FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Fingerprinting HTTP/2 Web Pages

Francisco Pedro Chorão Estêvão

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Professor Ricardo Santos Morla

July 16, 2017

Resumo

Hoje em dia a Internet está em todo o lado e é usada a toda a hora. É, portanto, de grande importância garantir que as pessoas a usem de forma segura, especialmente quando envolve a introdução de informação confidencial, como dados pessoais ou financeiros. Existe já uma grande variedade de mecanismos que visam assegurar o máximo de privacidade e segurança possível dos utilizadores quando acedem à Internet. No entanto, ainda há muitas técnicas que permitem não só a invasão de sistemas de rede, mas também a extração passiva de informação sobre a atividade dos utilizadores na Internet. Uma dessas técnicas são os ataques fingerprinting, que usam tamanhos de resposta HTTP como informação side-channel.

Com a implementação do HTTP/2, já não é trivial obter informações sobre os tamanhos de elementos web capturando tráfego da Internet. Isto acontece porque o HTTP/2 introduz pipeline e multiplexagem de respostas, e, portanto, acredita-se que isso dificulta que atacantes determinem tamanhos de respostas. No entanto, este assunto ainda não foi devidamente estudado.

Esta dissertação visa compreender as limitações que o HTTP/2 impõe nos ataques de fingerprinting a websites. Realizamos uma experiência de ataque fingerprinting na qual avaliamos a influência do HTTP/2 em websites. Primeiro, pelo ponto de vista de ground-truth, em que recolhemos tráfego desencriptado de websites, selecionamos as features que melhor contribuem para a sua identificação, e usamos classificadores e machine learning para determinar o desempenho do ataque. Depois fazemos o mesmo numa perspectiva de ataque. Recolhemos o tráfego encriptado e fazemos a sua análise usando estimativas de tamanhos de resposta HTTP feitas em trabalhos anteriores. Finalmente, comparamos os resultados de ambas as abordagens.

Os resultados mostram que ainda é possível fazer, com sucesso, fingerprinting a websites que utilizam HTTP/2. No entanto, o uso de pipelining e multiplexagem tem um impacto no desempenho do fingerprinting e dos algoritmos de classificação.

Abstract

In today's world, the Internet is everywhere and is used all the time. It is then of great importance to make sure that people are using the Internet in a secure way, specially when it involves the sharing of sensitive information, such as personal or financial data. There is already a great variety of mechanisms that aim at ensuring the maximum possible privacy and security of users when accessing the Internet. However, there are still many techniques that allow not only the invasion of network systems but also the passive extraction of information of user activity on the Internet. One of these techniques are the fingerprinting attacks that use side-channel information consisting of HTTP response sizes.

With the deployment of HTTP/2 it is no longer straightforward to obtain information about sizes of web elements by capturing Internet traffic packets. This happens because HTTP/2 introduces pipelining and response multiplexing, and so it is generally understood that this makes it difficult for eavesdroppers to determine response sizes. However this has not been adequately studied yet.

This dissertation aims at understanding the limitations that HTTP/2 imposes in website fingerprinting. We conduct a fingerprinting attack experiment in which we evaluate the influence of HTTP/2 in websites. Firstly, we take the ground-truth perspective in which we collect and decrypt website traffic, select features that best help identifying websites, and use machine learning and classification algorithms to determine the performance of the fingerprinting. Then we do the same for an attacking perspective. We collect encrypted traffic and we analyze it by using estimates of HTTP response sizes done in prior work. Finally, we compare results from both approaches.

Results show that it is still possible to successfully fingerprint websites which use HTTP/2. However, the use of pipelining and multiplexing in websites does have an impact on how well fingerprinting and classification algorithms perform.

Acknowledgements

Firstly, I thank my supervisor Professor Ricardo Morla, who was the one person that was always available and present to help me in the most technical parts. He was the person who was more directly related to this work, and without his inputs, advice and guidance, I would not be able to finish it.

I thank my parents and sisters, who have always unconditionally supported me throughout my academic career. I appreciate the monetary sacrifices they had to make for me.

My friends from FEUP. It was a pleasure to share working hours with you and thank you for giving me a hand whenever I needed.

My friends outside FEUP. Always ready to crack open a cold one during the weekends. Thank you all for the dinners, the moments and for clearing my head when I was struggling.

Francisco Estêvão

"You affect the world by what you browse."

Tim Berners-Lee

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AES	Advanced Encryption Standard
API	Application Programming Interface
B	Byte(s)
CBC	Cipher Block Chaining
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
DB	Database
DOM	Document Object Model
DoS	Denial of Service
GT	Ground-Truth
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over TLS
IP	Internet Protocol
JAP	Java Anon Proxy
JS	JavaScript
JSON	JavaScript Object Notation
JSSE	Java Secure Socket Extension
PDML	Packet Description Markup Language
RC4	Rivest Cipher 4
SChannel	Secure Channel
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TOR	The Onion Router
TLS	Transport Layer Security
URL	Uniform Resource Locator
VPN	Virtual Private Network
WWW	World Wide Web
XML	Extensible Markup Language
XSS	Cross-site Scripting

Chapter 1

Introduction

This chapter presents an overview of the background to which this dissertation belongs. We provide the general objectives as well as the motivations behind this work.

1.1 Context

Nowadays, the usage of Internet is everywhere. As of 2016 it was estimated that 81% of people in the developed regions of the world used Internet[1]. In an era when access to the Internet is done on a regular basis, it is of great importance to guarantee that people are doing it in a secure way, specially when it involves the sharing of sensitive information, such as personal or financial data.

Most of the services and products that exist have web applications associated with them, which usually follow the client-server model. Mobile or desktop, browser or app, users will lose control of much of their information, which is inevitably exposed on the network and it is impossible to fully guarantee its complete security and privacy, independently of how secure the system is and how well it is implemented. There will always be flaws and vulnerabilities.

With regard to the development of the WWW, we are at a stage in which there is already a wide range of mechanisms that aim at ensuring the maximum possible privacy and security of users, as is the case of encryption and access control. However, there are still numerous techniques that allow not only the invasion of network systems, but also the extraction of information for virtually every type of Internet activity.

One of these techniques are the side-channel attacks. Side-channel attacks are based on side-channel information, which is information that can be retrieved from the device responsible for encrypting content. However, this information is neither *plaintext* information nor the one resulting from the encryption process, called the *ciphertext*. Instead, it consists in information that the encryption device leaks. Examples include the varying power consumption during the encryption computation or timing information retrieved by observing responses provided by the system under attack.

[1] ICT Facts and Figures 2016, 2016. URL: http://www.itu.int:80/en/ITU-D/ Statistics/Pages/facts/default.aspx

Timing attacks are an example of side-channel attacks [1]. In these, attackers are able to extract information from a security system by making statistical analysis of the time the system takes to respond to queries [2].

Other side-channel attacks make use of the size of web objects. In this type of attack, the statistical analysis usually culminates in the defining of a *fingerprint* for each web application. This is the reason why this type of attacks can also be called *fingerprinting* attacks, mainly because what the attacker does is collect information about a web application and build a profile (the *fingerprint* of that web application). This means that every specific characteristic or behavior of a web application translates to the way the packets generated by it are organized, which is such that it can be identified afterwards. Stored profiles can then be compared with captured data from the victim [3]. Ultimately, this will enable the identification of the website the victim is accessing.

Encryption, and TLS in specific, encrypts application layer data. This means that some meta data of the communication may still be available. Source and destination IP addresses, hostnames and payload sizes are not encrypted. Thus, it is possible to directly tell the IP address of the destination server by a simple IP database lookup. However, this is a simplistic method, because there are several actions a user can do inside a website that has the same IP address. More importantly, the IP address of the server with which the browser is communicating may not reveal everything about the website's identity. This may happen due to the use of proxies or CDNs, which have a growing amount of traffic passing through them. These may share IP addresses between applications and thus it is not sufficient to inspect IP addresses to identify a website.

1.2 Motivation

Side-channel analysis techniques are something to be concerned about. These attacks can be generated and implemented quickly and using limited resources, which means that it is within the reach of virtually anyone with minimal knowledge of computers and networking. Furthermore, the time one has to spend in order to obtain valuable information from a side-channel attack only depends on the amount of websites to analyze, although it may depend on the type of attack and on the computational resources available.

In May 2015[2] a new version of the network protocol used on the WWW was released: HTTP/2. In October 2016, 10% of the largest 10 million websites had implemented and were using HTTP/2[3], including popular websites such as *Google*, *YouTube*, *Facebook* and *Twitter*. Furthermore, the trend is that these values will continue to grow. In only 8 months, the value has grown 50%, with almost 15% of the websites using this protocol. Figure 1.1 shows the recent historical trend of this statistic.

[2] Internet Engineering Task Force (IETF). Hypertext Transfer Protocol Version 2 (HTTP/2). URL: https://tools.ietf.org/html/rfc7540

[3] Web Technology Surveys. Usage Statistics of HTTP/2 for Websites, November 2016. URL: https://w3techs.com/technologies/details/ce-http2/all/all

[4] Web Technology Surveys. Usage Statistics of HTTP/2 for Websites, June 2017. URL: https://w3techs.com/technologies/details/ce-http2/all/all

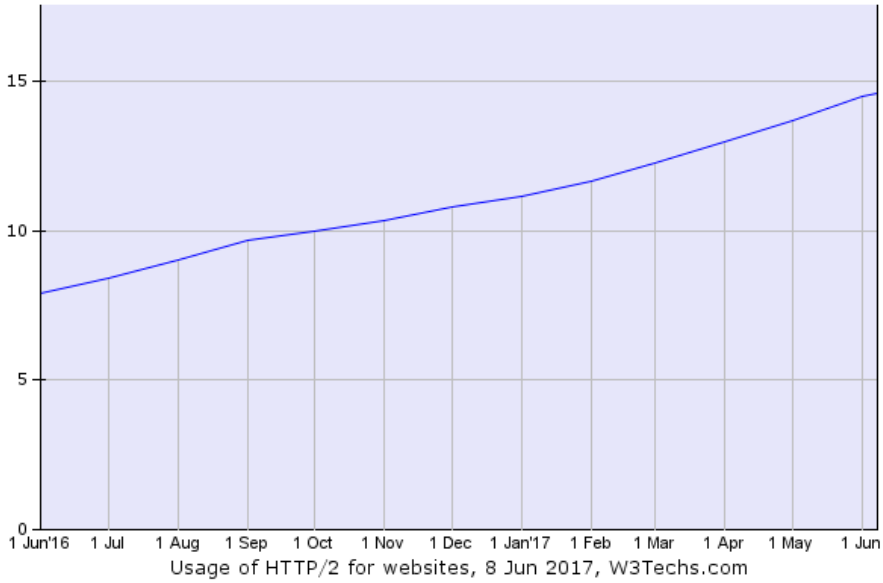

Figure 1.1: Usage of HTTP/2 for websites (%)[4]

With this new version, it is no longer straightforward to obtain information about sizes of web elements by capturing Internet traffic packets. This happens because HTTP/2 enables the pipelining of requests, as well as the multiplexing of responses over a single TCP connection, which makes it difficult to identify web elements and their size [4]. Thus, attacks that use element sizes of a web application as the main source of information may no longer work.

HTTP/2 has generally a better performance than its previous versions and so it is considered to be superior. It is then to be expected the rapid growth of the usage of this protocol and therefore this study gains another importance and dimension.

1.3 Objectives

The goal of this study is to develop an approach to identify a website by observing the application layer information. If this method is proven to be satisfactory, it can be the basis to the identification of several actions a user can take in a website, such as publishing a post on Facebook, loading a video on Youtube or messaging via Messenger [5].

This dissertation aims at characterizing and analyzing until what extent it is possible to fingerprint web applications that employ HTTP/2. The website fingerprinting done in this work seeks to identify web pages by passively observing the communication traffic of the application layer (HTTP and TLS). The traffic contains packet lengths, information about the direction of each packet, and application-specific features that could individually identify the web page. This website fingerprinting attack uses machine learning and classification algorithms. These techniques extract the most important information from the communication traffic and try to predict to which website does each traffic pattern belongs to.

In addition to using the classification algorithms, we systematically evaluate the performance of different sets of information we feed the classification algorithms. This allows us to understand the relevance of the different information we extract from the traffic.

The main contributions of this thesis are the following:

- Selection of the best features for the identification of a website
- Try to understand the limitations of fingerprinting in HTTP/2 web applications, using the estimation of HTTP response sizes described in [4]

1.4 Dissertation Outline

In addition to the Introduction, this dissertation contains more 6 chapters. In chapter 2, fundamental concepts are explained, the state of the art is described and related works are presented.

