG Conference on Communication in Distributed Systems* (Feb. 1991), pp. 451–463.

[65] Plonka, D. ip2anonip. URL *dave.plonka.us/ip2anonip.*

[66] Rennhard, M., and Plattner, B. Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection.

[67] Ribeiro, B., Chen, W., Miklau, G., and Towsley, D. Analyzing privacy in enterprise packet trace anonymization.

[68] Saponas, T. S., Lester, J., Hartung, C., and Agarwal, S. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *proc. of 16th USENIX Security Symposium (SS'07)* (Aug. 2007), pp. 55–70.

[69] Savage, S., Wetherall, D., Karlin, A., and Anderson, T. Practical network support for IP traceback. In *proc. of annual conference of ACM Special Interest Group on Data Communication (SIGCOMM'00)* (Aug. 2000).

[70] Serjantov, A., and Sewell, P. Passive attack analysis for connection-based anonymity systems. In *proc. of 8th European Symposium on Research in Computer Security (ESORICS'03)* (Oct. 2003).

[71] Slagell, A., Wang, J., and Yurcik, W. Network log anonymization: application of Crypto-PAn to cisco netflows. In *NSF/AFRL workshop on secure knowledge management (SKM'04)* (2004).

[72] Slagell, A., and Yurcik, W. Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization. In *Proceedings of the workshop on the Security and Privacy for Emerging Areas in Communication Networks* (Sep. 2005), pp. 80–89.

[73] Snoeren, A., C.Partridge, Sanchez, L., Jones, C., F.Tchakountio, Kent, S., and Strayer, W. Hash-based IP traceback. In *proc. of annual conference of ACM Special Interest Group on Data Communication (SIGCOMM'01)* (Aug. 2001).

[74] Song, D. X., and Perrig, A. Advanced and authenticated marking schemes for IP traceback. In *proc. of the 20th IEEE International Conference on Computer Communications (INFOCOM01)* (Apr. 2001), pp. 878–886.

[75] Song, D. X., Wagner, D., and Tian, X. Timing analysis of keystrokes and timing attacks on SSH. In *proc. of 10th USENIX Security Symposium (SS'01)* (Aug. 2001).

[76] Srivatsa, M., Liu, L., and Iyengar, A. Preserving caller anonymity in voice-over-IP networks. In *proc. of 2008 IEEE symposium on security and privacy (SP'08)* (May. 2008).

[77] Stoica, I., and Zhang, H. Providing guaranteed services without per flow management. In *proc. of annual conference of ACM Special Interest Group on Data Communication (SIGCOMM'99)* (Aug. 1999).

[78] Sun, Q., Simon, D. R., Wang, Y.-M., Russell, W., Padmanabhan, V. N., and Qiu, L. Statistical identification of encrypted Web browsing traffic. In *proc. of IEEE Symposium on Security and Privacy (S&P'02)* (Mar. 2002), pp. 19–30.

[79] TCPDump. http://www.tcpdump.org.

[80] the Global Internet Telephony Company, S. http://www.skype.org.

[81] Tor. http://www.torproject.org.

[82] Trappe, W., Wu, M., Wang, Z. J., and Liu, K. J. R. Anti-collusion fingerprinting for multimedia. *IEEE Transactions on Signal Processing 51*, 4 (Apr. 2003), 1069–1087.

[83] University, G. T. Cryptography-based prefix-preserving anonymization. URL *www.cc.gatech.edu/computing/Telecomm/cryptopan.*

[84] Wagner, R. A., and Fischer, M. J. The string-to-string correction problem. *Journal of ACM (JACM) 21*, 1 (1974), 168–173.

[85] Waldvogel, M. GOSSIB vs. IP traceback rumors. In *proc. of Annual Computer Security Applications Conference (ACSAC'02)* (Dec. 2002).

[86] Wang, M.-H., and Shmatikov, V. Timing analysis in low-resource Mix networks: attacks and defenses. In *proc. of 11th European Symposium on Research in Computer Security (ESORICS'06)* (Sep. 2006).

[87] Wang, X., Chen, S., and Jajodia, S. Tracking anonymous peer-to-peer VoIP calls on the Internet. In *proc. of 12th ACM conference on Computer and Communications Security (CCS'05)* (Nov. 2005).

[88] Wei, J., and Xu, C.-Z. sMonitor: A non-intrusive client-perceived end-to-end performance monitor of secured internet services. In *proc. of USENIX Annual Technical Conference (Tech'06)* (Jun. 2006), pp. 243–148.

