

**Master's thesis**

# Website fingerprinting of encrypted Wi-Fi traffic

Information disclosure through side-channel exploitation

**Embrik Hafskjold Thoresen**

Informatics: Information Security

60 ECTS study points

Department of Informatics

Faculty of Mathematics and Natural Sciences

Autumn 2024



**Embrik Hafskjold Thoresen**

# Website fingerprinting of encrypted Wi-Fi traffic

Information disclosure through  
side-channel exploitation

Supervisors:  
Håkon Jacobsen, Nils Gruschka



---

## Abstract

The IEEE 802.11 standard was developed to connect devices together over a wireless local area, and it has evolved to be the most widely deployed WLAN technology. This standard contains protocols used to provide data confidentiality during transmission. While the standard protocols are secure in theory, traffic patterns leaked through side channels can be used to recover information about data being accessed online.

This thesis presents a website fingerprinting attack on encrypted 802.11 frames and assesses a potential mitigation technique aimed at reducing the impact of side-channel attacks. Initial findings indicate that an attacker can successfully exploit information leakage using machine learning, highlighting how such leakage undermines the security that encryption intuitively should provide. Furthermore, this thesis evaluates a potential mitigation technique to decrease information leakage and illustrates that this technique has minimal impact on the attacker's accuracy.

---

---

## Preface

This thesis marks the completion of the requirements for my Master's degree in Information Security at the University of Oslo, and the end of a total of five years of study at the same university. The topic of this thesis is discussed and developed together with my supervisor Håkon Jacobsen.

## Acknowledgements

I would like to express my sincere gratitude to my supervisor, Håkon Jacobsen. Thank you for your guidance and support over the past two years. Your genuine interest in the topic and curiosity have inspired and motivated me when I needed it most. I greatly appreciate your time and effort in guiding me through this project and all the discussions we have had over the past couple of years. You have given me a deeper understanding of security.

I would also like to thank my other supervisor, Nils Gruschka. I might not have developed an interest in security if it were not for you and the various courses you have taught over the past years. Thank you.

Next, I thank my family for continuously being interested in science and understanding how things work. Growing up with this attitude has given me the joy of being exposed to new ideas and a curiosity about how technology works. Thank you for your support throughout my life.

Last but not least, I would like to thank all my friends, especially those who can simply turn any bad day into a good one by being present. Thank you for the happy memories created over the past five years at the University of Oslo. You all know who you are. Thank you for being a source of joy and support throughout my time here. The past five years would not have been the same without you. Also, thanks to my friends Thanh Thao Le and Vetle Håland Bergstad for all the snacks, lunches, and support and for providing feedback on my thesis. Thank you all.

EMBRIK HAFSKJOLD THORESEN  
*Oslo, November 2024*

---



# Contents

Abstract . . . . .	i
Preface. . . . .	iii
1 Introduction . . . . .	1
1.1 Motivation . . . . .	2
1.2 Objective . . . . .	4
1.2.1 Wi-Fi . . . . .	4
1.2.2 Research questions . . . . .	5
1.3 Contribution . . . . .	5
1.4 Related work . . . . .	7
1.4.1 Failure of countermeasures . . . . .	7
1.4.2 Fingerprinting LTE/4G traffic . . . . .	8
1.5 Outline of the thesis. . . . .	10
2 Theoretical background . . . . .	11
2.1 Cryptography . . . . .	12
2.1.1 Symmetric encryption schemes . . . . .	12
2.1.2 Standard security definitions . . . . .	12
2.1.3 Length-hiding security. . . . .	15
2.2 IEEE 802.11 . . . . .	20
2.2.1 Architecture . . . . .	20
2.2.2 802.11 MAC frame format . . . . .	21
2.2.3 WPA2 . . . . .	23
2.2.4 CCMP. . . . .	24
2.2.5 Padding in WPA2 today . . . . .	25
2.2.6 Aggregation . . . . .	26