Chapter 3 shows the approach used to do this dissertation, and explains it step by step: how the captures were done (3.1.2), how the data was extracted and processed (3.1.3), how the classification was done (3.1.4) and what metrics were used to analyze the performance of the classification algorithm (3.1.5). Furthermore, this chapter goes into detail about the data set used to make the study of this dissertation, presents statistical measures about it and explains them.

Chapter 4 enumerates the features chosen to analyze website traffic. These are the key to obtain better results. We also present the first results for these features. Here, there is complete access to every traffic detail. It is the result of the first approach of this study. On the other hand, chapter 5 presents the results for the study targeting web applications when there is encryption.

Finally, chapter 6 summarizes and debates the results as well as it proposes further studies that can be made in order to improve this study.

Chapter 2

Literature Review

In this chapter, the basic concepts necessary to understand this work are described. At the same time there are references and explanations of what has already been studied in this field, namely, the state of the art.

2.1 Encryption and Privacy

In general, the field of cryptography studies ways of ensuring secure communications while there are parties that try to access the information shared in that communication. A communication that enjoys the quality of only allowing authorized parties to access its information is considered a communication that possesses confidentiality.

Encryption is the process of converting ordinary and raw information (*plaintext*) into unintelligible information (*ciphertext*) through various transformations of the former [6]. Encryption began thousands of years ago. From the ancient Greeks and Spartans, who used an encryption tool called scytale [7], to the 20th century rotor machine *Enigma*, used in the First World War, until modern ciphers like AES, there has always existed the need of guaranteeing confidentiality in communications.

Where there is an intention of making something so that it cannot be accessed by everyone, there will inevitably be people pursuing ways of breaking those techniques. That is why cryptography has to be constantly improving and finding new algorithms and methods, which are better and more secure than the previous ones. Technology is constantly evolving, but it does not only improve security methods, but also methods to break them. Consequently the standards of every age in history are always eventually broken.

The focus of this thesis is privacy of users accessing web applications. There are several ways of attacking these applications, such as *XSS* or *SQL injection*. These attacks focus on the flaws of how the application itself is implemented and consist in actively breaking in the websites. Encryption does not play a role in defending against these. Rather, encryption prevents the interception of information and in the case of HTTP, the protocol responsible for that is *TLS*[1]. SSL was a previous

[1]Network Working Group. HTTP Over TLS. URL: https://tools.ietf.org/html/rfc2818

version of TLS and this protocol is sometimes referred to as TLS/SSL. The implementation of this protocol in web applications usually culminates in the usage of the *HTTPS* protocol, which consists in a communication over HTTP but having the connection encrypted by TLS. Thus, the security of HTTPS is the one of TLS [2].

The primary goal of this protocol is to provide privacy and data integrity between two communicating applications. Ideally, communications secured by this protocol have the guarantee that the connection is private and reliable [3], and that is why client-server applications use it to prevent eavesdropping and tampering.

There are many different TLS implementations, such as *OpenSSL*, *JSSE* (*Oracle*), *s2n* (*Amazon*), *SChannel* (*Windows*) or *Secure Transport* (*Apple*). Every TLS implementation varies in some aspects, such as:

- The version of the protocols. For example, OpenSSL supports DTLS 1.2 [4], but JSSE does not;
- Encryption algorithms. Some use block ciphers while others use stream ciphers;
- Data Integrity algorithms;
- Key exchange algorithms.

Key exchange algorithms are related to *session keys* (used in this work and explained in section 3.1.2), which are *symmetric keys* used for encrypting messages in a communication session, as it happens in web browser sessions. In turn, symmetric keys are the cryptographic keys used in symmetric-key algorithms, which use the same keys for both encryption of the plaintext and the decryption of the ciphertext.

Essentially, even with encryption some information can leak and this is due to the way different software implements TLS. Over the past there have been several attacks on TLS, exploiting the specifics of the protocol. Some of them attempt to discard TLS by modifying unencrypted protocols that request the use of TLS, which are called *SSL Stripping attacks*. Other attacks leverage cryptographic weaknesses, such as the *BEAST attack*, which exploits the stream cipher CBC, or attacks on the stream cipher RC4. There are some others that even make use of the server's private key so that they are able to decrypt sessions [8].

To protect users and to ensure their privacy, using encryption is a central issue. However, there are many aspects of privacy and not all related to cryptographic protocols. There are numerous examples of application service providers which use private data for purposes other than what the application needs. Search engines like *Google* or *Bing* possess significant amount of personal information about users' web browsing history, and many general retail websites track

[2] E. Rescorla. HTTP Over TLS. URL: https://tools.ietf.org/html/rfc2818

[3] Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. URL: https: //tools.ietf.org/html/rfc5246

[4] OpenSSL: OpenSSL 1.0.2 Notes, September 2014. URL: https://web.archive.org/web/20140904045720/http://www.openssl.org/news/openssl-1.0. 2-notes.html

visitor's shopping habits. The possession of this type of information without the individual's explicit concern is a violation of privacy. Some web application providers share this information with advertisers or other third parties [5] and even sell this information [6].

Some solutions have been proposed for the threat that exists to online privacy. Tools like *firewalls*, anonymizers (like *TOR*) and *VPN*s are examples of technology-enabled solutions for this problem. Another example is the social networking service that preserves user's privacy in social network applications, proposed in [9]. There are also regulation-enabled solutions like mandatory policies that providers have to comply with [10]. A summary of the existent solutions is shown in Figure 2.1. This work is related to encrypted-based solutions (bottom right of the figure), by testing until what extent they are reliable.

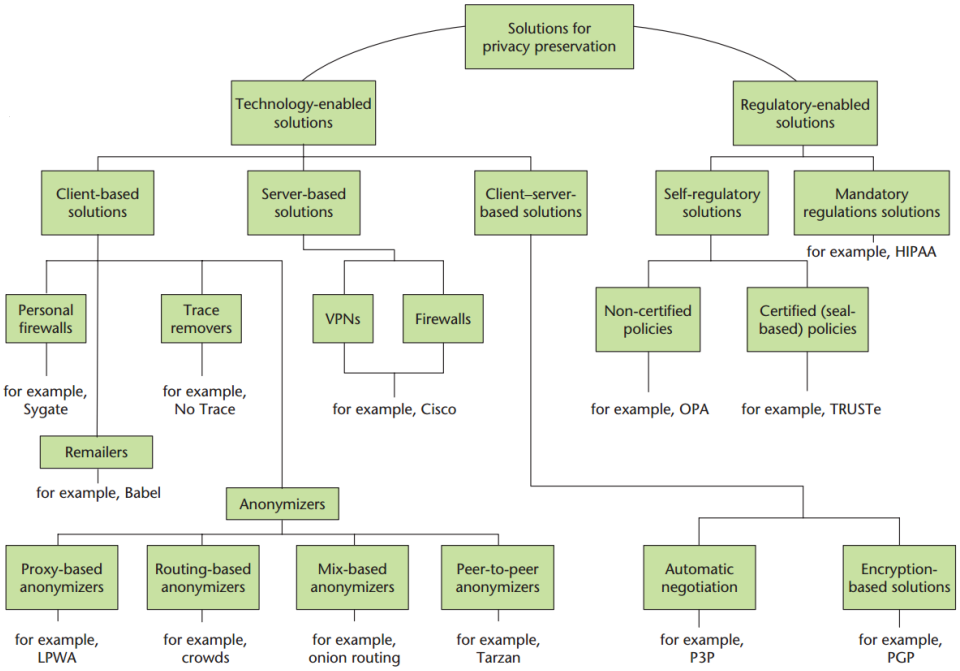

Figure 2.1: Solutions for privacy preservation in the Web [10]

2.2 Web Applications

Web applications are client-server software applications in which the client accesses the server side through a web browser. These applications can be of a wide range of types. There are web applications for webmail (web-based email), wiki, instant messaging services, social networks among many others.

[5] Tracking the Trackers: Where Everybody Knows Your Username. URL: /blog/2011/10/tracking-trackers-where-everybody-knows-your-username

[6] Goldman. Your phone company is selling your personal data. URL: http://money.cnn.com/2011/11/01/technology/verizon_att_sprint_tmobile_privacy/index.htm

Web applications are distinct from web pages in the way that the former are dynamic and are similar to a desktop software application, but instead they run in a web browser. Web pages can be dynamic or static. In static web pages the content (HTML, images, etc.) is always delivered in the same way, regardless of the visitor. On the other hand, dynamic web pages are controlled by scripting languages and have content that may change. These can be dynamic both on the server and on the client side. On the server side these are controlled by server-side scripts which rule the way the web page should behave on the client side. The content on the server is the same, but it contains dynamic code, which may serve different data to a visitor, depending for example on variables such as the geographical location of the visitor, the time of the day or even on who is accessing the web page. Web pages that are dynamic on the client-side are processed using scripting languages such as JavaScript, which determine the way the HTML is parsed into the DOM, which is the *special* language for web pages that are already loaded and ready to be *interpreted* by the browser [7].

In what regards traffic analysis, the variation of content can also be due to several other reasons such as advertisements (although these cause variation in incoming packets but not outgoing packets), and, for example, in news websites it is an inevitable event because news are constantly changing and so does the content of the website.

Fundamentally, the constant change of content in web pages handicaps the identification of websites, which is the ultimate goal of this study and is explained in section 2.4.

2.3 Side-channel Attack

As it was briefly described in 1.1, side-channel attacks are ways of collecting leaked information from a cryptosystem. However, instead of taking advantage of flaws of the cryptographic algorithms, or doing brute force attacks, side-channel attacks passively explore the implementation of the system and its weaknesses. Information resulting from this reconnaissance is called side-channel information, which consists in information which is neither the raw data (before being submitted to the cryptosystem) nor the resulting data. Thus, it is considered information that is not directly related to the data to be encrypted.

The idea that it is possible to extract information with a side-channel attack approach has existed since mid-last-century. In the example described in [11], there was a teletype machine which emanated electromagnetic signals that could be translated into the plaintext message that the machine was processing.

In 1985, Wim Van Eck published the first paper on the now called *Van Eck phreaking*, which is a form of eavesdropping in which side-band electromagnetic emissions are picked up from electronic devices that correlate to hidden signals or data for the purpose of recreating these signals or data. These emissions are present and can be captured from devices such as keyboards and printers and are used to spy on these electronic device [12].

[7] How does the Internet work - W3c Wiki. URL: https://www.w3.org/wiki/How_does_the_Internet_work

Side-channel attacks have been discussed in contexts other than encrypted communications, namely timing analysis, unintentional leaks of electrical or radio signals, power consumption, sounds and mechanical vibrations.

Regarding encrypted communications, there has been proof that timing attacks are effective in extracting private keys from web servers based on the software *OpenSSL* [13]. Further studies show that other characteristics of web applications, such as low entropy input, stateful communications and significant traffic distinctions make the side-channel leak of information a serious privacy problem [14] and something to be concerned about.

2.4 Fingerprinting

In [15] fingerprinting is described as the process of mapping an item with large data to a shorter version (the *fingerprint*), which uniquely identifies the original data. The name of this process is due to the similarity to human fingerprints, which are unique and can therefore identify an unknown person by finding a match in a database of known fingerprints [16].

A fingerprinting attack is the process of using fingerprinting techniques in order to break into computer systems or gaining access to information unduly. An example of this kind of attack is the *TCP/IP stack fingerprinting*, which consists in the process of determining the identity of an operating system of a remote host by analyzing packets from that host. There are other targets to fingerprinting attacks such as computing devices, audio and digital video.

This dissertation's approach is closer to website traffic fingerprinting, in which the attacker recognizes the traffic patterns and packets sent and received from specific web pages the victim is accessing. All this happens despite the fact that the victim is using encryption, which gives the victim a false feeling of security. The main objective of an attacker in a website fingerprinting attack is to identify the web page the user is visiting, by observing their communication traffic.

Figure 2.2 illustrates a general fingerprinting attack. The attacker first gathers information about websites. He accesses them and observes their contents, captures the traffic and stores the traffic information. Later, in order to do the attack, he passively observes and captures the traffic of his victim. This time, because it is encrypted, the attacker can't observe the contents. However, the attacker can compare it to the information he has stored and obtain valuable information in the communication of the victim.