[89] Wilson, C. Who checks the spell-checkers? URL *http://www.slate.com/id/2206973/pagenum/all/*, Dec. 2008.

[90] Wright, C. V., Coull, S. E., and Monrose, F. Traffic morphing: An efficient defense against statistical traffic analysis. In *proc. of 16th Annual Network & Distributed System Security Symposium (NDSS'09)* (Feb. 2009).

[91] Wright, C. V., Monrose, F., and Masson, G. M. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research 7* (Dec. 2006), 2745–2769.

[92] Xu, J., Fan, J., Ammar, M., and Moon, S. B. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptograph-based scheme. In *IEEE international conference on network protocols (ICNP'02)* (2002).

[93] Yaar, A., Perrig, A., and Song, D. Fit: Fast Internet traceback. In *proc. of the 24th IEEE International Conference on Computer Communications (INFOCOM'05)* (Mar. 2005), pp. 1395–1406.

[94] Yu, W., Fu, X., Graham, S., Xuan, D., and Zhao, W. Dsss-based flow marking technique for invisible traceback. In *proc. of 2007 IEEE symposium on Security and Privacy (SP'07)* (May. 2007).

[95] Zhu, Y., Fu, X., Graham, B., Bettati, R., and Zhao, W. on flow correlation attacks and countermeasures in mix networks. In *proc. of Privacy Enhancing Technologies workshop (PET'04)* (May. 2004).

Appendix A

Primitives for Similarity Comparison

Here we list the similarity comparison primitives mentioned in the body of this thesis, with explanations on applying them to measure the distance between website fingerprints. Please note that there are other similarity comparison primitives possible, e.g. support vector machine.

A.1 $L1$ Distance

$L1$ distance is also known as absolute value distance, rectilinear distance or city block (Manhattan) distance. Instead of the usual Euclidean distance, $L1$ distance between two vectors in an n-dimensional real vector space with fixed Cartesian coordinate system, is measured as the sum of the lengths of projections of the line segment between the points onto the coordinate axes.

$$L1(p,q) = \sum_{k=1}^{n} |p_k - q_k| \quad ,$$

where $p(p_1, p_2, ..., p_n)$ and $q(q_1, q_2, ..., q_n)$ are vectors, and $|d|$ denotes the absolute value of d.

$L1$ distance can be applied onto modeling the distance between points in a city road grid, or modeling the distance between squares on the chessboard for rooks in chess.

In the application of website fingerprinting, we can use $L1$ distance to measure the dissimilarity between packet size distributions of two website fingerprints. The packet size distributions are represented by vectors $p(p_1, p_2, ..., p_n)$ and $q(q_1, q_2, ..., q_n)$, where p_i or q_i represents the probability of a packet having size i. The $L1$ distance between the packet size distributions is then computed as the sum of absolute differences between their corresponding probabilities in all possible packet sizes.

A.2 Jaccard's Coefficient

Jaccard's coefficient measures the similarity of two sets, by evaluating the ratio of common elements to their union of elements. Jaccard's coefficient $Jac(X, Y)$ is computed as

$$Jac(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad ,$$

where X and Y are the sets to compare, and $|A|$ denotes the size of set A. Note that an input set can be a multiset that contains several elements of the same value, then the set union and intersection correspond to multisets.

In the application of website fingerprinting, Jaccard's coefficient can be used to compare two packet size distributions, or to measure the similarity between the distinct packet sizes of two HTTP streams. For the comparison in packet size distributions, the input sets are multisets of packet sizes. For the comparison in distinct packet sizes, the elements in each input set is unique.

A.3 Naive Bayes Classifier

Naive Bayes classifier assumes independence between all attributes, and estimates the probability of a set of values $A = \{a_1, ..., a_n\}$ belonging to a particular class C_i as:

$$p(C_i|A) \propto p(C_i) \prod_{j=1}^{n} (p(a_j|C_i))$$

In the application of website fingerprinting, we can employ naive Bayes classifier to classify an HTTP stream by its packet size distribution. A contains all the distinct packet sizes appearing in the stream, and $p(a_j|C_i)$ is the probability that a packet of website C_i has size a_j.

A.4 Edit Distance

Edit distance measures the minimal number of edit operations, such as insert, delete or substitute, to make two strings identical. There are well known computation algorithms for edit distance [84]. Essentially, they perform search considering the allowed edit operations.

In the application of website fingerprinting, we adopt and customize edit distance to measure the similarity between two sequences of packet sizes or web object sizes. The advantage of using edit distance over Jaccard's coefficient or naive Bayes classifier is that edit distance can consider the ordering information, in addition to the sizes information of packet streams.