## CONTENTS

---

2.3	Machine Learning . . . . .	28
2.3.1	Classification . . . . .	28
2.3.2	The naïve Bayes classifier . . . . .	29
3	Methodology . . . . .	31
3.1	Research design . . . . .	32
3.2	Setup . . . . .	32
3.2.1	Logical components. . . . .	32
3.2.2	Physical components . . . . .	35
3.2.3	Software and tools . . . . .	36
3.3	Method . . . . .	37
3.3.1	Collection of data . . . . .	37
3.3.2	Filtering and selection of data . . . . .	43
3.3.3	Training the classifier . . . . .	46
3.4	Limitations . . . . .	47
3.4.1	Selection bias . . . . .	47
3.4.2	Limited scope . . . . .	47
3.4.3	Measurement errors . . . . .	48
3.4.4	Sample size and time constraints . . . . .	48
3.4.5	Justification and mitigation . . . . .	48
4	Results . . . . .	53
4.1	Simple PDF fingerprinting . . . . .	54
4.1.1	Trivial success probability . . . . .	54
4.1.2	The classifier’s accuracy . . . . .	56
4.2	Wikipedia fingerprinting . . . . .	59
4.3	Domain fingerprinting . . . . .	61
5	Discussion . . . . .	63
5.1	The accuracy of the website fingerprinting attack . . . . .	64
5.2	Impact of aggregation . . . . .	65
5.3	Accuracy and the trivial success probability . . . . .	67
6	Conclusion . . . . .	69
6.1	Summary. . . . .	70
6.2	Future work . . . . .	71
	Bibliography . . . . .	77
	Appendices . . . . .	1
A	Appendix . . . . .	1
B	Appendix . . . . .	1



## CHAPTER 1

# Introduction

This chapter introduces the field of study and provides a context for the research. It aims to motivate the relevance of the research by highlighting the thesis's contributions and key results.

The chapter begins by outlining the motivation behind the thesis in [Section 1.1](#), followed by an overview of this thesis's objectives in [Section 1.2](#). Next, [Section 1.3](#) provides details on the thesis's contribution before related work is presented in [Section 1.4](#). The chapter concludes with an outline of the thesis structure in [Section 1.5](#).

## 1.1 Motivation

*Encryption* is a common technique to preserve confidentiality and users' privacy. Designing secure cryptographic protocols that effectively employ encryption, however, is not easily done. Standard protocols exist that can securely encrypt messages, and these protocols have been subjected to extensive cryptanalysis. However, attackers have proven to be able to exploit information leakage of these protocols to perform an attack. These types of attacks are called side-channel attacks. While the standard protocols are theoretically secure, attackers can recover information about accessed data by analyzing traffic patterns leaked through side channels.

As encryption schemes have developed, researchers have tried to define what security means theoretically. Commonly, these definitions do not incorporate any protection of the message length, i.e., they do not consider leakage of message lengths through side channels. In practice, an attacker can obtain information by observing the lengths of encrypted messages, revealing a gap between theoretical and real-world scenarios, as attackers can obtain the lengths of encrypted messages. Consequently, security definitions need to be broadened to capture the risk of leakage and implement a technique to mitigate it.

The concept of length-hiding encryption schemes was first introduced by Paterson et al. [33]. Their security model describes an encryption algorithm that preserves message confidentiality while concealing the message length through padding prior to encryption. *Padding* involves appending additional data to a message with the primary intention of obscuring the message length. However, [36] showed the inevitable occurrence of a considerable bandwidth overhead if one wants to hide the length of the messages, highlighting the importance of considering the practical side of padding. The security model put forth by Paterson et al. [33] requires the encryption algorithm to take an additional argument determining the padding applied to a message. A limitation here is that the selection of messages from the application must be chosen based on the choice of this parameter instead of the parameter being chosen to fit all messages from the application.

The notion of length-hiding encryption schemes served as the basis of two subsequent articles [7, 12]. The latter work [12] aimed to establish a security model that is independent of any specific application while being capable of capturing application settings. This model extends the security model put forth by Paterson et al. [33], which demonstrates that under certain assumptions of the encryption scheme, an adversary's success is reduced to the amount of information that is leaked through the ciphertext lengths, called the trivial success probability.