2.5 Machine Learning

Machine learning is the science of getting computers to learn from data by finding patterns and hidden insights, all without being explicitly programmed where to find them. It is derived from the study of pattern recognition and computational learning theory in artificial intelligence. As the name implies, machine learning studies the implementations of algorithms in machines, so that they can learn and make predictions on data. These algorithms improve with experience. Machine learning can also be defined as a method of analyzing data that automates the building of models.

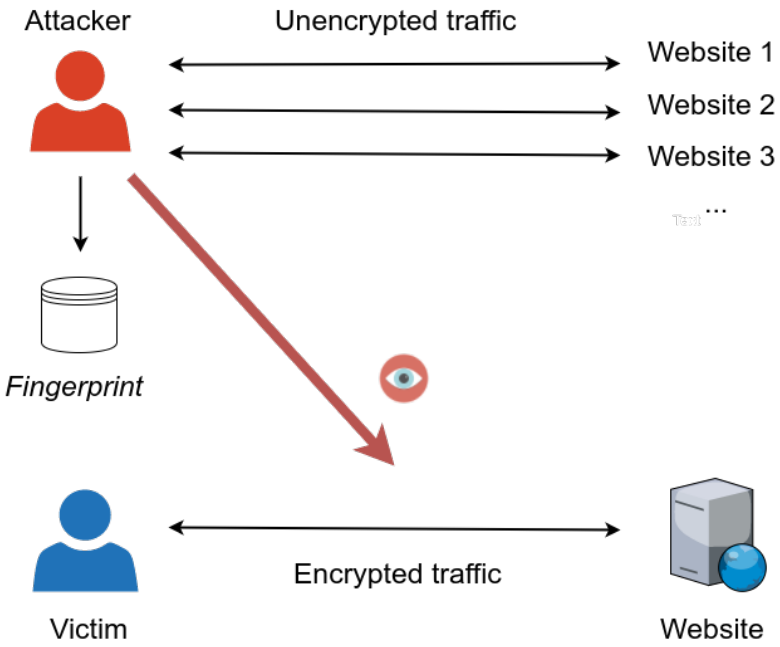

Figure 2.2: Fingerprinting

Classification is the field of machine learning that studies the prediction of the testing data set (observation), in terms of identifying to which category/label it belongs. This prediction is based on a training set of data, from which it is acknowledge to which label it belongs. Figure 2.3 illustrates in a general way how machine learning and classification algorithms work.

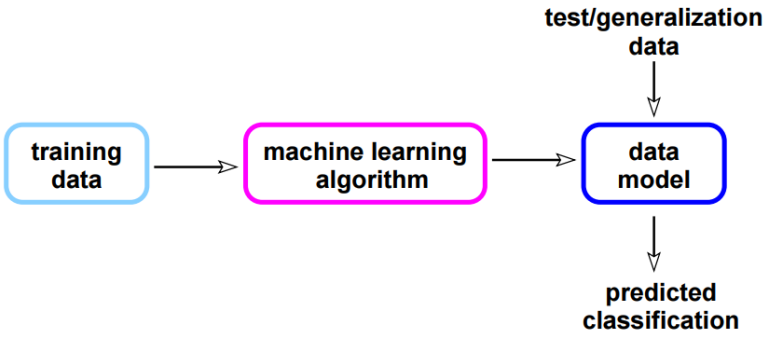

Figure 2.3: General schema for machine learning methods [17]

2.5.1 K-NN (K-Nearest Neighbors)

K-NN is one of the simplest algorithms used for classification and regression [18]. In this work, it was used for classification. Essentially, an object (testing data) is classified based on the class

to which most of its neighbours belong. The closeness of one object to its neighbors can be measured for example by their Euclidian distance. The number of neighbors to take into account is the parameter K, a positive integer. For example, if $K = 1$, it means that the object is going to be assigned to the class of its closest neighbor, if $K = 3$, the object is going to be assigned to the majority of its 3 nearest neighbors. If there is a tie in the number of neighbors, choosing one class randomly can be a method for breaking them.

Figure 2.4 shows an example of the application of the K-NN algorithm. In a case in which $K = 3$, the testing data (black dot in the middle) is going to be classified as a red triangle because the 3 nearest neighbours (inside the continuous circumference) are mostly red triangles. However, if $K = 5$, the testing data will be labelled as blue square, because its 5 nearest neighbours (inside the dashed circumference) are mostly blue squares.

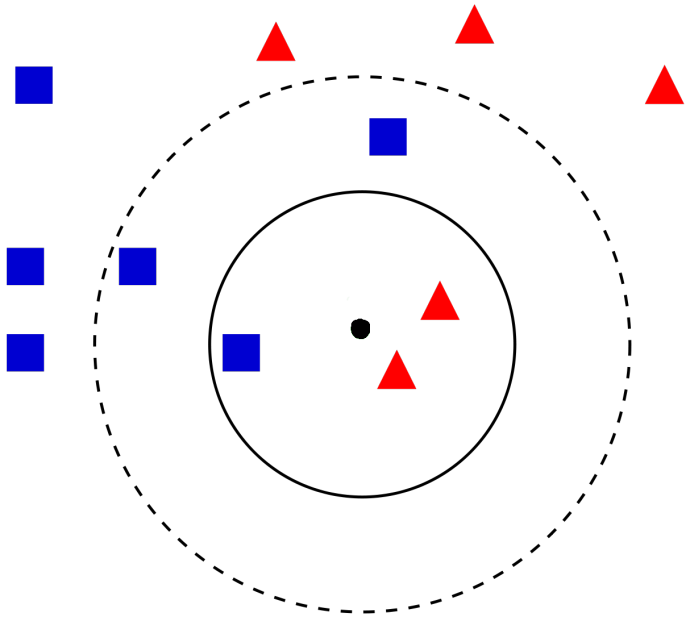

Figure 2.4: Example of K-NN Classification

The previous paragraph describes the importance of choosing the right parameters, which is the biggest challenge in any machine learning and classification algorithm. The values chosen for K in this dissertation were 1, 3 and 5. Firstly, we only chose odd values so that there were no ties while choosing classes. Then, we chose the smallest values so that the classification it was not computationally expensive.

Additional parameters can play an important role in this algorithm, such as the weight of different types of data. As it is further explained in this dissertation, in the data set, there is information which is more important than the rest, and so it deserves to have a bigger impact on the application of the algorithm.

2.6 Analysis of Related Work

2.6.1 Website Fingerprinting

One of the first uses of the term *Website Fingerprinting* was done by Hintz in [19]. However, the website fingerprinting idea had already been indicated in [20], when it was mentioned that encrypted SSL records could be identified by analyzing traffic. There have been several studies on the impact of such attacks in different situations. Dynamic web pages have proven to make fingerprinting techniques perform poorly [3]. The TOR browser has been a constant target for testing the feasibility of fingerprinting attacks, due to its anonymity based on layers of encryption. Some results show that when using this browser, the attacks have low accuracy, mainly when there are variables such as multi-tab browsing behavior, Internet connection and website variance over the time [21]. On the other hand, there have been results proving that these anonymity tools (TOR and JAP) can be correctly detected with a rate of almost 75% [22].

2.6.2 Feature Definition

Captured traffic contains every detail about the communication. However, as it was already stated, only application layer data is required for this approach and so it is necessary to define how to process that data. To do so, it is necessary to define which features of websites we are going to analyze.

In [23], the feature used was the frequency distribution of IP packet sizes, which is too general, as it is unaware of internal matters of the Privacy Enhanced Technologies (PET) targeted (VPNs, SSL proxies, OpenSSH tunnels, TOR). [3] proposes a feature termed as *Surge Period*, which is defined based on the timing of surges in traffic density or high bandwidth utilization.

This study is based on the feature definition of [22], which considers features such as the total transmitted bytes, occurring packet sizes, percentage of incoming packets and number of packets.

However, there has not been studied the influence of the number of web objects of specific sizes.

2.6.3 HTTP/2

HTTP/2 is an optimized version of HTTP, which came with the intention of enabling a faster and more efficient use of the network. It is based on Google's SPDY protocol, which already tries to solve the HTTP/1.1 latency problems through multiplexing. Besides supporting all the essential features of HTTP/1.1, HTTP/2 brings major improvements, mainly performance related. For example, in HTTP/2 each request/response exchange is associated with its own stream, which is independent from other streams and does not interfere with them, which results in the multiplexing of requests, as shown in Figure 2.5. In this protocol, flow control and prioritization are also present, which allow the multiplexing to be done in an efficient way. HTTP/2 introduces a new basic unit called *frame*, and each of these has a different purpose. Some build the basis of the

requests and responses (frames *HEADERS* and *DATA*) while others serve for other features (*SETTINGS*, *WINDOW_UPDATE*). Header compression is also offered in this version of the protocol, by compressing HTTP header fields that hold redundant data [8].

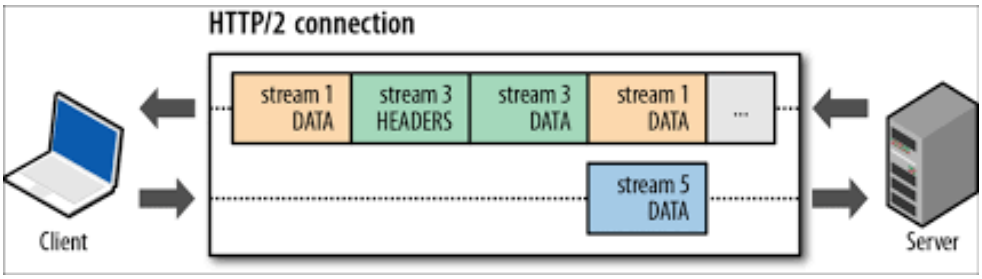

Figure 2.5: HTTP/2 multiplexing [9]

There has been great accession to HTTP/2 and most of the major browsers already support it [10] [11] [12]. HTTP/2 is specified to work with or without encryption [13][14], but in spite of this fact most web browsers only support it with encryption [15]. Besides web browser supporting it, there are also already millions of websites which use this version of HTTP [16].

With the arrival of a new version of a widely used protocol it is inevitable to test if there are no flaws or weaknesses regarding its security and performance. So far, only few experiments have been done, although there have been already some breakthroughs. [24] shows that it can be fairly easy to carry out DoS attacks by exploiting HTTP/2 packets. A measurement platform that monitors the performance in the adoption of HTTP/2 showed that 80% of websites supporting it decrease page load latency when compared to HTTP/1.1 [25]. However, there is still the urgent need for a reliable and thorough way of covering and analyzing HTTP/2 implementations.

HTTP/2 pipelining and multiplexing is assumed to prevent HTTP response size analysis attacks. This is believed to work because with request pipelining there is no need to wait for the response of an HTTP request in order to send the next request. Furthermore, response multiplexing enables servers to send consecutive responses without waiting for the first ones to be finished. However, this has not been proved yet to be an efficient way of hiding HTTP response sizes. There was made an initial study [4] on the effect of pipelining in the attempt at hiding this. The study described there makes an approach of estimating HTTP response sizes from TLS records. This estimation is a strong influence on this study and the basis of the fingerprinting of the websites.

[8] Mike Belshe, Martin Thomson, and Roberto Peon. Hypertext Transfer Protocol Version 2 (HTTP/2). URL: https://tools.ietf.org/html/rfc7540

[9] HTTP: HTTP/2 - High Performance Browser Networking (O'Reilly). URL: https://hpbn.co/http2

[10] https://blogs.msdn.microsoft.com/ie/2014/10/08/http2-the-long-awaited-sequel

[11] https://wiki.mozilla.org/Networking/http2

[12] https://www.mnot.net/blog/2014/01/04/strengthening_http_a_personal_view

[13] HTTP/2 Frequently Asked Questions. URL: https://http2.github.io/faq

[14] M. Belshe, R. Peon, and M. Thomson. Hypertext Transfer Protocol Version 2 (HTTP/2), May 2015. URL: https://http2.github.io/http2-spec

[15] Networking/http2 - MozillaWiki. URL: https://wiki.mozilla.org/Networking/http2

[16] Web Technology Surveys. Usage Statistics of HTTP/2 for Websites, November 2016,2016. URL: https://w3techs.com/technologies/details/ce-http2/all/all

The approach done there has two parts. In the first, the TLS records of each TCP connection are segmented into sets of HTTP request-response sequences that are contained entirely in each segment. Then they look for gaps in the sequence of timestamps of the records and use the gaps to segment the TCP connections. A gap is declared in the TCP connection when the difference between the timestamps of consecutive TLS records from the server is larger than 0.5 seconds or larger than 20 times the average back-to-back response gap from the server. In the second part, the sets of HTTP request-response of the segmented TCP connections are analyzed in order to compute response sizes.