Edit distance has been applied in a wide range of applications, such as spell checker in Google [59] and Microsoft Word [89], identifying plagiarism, comparing DNA sequences [38, 58], conducting fuzzy search in EXCEL [57, 3] and evaluating dialect distances [36]. We are the first in applying it to match features of network traffic. Edit distance is appropriate for the application because of the correspondence between edit operations and network feature values, since packets may be lost, reordered or retransmitted, and web objects may be added, removed or replaced.

The pseudocodes of Levenshtein Distance is shown below for reference.

```
LevenshteinDistance(sequence1, sequence2) {

  for i = 1 to len_sequence1
    for j = 1 to len_sequence2
      d[i,j] = minimum (
        d[i-1,j  ] + cost_delete,
        d[i  ,j-1] + cost_insert,
        d[i-1,j-1] + cost_substitute)
    end for
  end for

  return d[len_sequence1, len_sequence2]
}
```

Appendix B

Pseudocode of Edit Distance Extended with Split and Merge

Recall that Levenshtein distance measures edit distance with operations of insert, delete and substitute a character. We extend Levenshtein distance to allow two more operations, namely, split and merge, for traffic fingerprint comparisons.

We apply Dijkstra's shortest path algorithm to search for the edit distance between two sequences. It starts traversing from their first elements till it reaches end of the sequences. It updates the reduced distance whenever it finds a shorter path to some intermediate elements. Effectively, each distance d[i,j] is evaluated as

```
d[i,j] = minimum (
    d[i-1,j  ] + cost_delete,
    d[i  ,j-1] + cost_insert,
    d[i-1,j-1] + cost_substitute,
    d[i-1,j-2] + cost_split,
    d[i-2,j-1] + cost_merge,
    d[i-2,j-2] + cost_merge_split,
    d[i-1,j-3] + cost_double_split,
    d[i-3,j-1] + cost_double_merge,
    d[i-3,j-2] + cost_split_double_merge,
    d[i-2,j-3] + cost_merge_double_split,
    d[i-3,j-3] + cost_double_merge_split),
```

where i and j are indices of elements in the comparing sequences. It takes into consideration the costs of possible delete, insert, substitute, as well as split and merge operations.

In the demonstrating pseudocodes, we cater for a search space where a group of packets can be split up to three chunks, or merged *vice versa.* We assume the cost of substitution is higher than merge and then split, since packets are more likely re-grouped due to delay

```
ExtendedEditDistance(sequence1, sequence2) {

  d[1,1] = minimum (
    d[0,1] + cost_delete,
    d[1,0] + cost_insert,
    d[0,0] + cost_substititue)

  PriorityQueue Q
  Q.insert(d[i=1,j=1])
  while (Q.head != [i=len_sequence1, j=len_sequence2])
    d[i+1,j  ] = minimum (d[i+1,j  ], d[i,j] + cost_delete)
    d[i  ,j+1] = minimum (d[i  ,j+1], d[i,j] + cost_insert)
    d[i+1,j+1] = minimum (d[i+1,j+1], d[i,j] + cost_substitute)
    d[i+2,j+1] = minimum (d[i+2,j+1], d[i,j] + cost_merge)
    d[i+1,j+2] = minimum (d[i+1,j+2], d[i,j] + cost_split)
    d[i+2,j+2] = minimum (d[i+2,j+2], d[i,j] + cost_split_merge)
    d[i+3,j+1] = minimum (d[i+3,j+1], d[i,j] + cost_double_merge)
    d[i+1,j+3] = minimum (d[i+1,j+3], d[i,j] + cost_double_split)
    d[i+3,j+2] = minimum (d[i+3,j+2], d[i,j] + cost_split_double_merge)
    d[i+2,j+3] = minimum (d[i+2,j+3], d[i,j] + cost_merge_double_split)
    d[i+3,j+3] = minimum (d[i+3,j+3], d[i,j] + cost_double_split_merge)
    remove Q.head
    Q.insert d[i+1,j  ], d[i  ,j+1], d[i+1,j+1],
             d[i+2,j+1], d[i+1,j+2], d[i+2,j+2],
             d[i+3,j+1], d[i+1,j+3], d[i+3,j+2], d[i+2,j+3], d[i+3,j+3]
  end while

  return d[len_sequence1, len_sequence2]
}
```

variations than being substituted. So the search in the pseudocode always prefers merge and split than substitution.