The former [7] focused on evaluating the efficiency of different length padding strategies by combining the work of previous studies [33, 36], demonstrating the same as [12]. However, the research paper only addresses length-hiding security and does not consider other characteristics of encrypted traffic. Combining the two research papers [7, 12] indicates that choosing an efficient padding strategy reduces the adversary’s success rate. Consequently, this will reduce the attacker’s success rate when executing a side-channel attack.

A specific type of side-channel attack is called a website fingerprinting attack, involving an attacker who can monitor encrypted data and traffic patterns to determine which website a user is accessing [39]. A malicious actor merely requires the ability to monitor encrypted network traffic and being able to identify web pages visited by a user, as highlighted in several previous papers [9, 12, 13, 16, 28, 30, 39]. One scenario [13] demonstrated that an attacker could fingerprint a website without directly observing traffic patterns. According to the paper, the attacker can obtain sufficient information by sending probes from a distant location and exploiting a queuing side-channel in routers. The success rate of the considered attack was highly variable but could still threaten privacy. Another research paper [30] illustrated how an attacker could utilize supervised machine learning algorithms to automatically categorize data into one or more sets in order to identify individual pages within the same website with a high degree of accuracy. These papers illustrate an attacker’s capability to analyze encrypted traffic to determine what a user is accessing online, defeating the security that encryption intuitively should provide.

While the scenarios and objectives differ for each paper [9, 12, 13, 16, 28, 30, 39], they all have two things in common. First, all attacks consider an adversary observing encrypted network traffic on the network layer. While research indicates the possibility of executing a website fingerprinting attack on the network layer, it is not apparent that a similar attack works on a different layer. By moving the scenario down to the data link layer, the encryption scheme hides all data from higher levels, resulting in different data the attacker observes. Second, they all exploit information leakage revealed through encrypted network traffic. The capabilities of an attacker executing a website fingerprinting attack emphasize the importance of the underlying problem: while an encryption scheme may be cryptographically secure, it is not necessarily secure in practice if the encrypted traffic leaks information through side channels.

### 1.2 Objective

This thesis explores the impact of website fingerprinting attacks on encrypted network traffic on layer two. The objective is to examine the leakage of information through encrypted wireless network traffic in order to determine what a user is accessing in different scenarios.

Building on previous research [9, 12, 22] investigating the failure of different padding techniques, length-hiding encryption, and fingerprinting LTE/4G traffic respectively, this thesis takes a different approach. It considers a modified context where an attacker performs a website fingerprinting attack using machine learning, focusing solely on data observed on the data link layer. The accuracy of the website fingerprinting attack is compared to the trivial success probability in one of three scenarios, illustrating the relationship between the attack's performance and the information leaked by the lengths of encrypted messages in a theoretical setting. This comparison sets expectations for 802.11 [Section 1.2.1] in the absence of mitigating techniques. To the author's knowledge, performing a website fingerprinting attack exploiting information leaked through encrypted wireless traffic has yet to be previously explored.

#### 1.2.1 Wi-Fi

The IEEE 802.11 standard is primarily a layer two protocol developed to connect devices over a wireless local area in a home or office environment. Today, IEEE 802.11 has developed to be the most widely deployed WLAN (wireless local area network) technology, where the term IEEE 802.11 and Wi-Fi are used interchangeably in literature [3]. The two most widely used wireless security standards today are WPA2 and WPA3. While the encryption schemes differ, they have in common that they do not hide the lengths of the underlying messages, leaking the length of higher-level protocol messages and potentially making them vulnerable to website fingerprinting attacks.

A risk of wireless networking is that an attacker can eavesdrop on the traffic transmitted between a client and an access point. Eavesdropping can be achieved by configuring the wireless network interface in monitor mode, allowing the attacker to monitor all network traffic received on a wireless channel. As wireless networks operate over radio waves, the network inherently exposes transmitted data to potential attackers and side-channel analysis. The exposing of transmitted data strengthens the issue of information leakage through side channels, such as the size of the messages. The issue is further enhanced as the equipment needed is easily accessible and affordable. Consequently, mitigation techniques, such as padding, must be used to mitigate information leakage of encrypted traffic.