Website fingerprinting is something yet to be effectively tested in web applications using HTTP/2. This work is motivated by these previous studies and its aim is to improve the efficiency of fingerprinting attacks on web applications which employ HTTP/2.

Chapter 3

Methodology and Data Set

This chapter describes the procedure followed in this dissertation. We describe and explain each step taken and its purpose in the achieving of the objectives of this work. This chapter also analyzes the data set used in this study and take the example of a website (Google) and examine its data in more depth.

3.1 Methodology

3.1.1 Overview

The methodology used has two phases. The first one is the ground truth perspective, which means that there is complete access to all of the contents of the packets exchanged with the website. It is indispensable to know this ground truth so that it is possible to have full assurance of what is what, in order to later predict and try to identify a website. For example, if in a website there is a sequence of packets with a specific order, which are easily identifiable from every other set of packets (even with encrypted traffic), it is crucial to know what those packets are (if images or scripts, etc.) and to which website they belong to.

The other phase is from the perspective of the attacker. Here, only the encrypted information is available, which represents a situation of an actual attack. For this case, we used the estimate of HTTP response sizes used in [4]. The ultimate goal is to be able to tell, from that information, what is the website in question, with some degree of certainty. For each of the *200* websites chosen, there were made *10* repetitions for each.

3.1.2 Captures

The websites chosen were the top *200* websites in the world according to Alexa's list *The top 500 sites on the web*, as of February of 2017. This list ranks websites according to several indicators, such as:

- Daily Time on Site: the daily time on the website per visitor;

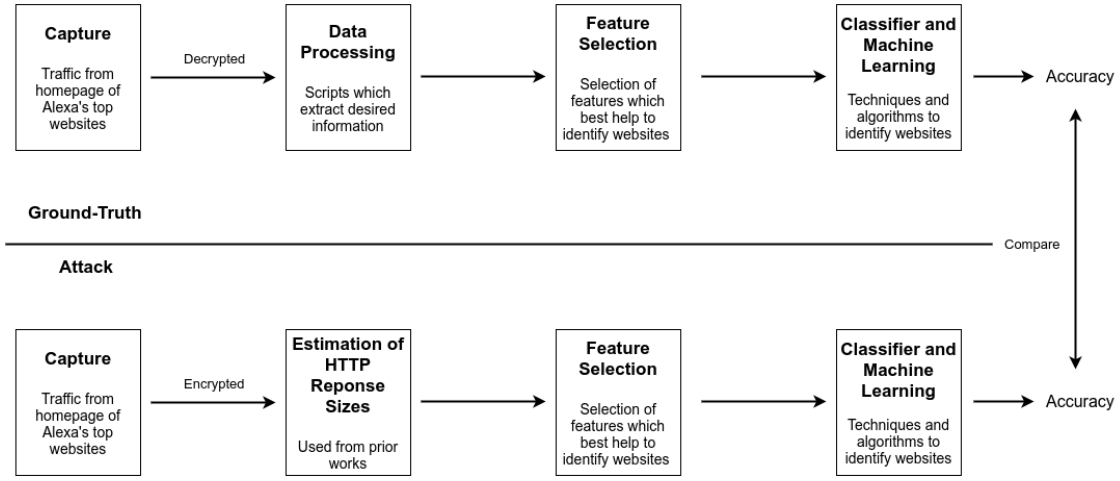

Figure 3.1: Methodology Overview

- Daily Pageviews per Visitor: the daily unique pageviews per visitor on the website;
- % of Traffic From Search: the percentage of all referrals that come from search engines;
- Total Sites Linking In: the total number of sites that link to the site.

To get the traffic data that websites generate, we resorted to the traffic capture of those websites, so that they could then be analyzed. To do so, we used the foremost and widely-used network protocol analyzer *Wireshark*, as well as the command-line packet analyzer *tcpdump*. In the first phase, in which we wanted the full disclosure of the packets' contents, there was the need to decrypt them.

Modern web browsers, such as *Google Chrome* and *Mozilla Firefox*, have a feature called *SSL Key Log*. This log file contains the symmetric session key used to encrypt the TLS traffic. Thus, after exporting the file, it can be used to decrypt the traffic with a protocol analyzer. If using the operating system Linux, as was the case, the command needed to extract the file is the following:

```
$ export SSLKEYLOGFILE = /path/to/sslkey.log
```

The capture of the website was straightforward. It consisted simply in opening the web browser (in this case it was *Google Chrome*), entering the URL of the website, including the specification of the HTTP protocol over TLS (*https://<url>*), and waiting for it to load completely, which is determined by the moment the title of the HTML page appears on the browser's tab. In this work, only the homepages of the websites were considered. Collecting a large data set required us to automate all these steps. In order to do so, we used the *Selenium* web driver as well as *ChromeDriver*. These tools work in the following way: *Selenium* provides the language bindings, and *Chromedriver* is an executable which acts as a bridge between the web browser (*Chrome*) and the driver. [1]

[1] https://github.com/SeleniumHQ/selenium/wiki/ChromeDriver

The final product of this step is a *.pcap* file, which is the standard for files originated by capturing network traffic. This file is the input for the next step, which is the data extraction from the capture.

3.1.3 Data Extraction and Processing

From the .pcap files resulting from the capture of each website, we proceeded to the extraction of the relevant information in those files. To automate, organize, filter and select the intended capture fields, (TLS and HTTP/2) we made use of scripts. The ultimate goal was to obtain a *.db* file storing all the desired information.

The information in the .pcap file was first extracted using *tshark*, a terminal-based *Wireshark*. The command was the following:

```
$ tshark -nr <pcapfilename> -2 -o \
        ssl.keylog_file:<sslkeysfilename> -T pdml ssl
```

where:

- ***-nr <pcapfilename>*** disables all name resolutions (n) and sets the filename to read from (r), which is followed by the .pcap file name
- ***-2*** sets the processing as a two-pass analysis, which causes tshark to buffer output until the entire first pass is done, but allows it to fill in fields that require future knowledge, such as 'response in frame #' fields. Also permits reassembly frame dependencies to be calculated correctly [2]
- ***-o ssl.keylog_file:<sslkeysfilename>*** sets the SSL key log file we want to use in order to decrypt the capture
- ***-T pdml*** defines the format of the output file

The database engine used was *SQLite*, which has a rich API interface with *Python*, which was the programming language used throughout this work.

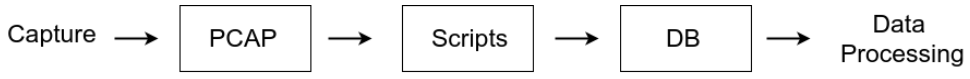

Figure 3.2: Data Flow

The scripts responsible for the transformations and conversions of the data until obtaining the .db file were not developed in this dissertation, and were instead made available from prior work.

[2] https://www.wireshark.org/docs/man-pages/tshark.html

After what is resumed in Figure 3.2, the .db file is the input of the Python script developed to process the data. In the approach of the attack the DB file already contains the estimates of the HTTP response sizes, in which the traffic is all encrypted. The Python script used to process the .db file directly selects some features from it and calculates others in order to create new features, such as the averages, as described in chapter 4.

3.1.4 Classification

In order to predict websites with a fingerprinting attack, we used the KNN classification algorithms described in 2.5.1. We chose three values for K: 1, 3 and 5. To divide the data set in training and testing, we simply randomized the data order, and took 75% of it for training and 25% for testing. To put this algorithm in practice, we used the *R 3.3.2* software and programming language.

The challenge of this work is to determine the features, among all of the proposed ones (Section 4.1), which better represent and help identifying the websites. In order to discover those we tested, for each K, every possible combination of features and applied the KNN algorithm. This way, it was possible to verify which features and combinations of features have the highest accuracy and thus are the best for the purpose of this work.

3.1.5 Measures

To measure how good the prediction results are, there are several statistical measures one can use. From the classification algorithm used, a possible visualization of the performance of the algorithm is through a *confusion matrix*. This matrix actually consists in a table in which the columns represent the prediction and the rows represent the actual values of each class (or vice versa, but in this study we used the rows as the true and the columns as the prediction). This table aims at measuring the algorithm's correct labelling of classes, as well as its mistakes. Table 3.1 shows an example of a confusion matrix. Note that this is a general example of a confusion matrix, which means that one can have more columns and rows, depending on the number of classes. For purposes of showing the four different concepts in a confusion matrix, it was shortened. In this study the confusion matrix is 200x200.

	Positive Prediction	Negative Prediction
Positive Condition	True Positive (TP)	False Negative (FN)
Negative Condition	False Positive (FP)	True Negative (TN)

Table 3.1: Confusion matrix outline

In a confusion matrix, the four general names given to its cells are [26]:

- **True Positive**: Capture traffic from a website correctly predicted as the same website

- **False Negative** (**error**): Capture traffic from a website predicted as a different website
- **False Positive** (**error**): Predicted a website for which the capture traffic did not belong to
- **True Negative**: Capture traffic which is not from a website, was not predicted as that website

From the values of these fields, there are several metrics to evaluate. The ones used in this work were [26]:

- **Recall** (True Positive Rate), which measures the proportion of positive conditions that are correctly identified as such: $TPR = \frac{\Sigma TruePositive}{\Sigma PositiveCondition}$
- **Precision** (Positive Predictive Value), which measures the proportion of the relevant instances among all of the retrieved ones: $PPV = \frac{\Sigma TP}{\Sigma PositivePrediction}$
- **Accuracy**, which measures the proportion of correct predictions among every prediction, or, in other words, the degree of closeness of the predictions to their true value [27]: $Accuracy = \frac{\Sigma TP + \Sigma TN}{\Sigma TotalPopulation}$
- **F1 score**, which is a measure of the accuracy, by calculating the harmonic mean of precision and sensitivity: $F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2TP}{2TP+FP+FN}$

None of this features is a reliable metric for the performance of a classification algorithm by themselves, because they are vague. For example, by the value of the accuracy it is not possible to know if the majority of the errors were due to False Positives or False Negatives. Moreover, for example when we want to tell, from a capture traffic of a website, how many of the predictions made were from that website in relation to the ones that are actually from that website. If we tell that all of them are from that website, we have a precision of 100%, which is misleading. This was the reason why additional metrics were used, in order to have more significant and detailed information about the results and the way the classifier performs.

Ultimately, these measures are what enables us to access the quality of our results.

3.2 Characterization of the Data Set

3.2.1 Diversity of Websites

Websites behave differently in several aspects, one of those being that some websites redirect the client to other servers. These servers, which are chosen by the the way the website is designed and implemented, may be diverse. They can use different software as HTTP server, with examples such as *Apache* or *Nginx*, but they can also use different versions of HTTP. In this study we only considered connections which use HTTP/2.

Different captures of the same website are likely different, for a number of reasons including:

- Ads. Website traffic is highly influenced by the amount of publicity they have. Moreover, the advertisement contents in a website are very likely to change over time.
- News websites vary their content regularly. Everyday there are different news, and so the website usually has a different layout
- Social Networks. Although social networks' homepages are generaLly the same, the website's feed is always changing and its content is unpredictable.
- The fact that the user is logged in or not. In some websites, if the user is already logged in, the website will redirect it to other type of page whereas if it is not, the homepage will be displayed.

Besides the fact that a great majority of websites change over the time, there is also the variant that even doing several consecutive captures to the same website, we may end up having different samples. Firstly, the simple geographical position of where the experiment was done makes the difference. One of the reasons is because there is a completely distinct experience when accessing a website with a nearby server than accessing other with a server on the other side of the world. The latency will be higher and it will affect the connection. The other reason can be that depending on the geographical position of the client, the website may redirect not only to a different server, but also to a different website.

3.2.2 Web Object Size for All Captures

Taking a general approach, when evaluating the whole data set, the web object sizes have clear pattern in the web object sizes. Figure 3.3 shows de CDF of the web object sizes, revealing that one of the largest portions has a very small size (less than 100 bytes). From there, there is an approximately even distribution of web object sizes, while there is a reduced amount of web object sizes higher than 100KB. 13% of the web objects had a size of zero bytes and were not included in the graphic.

3.2.3 Large Web Objects

The majority of websites have either 0, 1, 7 or 8 response data packets with a size larger than 100KB (figure 3.4). Furthermore, the website with the highest number of packets with a size bigger than 100KB (website 35: *tmall.com*) has only 14 of these packets, which tells that this is a unique feature, meaning that websites that have a value for this different than zero can be easily identified.

3.2.4 Example: Google

Taking the example of the first website of the *Alexa*'s list: *google.com*. Its homepage is the one shown in figure 3.5. This already exemplifies what has been described in section 3.2.1, which is

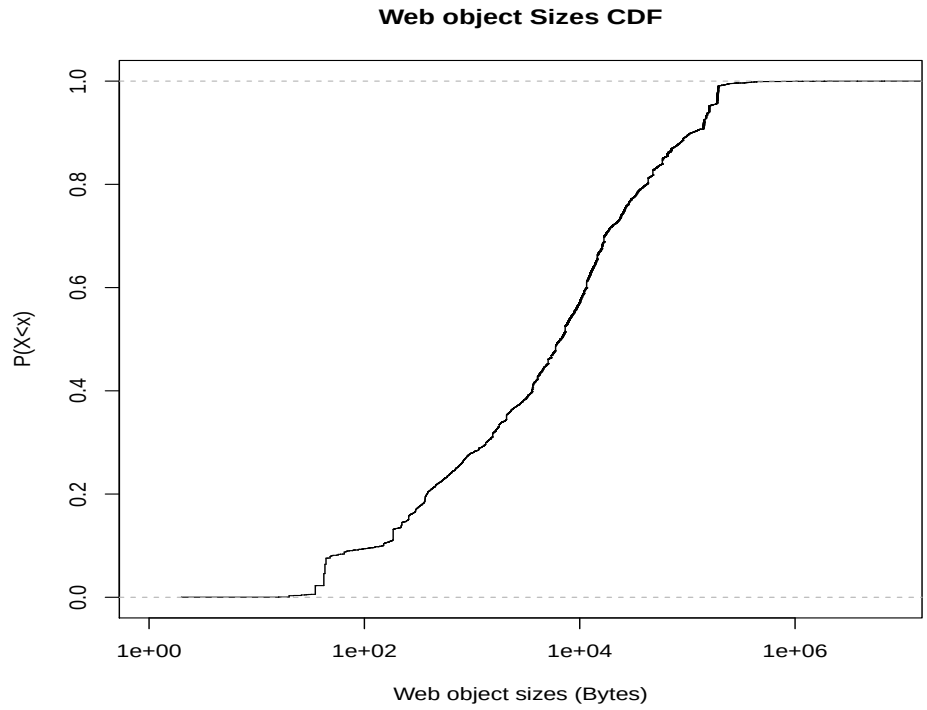

Figure 3.3: CDF of HTTP Response Data Sizes

the fact that although the URL submitted was *https://google.com*, the server redirected the client to Google Portugal (*https://google.pt*), which was the country in which the experiment took place.

3.2.4.1 Web Object Sizes

From the 10 repetitions made, this website's web object (HTTP response data) sizes have a CDF as shown in figure 3.6.

3.2.4.2 IP Addresses and TCP Connections

There are also several IP address that participate in the connection to this website. Table 3.2 enumerates the ones with at least one occurrence as well as their frequency.

For the 10 repetitions made for this website there was an average of **13,2** TCP connections per website access and figure 3.7 shows the outline of the size of the SSL records for each TCP connection. Note that the numbers of the TCP connections in the x-axis only represent TCP connections in the access to the website which are directly related to SSL records of the website. In other words, if for example the first TCP connection registered is not related to the website or doesn't have SSL, it is not considered.

It is possible to observe a clear pattern: in the beginning of the connection, for the first 6 TCP connections there are small SSL records. Then, in the TCP connections number 6 and 8 we have the majority of the information with SSL records over 300KB, and in the end it is more

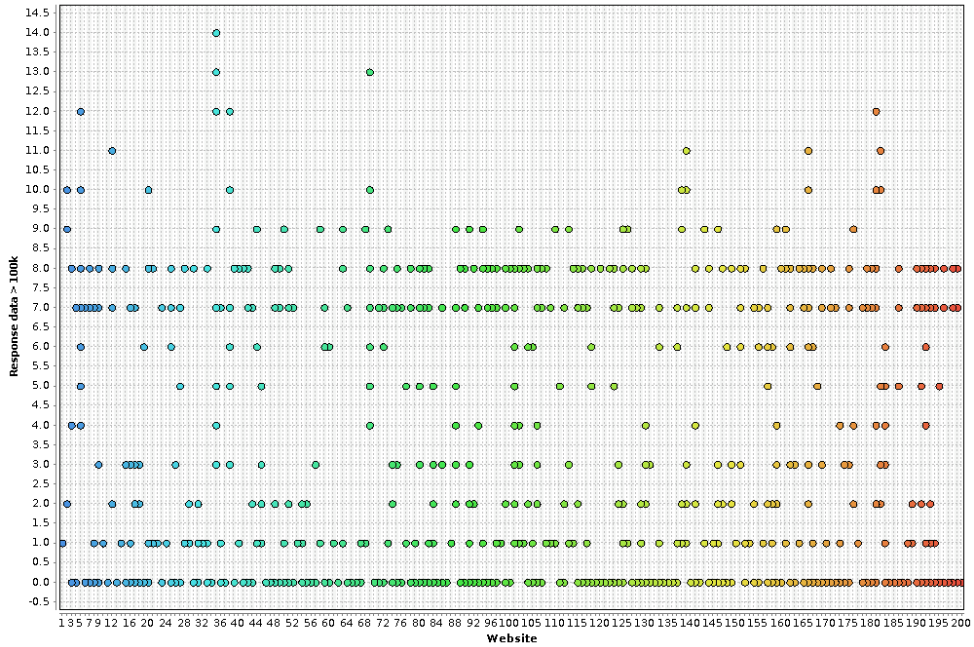

Figure 3.4: HTTP response data sizes higher than 100KB

unpredictable to determine the size of the SSL records, although they are relatively small (less than 100KB).

Regarding the number of web objects per TCP connection, table 3.3 shows the behaviour of the website for each repetition made.

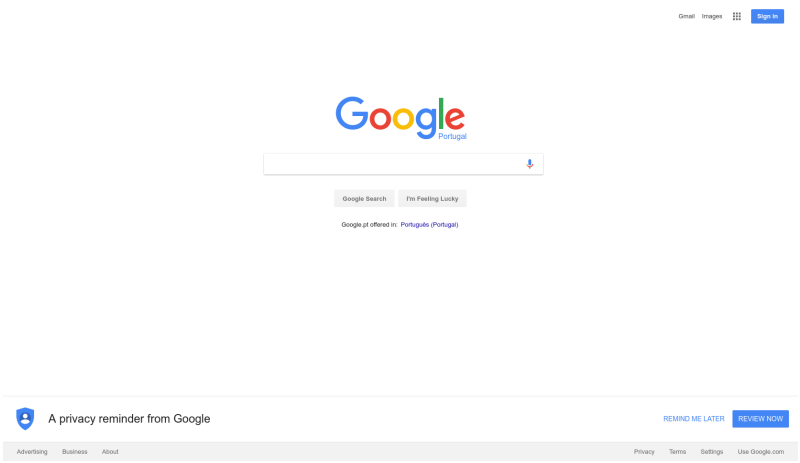

Figure 3.5: Google Homepage

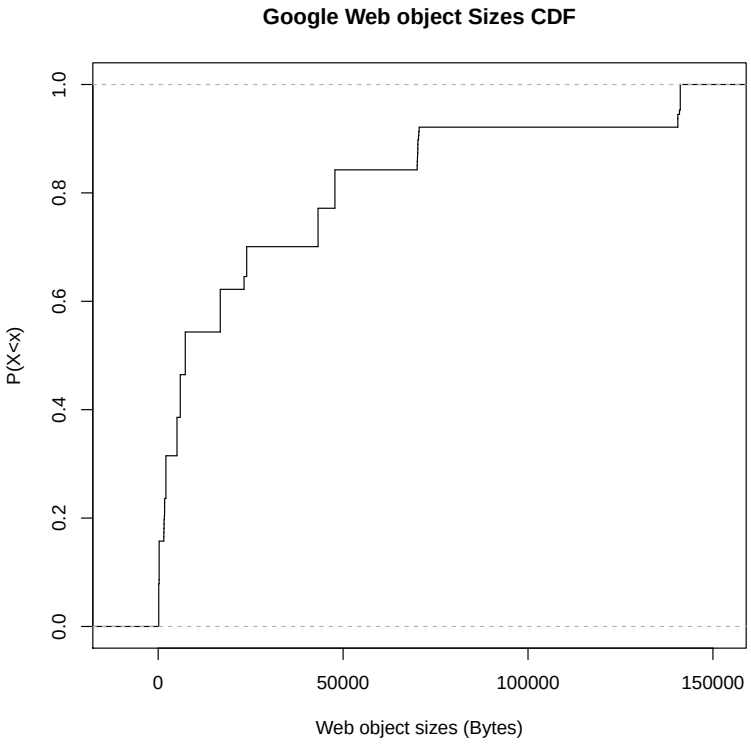

Figure 3.6: CDF of Google's Web Object Sizes

IP Address	Frequency (%)
216.58.211.227	8,33
54.230.197.64	6,06
216.58.211.195	5,30
52.18.124.61	5,30
52.88.7.60	5,30
54.230.197.121	5,30
216.58.214.174	4,55
35.164.91.224	3,79
52.38.179.134	3,79
54.148.163.250	3,79
216.58.211.206	3,03
216.58.214.163	3,03
54.230.197.100	3,03
216.58.210.142	2,27
35.163.57.6	2,27
54.191.71.231	2,27
54.230.197.240	2,27
54.230.197.88	2,27
216.58.201.131	1,52
216.58.210.174	1,52
52.35.147.139	1,52
54.154.207.225	1,52
54.230.197.239	1,52
52.89.48.161	1,52
54.230.197.11	1,52
194.210.238.170	1,52
194.210.238.155	1,52

Table 3.2: IP Addresses used when accessing Google

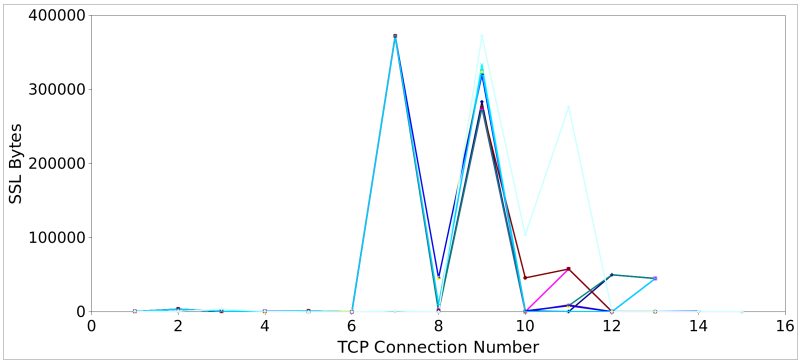

Figure 3.7: Number of SSL Bytes in TCP Connections

		Web Objects			
Repetition	# TCP Streams	Average	Minimum	Maximum	Standard Deviation
1	5	3,2	1	11	4,38
2	4	4	1	10	3,78
3	4	4	1	12	5,35
4	3	5,33	1	13	6,66
5	3	5,33	2	10	4,16
6	4	4	1	10	4,08
7	6	2,67	1	10	3,61
8	5	3,2	1	11	4,38
9	3	5,33	1	12	5,86
10	2	8	6	10	2,83

Table 3.3: Web Objects per TCP Stream

Chapter 4

Feature Selection and Analysis

A connection to a website is a two-way communication. The client (usually a web browser) requests a web page, and the server responds, if possible, with the information requested. Usually this consists in a combination of HTML documents, JS scripts and images. This information reveals information about how a website behaves. Naturally, being below the encryption layer of TLS, it will not be possible to have access to the contents of the connection.

To be able to fingerprint the websites, it is necessary to know what to fingerprint. Every website is implemented in a different way, and all of them have different contents: different HTML pages, different JS scripts and different styles. Moreover, the order with which each of these contents is retrieved is also singular. It depends on how the website is built, how it is programmed and also on the HTTP server used, which defines the way the web pages are stored, processed and delivered to clients. There are numerous reasons why websites are implemented in different ways. Different websites have different load limits, different vulnerabilities to attacks, and mainly different contents. It is very different to have a website which has dynamic content when compared to one which has static content. Plus, websites store cookies, cache and form data in distinct ways. And ultimately, even the hardware and software of the operating system of the machine on which the web server runs, makes the difference. These characteristics of websites are what make them unique. However, there are still some that are more similar between them when compared to others.