Modern Wi-Fi allows padding in *frame aggregation*, combining multiple data frames into one potentially larger payload. This single frame is transmitted as one unit over lower-level protocols. At the data link layer, frame aggregation is called aggregate MAC service data unit (*A-MSDU*). The concept of A-MSDU will be further explained in [Section 2.2.6](#). While A-MSDU enhances throughput, it could also obscure traffic patterns, mitigating website fingerprinting and other side-channel attacks.

### 1.2.2 Research questions

The thesis aims to answer the following questions:

**Q.1** What is the baseline accuracy of an attacker performing a website fingerprinting attack on encrypted Wi-Fi traffic, considering different traffic features and varying conditions?

**Q.2** How does A-MSDU frame aggregation impact the accuracy of a website fingerprinting attack on encrypted Wi-Fi traffic?

**Q.3** How does the accuracy of a website fingerprinting attack on encrypted Wi-Fi traffic compare to the trivial success probability when distinguishing between different data types within a single domain?

## 1.3 Contribution

This thesis's main contribution is demonstrating how an attacker can exploit information leakage of encrypted Wi-Fi frames to execute a website fingerprinting attack. Understanding the impact of such leakage is necessary to determine the severity of the issue. Insights gained from such studies could help in the future development of countermeasures to enhance user privacy online. The accuracy of the website fingerprinting attack executed should be interpreted as a baseline, considering the study's limitations.

The thesis considers a setting where an adversary observes raw encrypted 802.11 frames sent between a client and an access point. The attacker does not need to be associated with the access point or be able to read the decrypted traffic. The only requirement is the ability to observe the headers and the encrypted frames. The aim is to perform the website fingerprinting attack with a relatively high accuracy using a machine learning algorithm, focusing on traffic features related to the encrypted frames. Exploring the impact of website fingerprinting attacks on encrypted Wi-Fi traffic provides the following.



**Theoretical security definitions.** Examining features of encrypted 802.11 frames could enhance security definitions for length-hiding encryption. This information could contribute to the future development of theoretical security.

**Privacy risks.** Performing a website fingerprinting attack illustrates an adversary's capability to invade users' privacy. Suppose the success rate is high. In that case, this could have consequences in other security areas. Insight into the privacy risks is important, as the equipment needed to perform such an attack is easily accessible and affordable.

**Future development of countermeasures.** Understanding the limitations of security protocols' ability to hide information origination from encrypted 802.11 frames can contribute to developing new countermeasures against fingerprinting attacks.

**Capabilities of machine learning.** Acquiring a better understanding of the effectiveness of using machine learning algorithms in website fingerprinting attacks could provide valuable information that could be considered in future discussions of using such algorithms in various parts of society.

The thesis explores three different website fingerprinting scenarios, termed 'simple PDF fingerprinting'<sup>1</sup>, 'Wikipedia fingerprinting,' and 'domain fingerprinting.' Each scenario considers with and without the use of A-MSDU and uses machine learning to assess the accuracy of the website fingerprinting attack. A comparison between the attack's accuracy and the trivial success probability will also be provided in the 'simple PDF fingerprinting' scenario. The scenarios considered can be summarized as follows.

- **Simple PDF fingerprinting.** This scenario considers a closed environment where the client accesses a primitive web page developed by the author. The web page features a simple front-end hosting numerous PDF files. The attacker's objective is to identify the exact PDF file accessed by the user.
- **Wikipedia fingerprinting.** The attack progresses from the 'simple PDF fingerprinting' scenario and centers on a client accessing Wikipedia pages within a selected subset of available pages. The attacker aims to determine which Wikipedia page the client is accessing, given a chosen subset.
- **Domain fingerprinting.** The scope broadens, as the attacker has no prior knowledge of the website the client is accessing. The attacker's

---

<sup>1</sup>Technically, only the last two scenarios are technically website fingerprinting attacks, as 'simple PDF fingerprinting' involves distinguishing between data within a single web page. However, this scenario is included as a baseline scenario before introducing more complex situations.

objective is to identify the particular domain accessed by the client, given a subset of popular websites.

## 1.4 Related work

This section discusses relevant research on the failure of padding techniques as a countermeasure against fingerprinting attacks and fingerprinting LTE/4G traffic.

### 1.4.1 Failure of countermeasures

Previous research [9] evaluated the effectiveness of different padding techniques employed by modern protocols such as IPsec, TLS, and SSH. These protocols have standardized padding countermeasures [21, 24, 40] to address issues such as traffic analysis attacks and to prevent message length leakage [8, 20, 32].

The research considered HTTP traffic over encrypted tunnels. In this scenario, the attacker possesses prior knowledge of the potential websites a user might visit and the capability of obtaining network traffic metadata. The attacker uses this information to train and test machine learning algorithms on traffic traces, a record of the timings and lengths of ciphertexts. The attacker's goal is to determine the website the user is accessing.

There are several ways of choosing a padding strategy. Due to this, the researchers [9] picked various countermeasures and categorized them into three distinct categories: SSH/TLS/IPSec-motivated countermeasures, other padding-based countermeasures, and distribution-based countermeasures. The different types can be viewed in Fig. B.1. These are the countermeasures evaluated in the research paper.

By comparing the most effective padding techniques from each category, their research shows that the attack's accuracy drops only to roughly 90% for the first two categories, still recognizing the websites with a relatively high accuracy. Even for the best countermeasure in their scenario, the average accuracy for the various machine learning algorithms employed in the research paper only drops to around 85%. Their results suggest that the different padding techniques obfuscate some features that can improve security, but an attacker can still identify the different websites with a relatively high accuracy.

Given the scenario, they conclude that none of the nine countermeasures effectively prevent website fingerprinting attacks. The results indicate that an attacker can identify information originating from coarse features within encrypted traffic with relatively high accuracy. The obtained accuracy is

achievable even though the techniques are designed to, among other things, obscure identifiable traffic features through the use of padding and dummy packet insertion. The authors conclude that one should not rely solely on the examined countermeasures to counter website fingerprinting attacks.

Previous research [12] focusing on length-hiding encryption has concluded that even a modest amount of padding can substantially reduce the adversary's advantage at the cost of a slight increase in bandwidth overhead. The choice of padding has been further researched in [7], concluding that sampling the padding length from a Gaussian distribution is favorable for two reasons. Firstly, the Gaussian distribution holds a significant role in various aspects of cryptography and, consequently, has received much scrutiny in different settings. Secondly, suppose the ability to hide the message length depends on the padding technique. In that case, the adversary's goal is correlated to being able to distinguish between two distributions given a sample.

Consequently, it appears advantageous to choose a distribution that concentrates their probability mass in the center and exhibits minimal tails. However, it is still necessary to prevent the leakage of padding size information through side-channel attacks. On the other hand, research [9] indicates that relying on padding techniques alone does not suffice, as attackers can still perform website fingerprinting attacks with relatively high accuracy.

### 1.4.2 Fingerprinting LTE/4G traffic

Previous work [22] investigated the impact of fingerprinting attacks on encrypted Long Term Evolution (LTE)/4G traffic. Here, the attacker performed a fingerprinting attack exploiting metadata information, having no access to the payload itself or the metadata from the network layer, such as the IP header information of a packet. It is also assumed that the attacker does not know any internal LTE identities but can distinguish multiple device connections. Distinguishing connections allows the adversary to map traffic to connections. The paper is restricted in some areas, such as limited to state-of-the-art machine learning algorithms that exclude deep learning. However, the paper intends to determine if website fingerprinting is possible on LTE traffic.

As highlighted in the paper, there are differences between usual website fingerprinting and fingerprinting LTE traffic. Conventional attacks often require control over physical nodes and adversarial access to the network infrastructure. A passive LTE adversary uses a downlink sniffer to access all transmissions, which uses affordable equipment and can hardly be backtracked. As a consequence, the website fingerprinting attack performed does not require

control of any network device, such as a switch or router. Secondly, the metadata information set is different when obtaining traffic.