We are looking for a set of characteristics and features of a website which best enable their identification. Ideally, the chosen features will make it possible for websites to be identified among all the others, which together with the ideal machine learning and classification algorithm(s), would represent a prediction efficiency of 100%. In the phase in which there is complete disclosure on the content of the traffic generated by each website, there is a much wider range of features to choose from. This is because we have at our disposal every information from the application layer, including the same which would be encrypted in a situation of an actual attack.

4.1 List of Proposed Features

The set of features that were chosen was intended to be varied. The features only include packets which include the application layer, i.e., HTTP, whether it is encrypted or not. The features chosen were:

4.1.1 General Features

- **F1**: Number of SSL records;
- **F2**: Total size of SSL packets. This feature sums the size from every SSL packet in each capture of each website;
- **F3**: Number of client to server packets. Number of packets that have the browser IP address as the source and the server's IP address as the destination;
- **F4**: Number of server to client packets. Similar to **F3**, except for the server to client direction

4.1.2 HTTP Features

- **F5**: Total size of HTTP request header. Feature that sums the size of the header in an HTTP request, for every packet that contains it;
- **F6**: Average size of HTTP request header. Total size of HTTP request header divided by the number of packets that contain a header in an HTTP request;
- **F7**: Total size of HTTP request data. Feature that sums the size of the data in an HTTP request, for every packet that contains it;
- **F8**: Average size of HTTP request data. Total size of HTTP request data divided by the number of packets that contain data in an HTTP request;
- **F9**: Total size of HTTP response header. Feature that sums the size of the header in an HTTP response, for every packet that contains it;
- **F10**: Average size of HTTP response header. Average size of HTTP response header. Total size of HTTP response header divided by the number of packets that contain a header in an HTTP response;
- **F11**: Total size of HTTP response data. Feature that sums the size of the data in an HTTP response, for every packet that contains it. HTTP response data carries the web objects sent from the server;
- **F12**: Average size of HTTP response data. Total size of HTTP response data divided by the number of packets that contain data in an HTTP response;

4.1.3 Size Group Features

HTTP response data size is a feature with major importance. It is so because in the network traffic it is the most direct characteristic of a website. It consists on the actual content a website delivers to the client, and so it requires a more detailed analysis. For that, we divided this feature by size:

- **F13**: Number HTTP response data smaller than 100B
- **F14**: Number HTTP response data size between 100B and 1kB
- **F15**: Number HTTP response data size between 1kB and 10kB
- **F16**: Number HTTP response data size between 10kB and 100kB
- **F17**: Number HTTP response data size higher than 100kB

These were the totality of features taken into account by the study. The line of thinking was that the more features were considered, the more information would come as a result, and consequently, the more accurate would be the fingerprinting, although this may not always true.

4.2 Characterization of the Features

Table 4.1 shows some statistics regarding the totality of the collected data set for every feature, namely fields such as minimum, maximum, average and standard deviation values. This table serves to illustrate the wide variety of values that the features have.

4.3 Correlation

Correlation is a statistical measure that describes the degree of dependence between two variables. However, simply saying correlation covers a wide range of definitions. There are several ways of characterizing correlation, and consequently, different ways of calculating this value. One measure of the linear correlation between two variables is the *Pearson correlation coefficient*[1]. This coefficient is a value between -1 and 1, where 1 is the total positive linear correlation (direct proportionality, which means that a change in one variable is accompanied by a change in the other, related by a constant multiplier), 0 represents no linear correlation, and -1 is the total negative linear correlation (inverse proportionality, the opposite of direct proportionality). Its formula is:

$$\rho_{X,Y} = corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{4.1}$$

where:

- $cov(X,Y)$ is the covariance of X and Y

[1] Pearson Correlation. Kent State University. URL: http://libguides.library.kent.edu/SPSS/PearsonCorr

Feature	Minimum	Maximum	Average	Deviation
Number of SSL records	0	3K	514,03	512,06
Total Size of SSL records	0	19,62M	1,39M	1,61M
Number of Client to Server Packets	0	466	73,32	76,63
Number of Server to Client Packets	0	2,62K	440,71	454,09
Total Size of HTTP Request Header	0	32,63K	3,42K	4,83K
Average Size of HTTP Request Header	0	650	156,23	135,26
Total Size of HTTP Request Data	0	39,25K	144,88	1,37K
Average Size of HTTP Request Data	0	3,57K	59,24	272,42
Total Size of HTTP Response Header	0	67,07K	4,06K	7,69K
Average Size of HTTP Response Header	0	883	156,66	139,86
Total Size of HTTP Response Data	0	10,39M	524,46K	778,09K
Average Size of HTTP Response Data	0	76,39K	31,3K	57,87K
Number of HTTP response data sizes less than 100B	0	53	1,79	4,45
Number of HTTP response data sizes between 100B and 1KB	0	83	3,14	7,83
Number of HTTP response data sizes between 1KB and 10KB	0	68	4,95	9,86
Number of HTTP response data sizes between 10KB and 100KB	0	88	5,47	10,34
Number of HTTP response data sizes higher than 100KB	0	14	1,77	2,86

Table 4.1: Data Set Statistics by Feature

- σ is the standard deviation
- μ is the mean
- E is the expectation

Applying this equation to the features mentioned above, the correlation result was the one visible in Figure 4.1.

From this graphic there are several conclusions that can be drawn. One is that the number of SSL records and the number of server to client packets have a high correlation coefficient. As a result , one of these features could be ignored for this specific data set.

After testing every possible combination of all the features on the classifiers, we made some adjustments. These consisted in selecting only the features which had essential information (not redundant) and discarding those that introduced noise to the experiment. The choice of which features to select and which to discard was directly related to the correlation obtained of the first set of features, as it is shown in Figure 4.1.

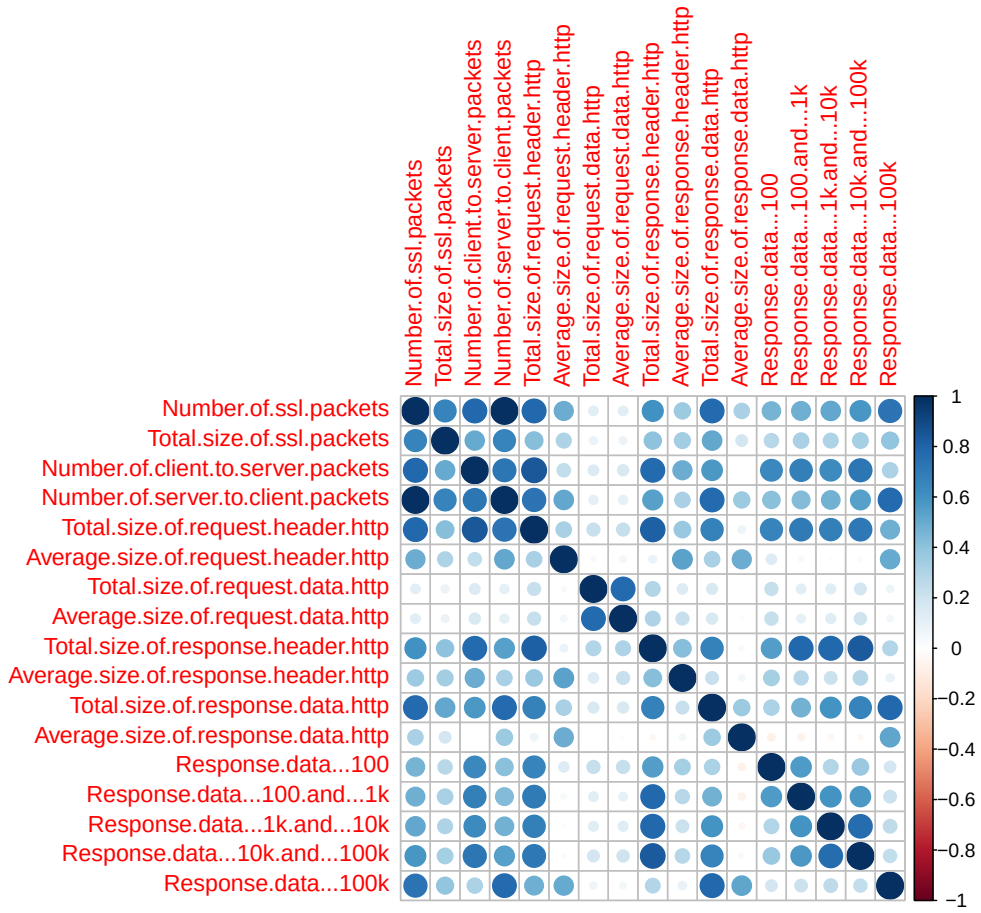

Figure 4.1: Correlation of Features

4.4 Features Effectiveness

4.4.1 General and HTTP Features

In a first experiment, we tested the data set in the classifier algorithm with all of the features in sections 4.1. For this experiment, we used 75% of the data set as training and 25% as testing, and for K=3 in the KNN classifier we had as a result the confusion matrix in figure 4.2.

In this confusion matrix, the darker the dots, the higher the number in the correspondent cell. At a first glance, the perfectly visible bold diagonal line catches your attention. However, there is some dispersion in the rest of the graphic. This means that the classification was not perfect, in fact, the accuracy was just 36.8%.

The results when using every single feature chosen were not impressive. The best accuracy value was 44,6% for K=1, while for K=3 was 34,6% and for K=5 it was 33,4%. This shows that the use of all of the originally proposed features is not a good choice for fingerprinting websites in the data set used.

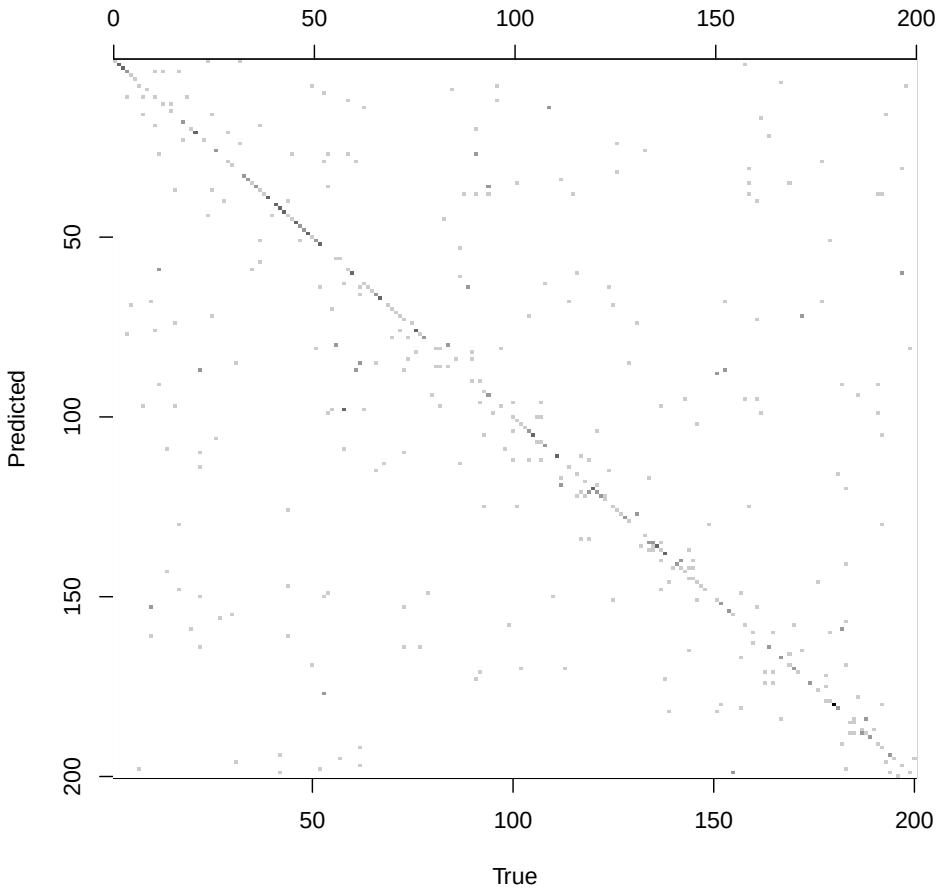

Figure 4.2: First Confusion Matrix

4.4.2 Worst Case Result Feature Combinations

Tables 4.2, 4.3 and 4.4 show the worst features and combinations of features and the respective accuracy for every K value chosen (1, 3 and 5). Here we use the feature nomination of section 4.1.