The research experiment consisted of two parts. First, a private LTE network was created for a controlled lab environment. In this environment, the researchers studied the different factors for website fingerprinting. Based on these factors, the researchers chose an machine learning algorithm. Finally, they turned to a real-world scenario after testing in a controlled lab environment.

The results [22] show that in the controlled lab environment, the attack's success rate was between 92% to 95%, while obtaining a success rate of roughly 90% in the real-world scenario. They conclude that the experiment proves the feasibility of fingerprinting LTE traffic.

### 1.5 Outline of the thesis

**Chapter 2** defines the necessary definitions and terminology for understanding security definitions. It will address fundamental wireless network terminology and provide detailed information about protocols used in wireless networks.

**Chapter 3** formally outlines the framework on which the experiments are based, including the research design, setup, and analytical approach employed in the work. All definitional choices that have been made are discussed and justified in this chapter.

**Chapter 4** presents the results of the website fingerprinting attack across different scenarios. It includes visual illustrations of the attack's success rate, comparing the outcomes with and without aggregation. All relevant results obtained will be addressed in this chapter.

**Chapter 5** discusses the experimental results and implications of the information gained in **Chapter 4**. It also offers suggestions for future work.

**Chapter 6** concludes this thesis by answering the research questions presented in **Chapter 1 Section 1.2.2**.

**Appendix A** and **Appendix B** contains supplementary figures and additional information. It primarily illustrates the format of protocols and provides more detailed insights from previous research.

## CHAPTER 2

# Theoretical background

This chapter will provide the theoretical background necessary for understanding this thesis.

The chapter begins by discussing cryptography in [Section 2.1](#), followed by an overview of the IEEE 802.11 standard in [Section 2.2](#). Finally, machine learning will be covered in [Section 2.3](#).

## 2.1 Cryptography

Cryptography is a crucial part of modern network communication, as it can provide both confidentiality and data integrity to achieve data privacy. Since modern cryptographic algorithms and protocols were created and standardized in the 1970s, researchers have also tried to define security formally.

### 2.1.1 Symmetric encryption schemes

A symmetric encryption scheme can be defined as a tuple

$$\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$$

where  $\Sigma.\text{KeyGen}$  is an algorithm to generate a secret key,  $\Sigma.\text{Enc}$  is the encryption algorithm and  $\Sigma.\text{Dec}$  is the decryption algorithm. It should be stated that the  $\Sigma.\text{KeyGen}$  algorithm and the  $\Sigma.\text{Enc}$  algorithm can be randomized, while the  $\Sigma.\text{Dec}$  algorithm must be deterministic. Correctness requires that given a key  $K$  and a message  $M$ ,

$$\text{Dec}(K, \text{Enc}(K, M)) = M$$

### 2.1.2 Standard security definitions

Standard cryptographic security definitions for symmetric encryption schemes consider a security experiment in which the properties of the definition are defined in a game context. In this context, the adversary's objective is to demonstrate the capability of breaking the encryption scheme with a non-negligible advantage while following the rules of the game.

Intuitively, the game starts with a challenger  $\mathcal{C}$  flipping a coin. The outcome of flipping the coin remains a secret for the adversary, and the value is stored in a variable 'b'. The value will remain the same throughout the experiment. The adversary aims to determine the value 'b' through the encryption scheme. In other words, the adversary wants to determine the outcome of the coin flipping by granting the adversary access to one or more oracles, where each oracle has a specific function. What the adversary has access to depends on the capabilities of the encryption scheme.

While many different cryptosystems exist, standard definitions are often used in cryptography. These definitions describe specific desired properties present in many cryptosystems today. An example of this is an instance of indistinguishability under chosen-ciphertext attack, focusing on

<u>Initialize():</u> 1: $b \xleftarrow{\$} \{0, 1\}$ 2: $K \xleftarrow{\$} \Sigma.\text{KeyGen}$ 3: $L \leftarrow \emptyset$ 4: <b>return</b>  <u><math>\mathcal{E}(m)</math>:</u> 1: <b>return</b> $\Sigma.\text{Enc}(K, m)$  <u><math>\mathcal{D}(c)</math>:</u> 1: <b>if</b> $c \in L$ : 2: $m' \leftarrow \perp$ 3: <b>else</b> : 4: $m' \leftarrow \Sigma.\text{Dec}(K, c)$ 5: <b>return</b> $m'$	<u>Challenge(m):</u> 1: $c \xleftarrow{\$} \Sigma.\text{Enc}(m)$ 2: $r \xleftarrow{\$} \{0, 1\}^{ c }$ 3: <b>if</b> $b = 0$ : 4: $c' = c$ 5: <b>else</b> : 6: $c' = r$ 7: $L \leftarrow L \cup \{c'\}$ 8: <b>return</b> $c'$  <u>Finalize(b'):</u> 1: <b>Output</b> ( $b' = b$ )
--	---

Figure 2.1: The IND\$-CCA security game.

indistinguishability between a ciphertext string and a random string (IND\$-CCA).

More formally, IND\$-CCA security can be structured as a game-based model [4], considering a game with a start procedure, optional oracles (each one a procedure), and an output procedure. The game involves an adversary, where an encryption scheme is considered secure if no computationally efficient adversary can win the game with a probability greater than a negligible amount. The IND\$-CCA security game is illustrated in Fig. 2.1 and consists of the following.

**Assigning a value b:** The game starts by running the ‘initialize’ procedure, which chooses a value  $b \in \{0, 1\}$  uniformly at random. The outcome determines the behaviour of the ‘challenge’ procedure.

**Generating a secret key:** It is assumed that the game has a  $\Sigma.\text{KeyGen}$  algorithm to securely generate a secret key,  $K$ , in order to encrypt and decrypt queries. The adversary does not have access to this key.

**Oracles:** The adversary  $\mathcal{A}$  gets access to two oracles: an encryption oracle and a decryption oracle. By querying a message  $m$  to the encryption oracle, the oracle will respond with the encryption of a message  $m$ . By querying a ciphertext  $c$  to the decryption oracle, the oracle will respond with a message  $m'$ . More details will be provided when explaining the function of the list.



**Initializing an empty list:** The start procedure initiate an empty list called  $L$ . When the encryption oracle returns a response to the adversary, the encrypted string will be appended to the list. When the decryption oracle gets a query, it will check to see if the query is already in the list  $L$ . If it is, the procedure will return  $\perp$ , indicating that the ciphertext was invalid. If not, the adversary will receive the decryption of the query  $c$ .

If the secret bit is 0, querying a message  $m$  will result in the ‘challenge’ procedure returning the encryption of the message  $m$ . If the secret bit is 1, then querying a message  $m$  will result in the ‘challenge’ procedure returning a random string  $r$  chosen uniformly at random, where the length of the message  $r$  equals the length of the encryption of message  $m$ . Given the oracles, the adversary  $\mathcal{A}$  now has to determine what the value of the secret bit is using the ‘finalize’ procedure. How well an adversary  $\mathcal{A}$  does against the encryption scheme is called the advantage, defined as

$$\text{Adv}_{\Sigma}^{\text{IND\$-CCA}}(\mathcal{A}) = |2 \cdot \Pr [\text{IND\$-CCA}(\mathcal{A}) = 1] - 1|$$

where  $\Pr [\text{IND\$-CCA}(\mathcal{A}) = 1]$  is the probability that the adversary  $\mathcal{A}$  can determine the value of the secret bit, for a given encryption scheme  $\Sigma$ . The value of the advantage is a number scaled between 0 and 1. The adversary is doing poorly if the advantage is close to 0, meaning that the algorithm is considered secure if the probability of an adversary performing a successful attack is negligible. The advantage is doing well if the advantage is close to 1.

Some things should be mentioned. Firstly, the reason for the restriction of messages that can be decrypted is that the adversary should not be able to send  $c$  to the decryption oracle and receive back  $m'$  if  $c$  is the return value of a previous query to the encryption oracle. If the restriction on the messages to be decrypted is not present, the adversary can determine if the secret bit is 0 or 1 with certainty. Secondly, there is no guarantee that the decryption of  $c$  makes any sense in practice.