Features	Accuracy (%)
F2, F13, F15, F16	25,2
F1, F2, F3, F13, F17	26
F2, F4, F8, F13, F14	26,4
F2, F3, F4, F8, F16	26,4
F2, F4, F10, F15	26,4

Table 4.2: Worst Features for K=1

Features	Accuracy (%)
F1, F2, F3, F8, F17	18,8
F2, F4, F13, F14, F15	19
F1, F2, F7, F14, F15	19
F1, F2, F3, F4, F5, F6, F7, F8, F13, F14, F16, F17	19,2
F2, F3, F7, F14, F16	19,2
F1, F2, F7, F13	19,2

Table 4.3: Worst Features for K=3

Features	Accuracy (%)
F2, F4, F6, F10, F13	16,2
F2, F7, F10, F13	16,4
F1, F2, F4, F6, F8	17,4
F2, F6, F7, F17	17,4
F2, F3, F4, F6, F16	17,6
F1, F2, F8, F14, F15	17,6
F2, F14, F15, F17	17,6

Table 4.4: Worst Features for K=5

From these tables, we can straightaway draw the conclusion that for the KNN classifier, K=1 is clearly the one with best results and K=5 the one with the worst. This may happen due to the fact that the higher the number of neighbors to take into account, the more noise is being introduced.

4.4.3 Best Case Feature Combination

Tables 4.5, 4.6 and 4.7 show the best features and combination of features and the respective accuracy for every K value chosen (1, 3 and 5).

Features	Accuracy (%)
F13	98,4
F16	98,4
F14, F16	98,4
F13, F15	98,2
F13, F14, F15, F16	98,2

Table 4.5: Best Features for K=1

The features that give the best accuracy results have the similarity of being part of the features that divide HTTP response size by sizes (section 4.1.3). Like for the features with the worst accuracy, the order of best parameter for K is still the same.

Features	Accuracy (%)
F14, F16	95,6
F16	95,2
F14, F15	95,2
F13	95
F13, F14	95

Table 4.6: Best Features for K=3

Features	Accuracy (%)
F13, F14, F16	93
F14, F16	90,8
F14, F15, F16	90,8
F13, F15	90

Table 4.7: Best Features for K=5

Chapter 5

Results on Efficiency with HTTP/2

Here we describe the second approach of this work. We take the perspective of the attacker, in which we do not have access to the decrypted traffic. The features used in this perspective result from the estimation of HTTP response sizes from TLS records, done in [4], as it was explained before.

The features chosen in chapter 4 were specific of the data set used in section 3.2. It was then necessary to test them in a new data set to check if, in fact, they have good results. Consequently, the data set used in this chapter is resultant from a new experience.

5.1 Comparison with Ground-truth Features

To be able to evaluate the limitations of fingerprinting in HTTP/2 web applications we need to compare the results of the sets of features of the two approaches. The features and combinations of features chosen were the ones which had higher accuracies, as described in section 4.4, and which were common between both the ground-truth and the attacking perspectives. Figure 5.1 shows, for K=1, the comparison of accuracies for the same features in the two perspectives. Figures 5.2 and 5.3 show the same for K=3 and K=5, respectively.

It is visible that even for a new data set, the best features for the Ground-Truth perspective are approximately the same, and have high accuracies, just like for the original data set. This proves that the features were chosen independently of the data set used.

We can then observe that for the same features and combinations of features the accuracies are roughly the same. This can be due to two reasons:

- The estimation of HTTP Response Size performed well, and so the estimations were similar to the actual values of HTTP Response Size and consequently the results were identical.

- The best features were identical for both perspectives. If the case was that the best features were considerably different for both perspectives, the results would not be so similar.

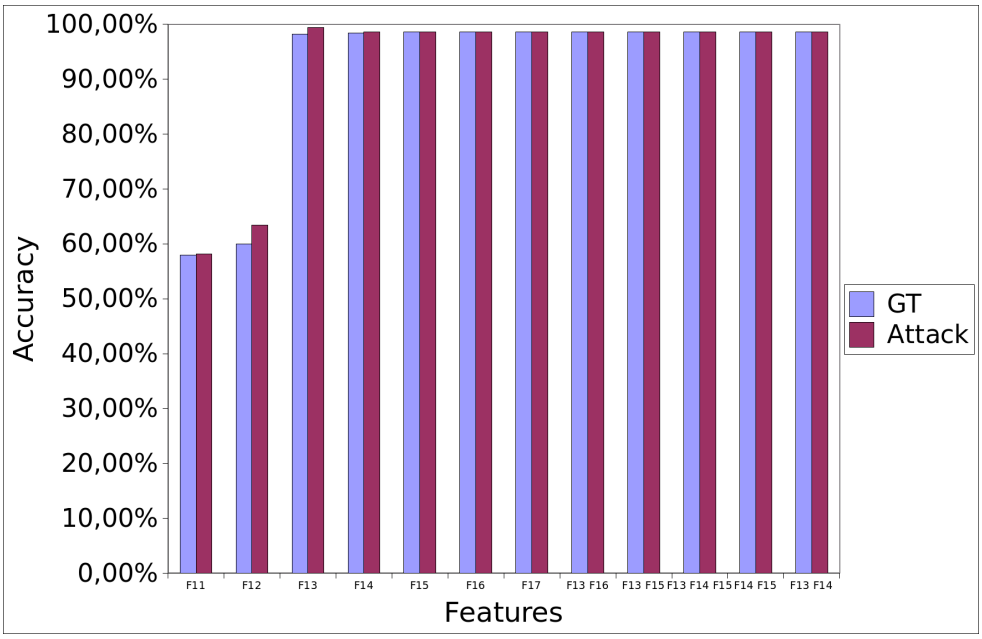

Figure 5.1: Comparison of the Accuracy of Ground-Truth and Attack Features (K=1)

5.2 Pipelining and Multiplexing

The major difference in HTTP/2 when compared to previous versions of HTTP is the use of pipelining and multiplexing. It is then of great importance to evaluate how these mechanisms influence fingerprinting attacks. We first determined the amount of pipelining and multiplexing each website used. For that, we came with a metric for each, a pipelining and a multiplexing value. The pipelining value is the number of HTTP/2 pipelined bytes divided by the total number of HTTP/2 bytes. The multiplexing value was determined by dividing the HTTP/2 multiplexed bytes by the number of HTTP/2 pipelined bytes.

To evaluate the influence of these two aspects, we separated websites in three categories:

- No pipelining: websites which have an average of pipelining smaller than 0.05
- With pipelining: websites which have an average of pipelining higher than 0.05
- With multiplexing: websites which have an average of multiplexing higher than 0.05

61.5% of the websites are in 'No Pipelining', 38.5% in 'With Pipelining' and 11.5% in 'With Multiplexing'.

Then we evaluated in a CDF the **F1 Score**, as defined in section 3.1.5, per website, for each of the above categories. The result is shown in figure 5.4.

It is clear that the pipelining and multiplexing mechanisms have influence on the website's accuracy. This can be concluded by analyzing the different lines in the CDF graph. It is visible that there is higher occurrence of low F1 scores with multiplexing and pipelining. On the other hand,

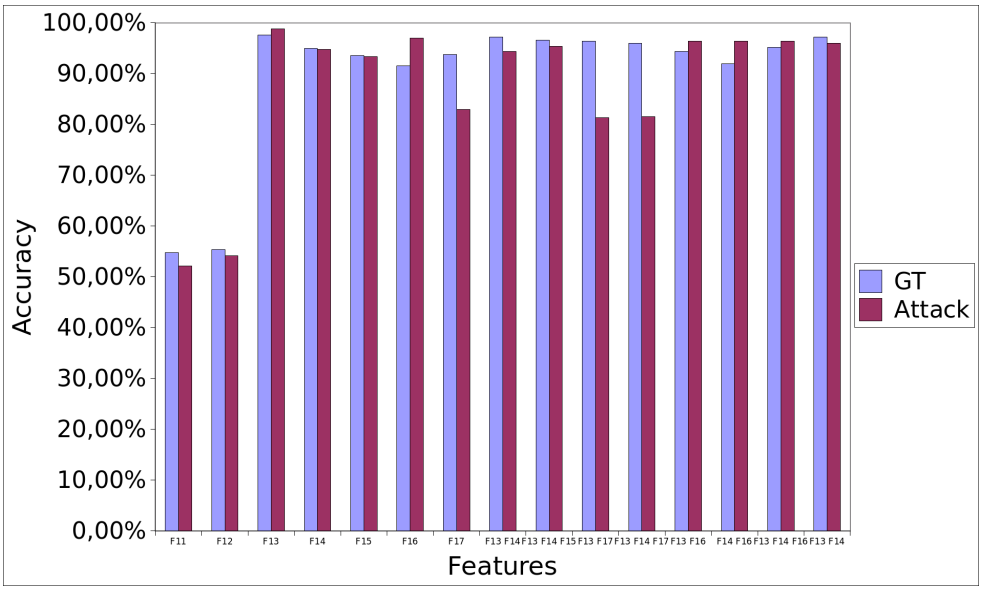

Figure 5.2: Comparison of the Accuracy of Ground-Truth and Attack Features (K=3)

websites which don't use pipelining have higher probability of being successfully fingerprinted. In fact, more than 30% of websites which do not use pipelining have an F1 score of 100%.

Afterwards, we did the same, but instead of considering the F1 score of the attacking perspective, we used the difference between F1 scores of both the ground-truth and attacking approaches. Thus we can evaluate not only the impact of pipelining and multiplexing, but also the difference it makes when changing the approach. The result is shown in figure 5.5. The x-axis of the graphic shows the absolute value of the difference of the F1 scores. F1 scores range from 0 to 1, so in this graphic, values of the difference closer to 1 mean that the ground-truth and the attack perspective have higher amount of different F1 scores, while values of the difference closer to 0 mean that both perspectives have similar results.

The results are a bit different than the previous figure. In the two perspectives, multiplexing is the category which is the least affected, having more F1 score differences closer to 0. Nonetheless, for all the categories, almost 20% of websites have an equal score both for the ground-truth and attacking approaches (difference equal to 0).

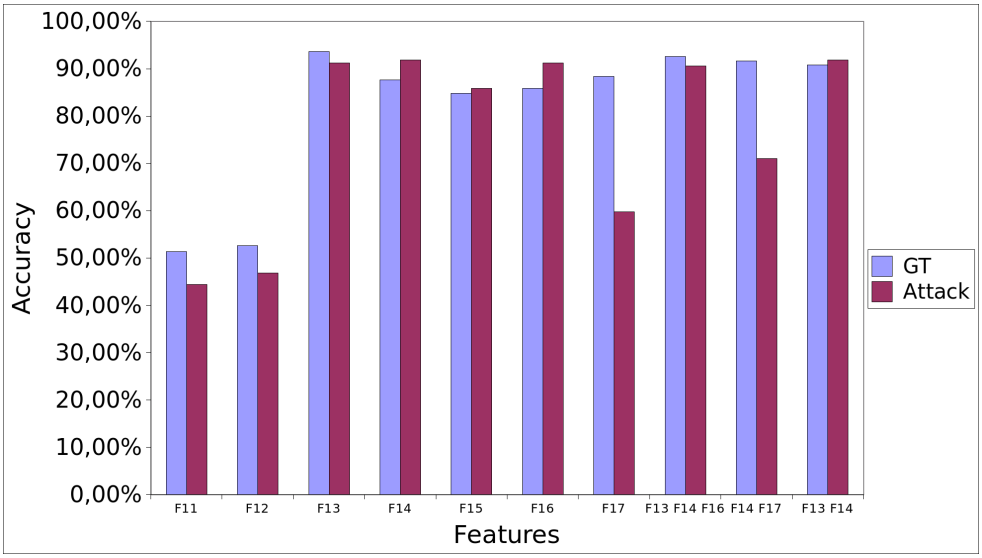

Figure 5.3: Comparison of the Accuracy of Ground-Truth and Attack Features (K=5)

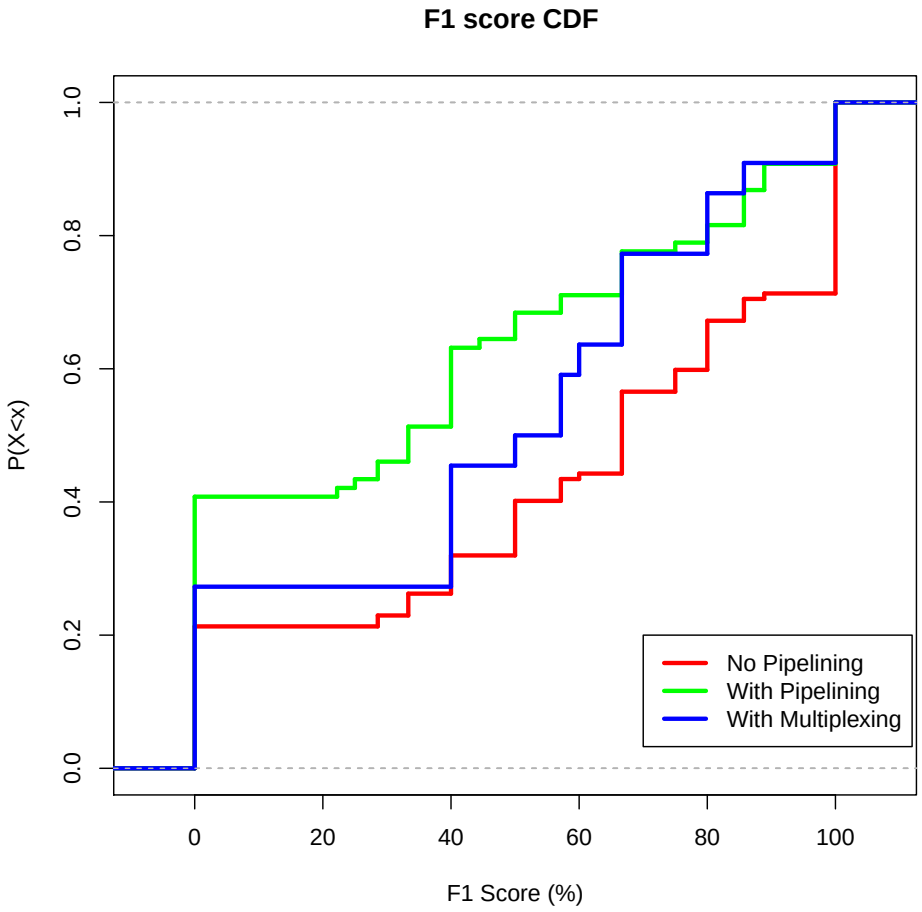

Figure 5.4: F1 score CDF, according to pipelining and multiplexing levels

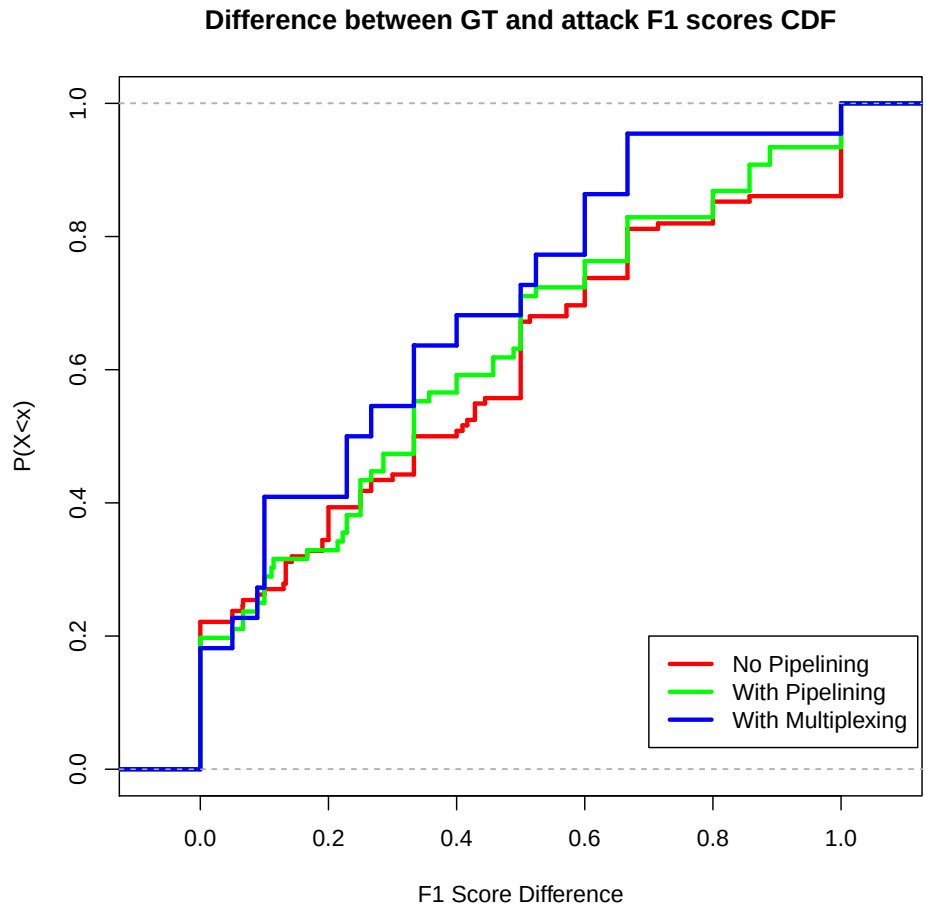

Figure 5.5: Difference between GT and attack F1 scores CDF, according to pipelining and multiplexing levels

Chapter 6

Conclusions

In this chapter we present the conclusions for this work. We summarize, discuss and try to explain the results obtained. From there we try to present ways in which this work could be improved as well as what can be studied further in order to have a deeper and more detailed approach to the problem addressed in this work.

6.1 Discussion of Results

The results obtained in this work are promising. We show that it is feasible to successfully employ fingerprinting attacks in websites using HTTP/2.

The features which provide higher accuracies are the ones related to the content websites retrieve, namely the HTTP response sizes. It was somewhat expected that these would be the ones with higher accuracies rather than other features such as the size of HTTP requests, because the size of web objects is the size of the content of a website, which is what defines it best. We improved the results by dividing this feature by intervals of sizes, instead of considering only the total and the average values.

The best parameter of the K-NN algorithm was K=1. Because this algorithm makes its classification based on the nearest objects, the more objects there are to consider, the more likely it is to consider objects which belong to other classes.

The accuracies of the fingerprinting attack were above 90%. This high number can be due to the relatively small data set that was used, and also to the fact that, from the websites observed, a large number of those use little pipelining and multiplexing. In fact, as described in section 5.2, only 11.5% of websites used multiplexing and 38.5% used pipelining.

The use of pipelining and multiplexing was also proved to have a big impact on the ability to employ fingerprinting attacks. The results are much better when the levels of pipelining and multiplexing are low, which proves that using these mechanisms complicates the employment of the attack.

The estimations of HTTP response sizes from TLS records proved to have a good performance. This is particularly relevant because it was a fundamental part of this dissertation, due to its importance in the attacking perspective.

6.2 Future Work

This study has surely had some promising results. There is a clear relationship between the usage of HTTP/2 by websites and the ability to fingerprint them. Furthermore, the amount of pipelining and multiplexing applied to the websites has also a strong influence in their identification. However, there can be made some improvements to this study.

6.2.1 More substantial Results

Firstly, there may be used a bigger data set. 200 websites is practically nothing among the immense quantity of websites existent. Doing this study to a reduced data set, although the targets were the biggest websites, it does not come close to the characterization of the Internet. Therefore, there should and must be used a much larger number of samples in order to have a more reliable study.

The classification is of great importance in this work. We used the K-NN algorithm and we even used varied parameters in order to find the best one. However, it does not mean that this is the best results we could achieve. There are numerous machine learning and classification algorithms that could be applied for this work. An improvement to this study would be to apply more classifiers to it and check how the results would differ.

6.2.2 New Contributions

When evaluating the web objects and their sizes, we treated all of them in an equal way. This is an important issue, because there are web objects that can be shared by more than one website and because in different captures of the same website there can be a variation on the web objects. For that, we could use the *entropy*, which measures the amount of information in a message, or in this case, a web object. The formula uses the probability of a web object size to appear in a website:

$$Entropy_k = \sum_{i=1}^{n} p_{ki} * log(\frac{1}{p_{ki}}) \tag{6.1}$$

where:

- k is the index of the web object size
- n is the number of websites
- p_{ki} is the probability of the web object size k to appear on website i

With this measure we could then evaluate every web object by this value. Instead of identifying web objects by intervals of sizes, we could identify them by their exact value. From here we would only consider those with low entropy, because those would be the most unique ones. Using only these when analyzing the web objects could mean better results.

With a big data set, comes automation. For this dissertation it was not viable to analyze every website in depth. We made a detailed approach for the case of Google in section (3.2.4), but that was just made in order to give an example of the information gathered. To have a more significant study, there could be made a more detailed and individual analysis for each website so that we could extract every single website profile.

If an attack is proved to be successful, the natural order would be to find ways of defending and preventing it. So, future work may include the definition of countermeasures in order to make sure Internet users will not be victims of this type of attacks.

References

[1] Daniel J. Bernstein. *Cache-timing attacks on AES*. Technical report, 2005. Cited on page 2.

[2] Paul C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, pages 104–113. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. Cited on page 2.

[3] Y. Shi and S. Biswas. Website fingerprinting using traffic analysis of dynamic webpages. In *2014 IEEE Global Communications Conference*, pages 557–563, Dec 2014. Cited on pages 2 and 12.

[4] Ricardo Morla. An initial study of the effect of pipelining in hiding HTTP/2.0 response sizes. *CoRR*, abs/1607.06709, 2016. Cited on pages 3, 4, 13, 15, and 35.

[5] Qinglong Wang, Amir Yahyavi, Bettina Kemme, and Wenbo He. I know what you did on your smartphone: Inferring app usage over encrypted data traffic. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 433–441. IEEE, 2015. Cited on page 3.

[6] David Kahn. *The codebreakers : the story of secret writing*. Scribner, New York, 1996. Cited on page 5.

[7] Thomas Kelly. The myth of the skytale. *Cryptologia*, 22(3):244–260, 1998. Cited on page 5.

[8] Ralph Holz, Yaron Sheffer, and Peter Saint-Andre. Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS). Cited on page 6.

[9] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communications Magazine*, 47(12), 2009. Cited on page 7.

[10] A. R. A. Bouguettaya and M. Y. Eltoweissy. Privacy on the Web: Facts, challenges, and solutions. *IEEE Security & Privacy*, 99(6):40–49, 2003. Cited on page 7.

[11] Ryan Singel. Declassified NSA Document Reveals the Secret History of TEMPEST, 2008. Cited on page 8.

[12] Wim van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Comput. Secur.*, 4(4):269–286, December 1985. Cited on page 8.

[13] David Brumley and Dan Boneh. Remote timing attacks are practical. In *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12*, SSYM'03, pages 1–1, Berkeley, CA, USA, 2003. USENIX Association. Cited on page 9.

[14] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 191–206, Washington, DC, USA, 2010. IEEE Computer Society. Cited on page 9.

[15] Andrei Z. Broder. Some applications of rabin's fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science*, pages 143–152. Springer-Verlag, 1993. Cited on page 9.

[16] Matthew Smart, G. Robert Malan, and Farnam Jahanian. Defeating tcp/ip stack fingerprinting. In *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9*, SSYM'00, pages 17–17, Berkeley, CA, USA, 2000. USENIX Association. Cited on page 9.

[17] Tom M. Mitchell and others. *Machine learning*. McGraw-hill, 1997. Cited on page 10.

[18] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992. Cited on page 10.

[19] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Proceedings of the 2Nd International Conference on Privacy Enhancing Technologies*, PET'02, pages 171–178, Berlin, Heidelberg, 2003. Springer-Verlag. Cited on page 12.

[20] David Wagner and Bruce Schneier. Analysis of the ssl 3.0 protocol. In *Proceedings of the 2Nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, WOEC'96, pages 4–4, Berkeley, CA, USA, 1996. USENIX Association. Cited on page 12.

[21] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 263–274, New York, NY, USA, 2014. ACM. Cited on page 12.

[22] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11, pages 103–114, New York, NY, USA, 2011. ACM. Cited on page 12.

[23] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 31–42. ACM, 2009. Cited on page 12.

[24] Erwin Adi, Zubair Baig, Chiou Peng Lam, and Philip Hingston. Low-Rate Denial-of-Service Attacks against HTTP/2 Services, August 2015. Cited on page 13.

[25] Matteo Varvello, Kyle Schomp, David Naylor, Jeremy Blackburn, Alessandro Finamore, and Konstantina Papagiannaki. Is the Web HTTP/2 Yet? In *International Conference on Passive and Active Network Measurement*, pages 218–232. Springer, 2016. Cited on page 13.

[26] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition. Cited on pages 18 and 19.

[27] Burghart Brinkmann. *Internationales Wörterbuch der Metrologie: Grundlegende und allgemeine Begriffe und zugeordnete Benennungen (VIM) Deutsch-englische Fassung ISO/IEC-Leitfaden 99: 2007*. Beuth Verlag, 2012. Cited on page 19.