While these standard security notions aim to capture what security means theoretically, they do not necessarily reflect a real-world scenario. As network traffic involves messages of different lengths, the security game IND\\$-CCA [Fig. 2.1] introduces a crucial restriction where the length of messages observed by the adversary in the theoretical setting is equal. This restriction is introduced in the encryption oracle  $\mathcal{E}$  where the oracle either returns  $c_0$  or  $c_1$ , where  $c_1$  is a random string of the same length as  $c_0$ . If the encryption scheme uses counter mode to encrypt, then the produced ciphertext has the same length as the plaintext. As a result, the encryption scheme may be theoretically secure

even though the observed ciphertext produced by  $\mathcal{E}$  leaks information about the plaintext lengths, illustrating how the conventional security definitions fail to capture different message lengths.

Padding involves appending additional data to a message prior to encryption. It is often used by symmetric encryption algorithms such as block cipher to extend the plaintext lengths to a multiple of the block size. Padding could be used as a mitigation technique to counteract information leakage by padding the plaintext to conceal the message length. Doing this would complicate the attacks on encrypted network traffic, taking advantage of the information revealed through side channels.

There are drawbacks to employing length-hiding padding. While entirely concealing the message length may seem achievable theoretically, the unavoidable bandwidth usage increase must be considered. The increase in bandwidth due to padding has been previously researched [36] where the scenario permits the attacker to select messages of varying lengths, demonstrating that achieving negligible distinguishability using standard cryptographic security definitions necessitates exponentially sized padding. The paper essentially showed that it is impossible to hide the length of the messages for arbitrary message distribution efficiently, indicating that practically concealing the entire message length is not feasible. The trade-off between a sufficient level of security and minimizing the bandwidth overhead must be considered.

### 2.1.3 Length-hiding security

Length-hiding encryption (LHE) was introduced by Paterson, Ristenpart, and Shrimpton [33] where the encryption algorithm used by an application includes an additional parameter, denoted as  $\ell$ . This parameter specifies the padding to be applied to a given message and is based on a different value,  $\Delta$ , derived from the left-or-right security experiment. The value of  $\Delta$  has the limitation that  $0 \leq \|m_0\| - \|m_1\| \leq \Delta$  given two messages  $m_0$  and  $m_1$ . The introduction of  $\ell$  chooses messages by the application to be constrained by the selection of the value  $\Delta$ , limiting the range of messages that can be chosen.

A solution to this problem is to treat the length-hiding parameter  $\ell$  as a fixed parameter and to make ciphertexts dependent on both the parameter  $\ell$  and the size of encrypted messages. Treating  $\ell$  as a fixed system parameter is the key idea in previous research done by Gellert et al. [12] and is achieved by establishing a modified security model building upon the concept of LHE [33]. This security model is independent of any specific application and can still capture complex application settings.

The authors [12] start by expanding the previous definition of symmetric-key encryption to include the length-hiding parameter  $\ell$ . First, they define a deterministic function mapping the length of plaintexts to specific ciphertext lengths. This function is called **pad** and is defined by taking the plaintext and the parameter  $\ell$  as input and outputs a padded plaintext such that

$$|\mathbf{pad}(m, \ell)| = \left\lceil \frac{|m|}{\ell} \right\rceil \cdot \ell$$

Defining a padding function introduces a trade-off between increasing bandwidth overhead from the length-hiding padding and security against message length attacks. Second, they expand the previous definition of an encryption scheme such that

$$\text{Enc}^{(\ell)}(K, m) = \text{Enc}(K, \mathbf{pad}(m, \ell)) \text{ and } \text{Dec}^{(\ell)}(K, c) = \mathbf{pad}^{-1}(\text{Dec}(K, c))$$

This approach allows the authors to define a symmetric-key encryption scheme that also captures ‘perfect’ length-hiding padding, meaning that all ciphertexts possess identical lengths. As a result, no information about the length of the plaintexts is leaked. Further, they introduce several terms to concretely assess the impact of length-hiding encryption on the security of a given application. These terms are defined as follows <sup>1</sup>.

**Definition 2.1.** Let  $\mathcal{A}$  be an adversary. Given an encryption scheme  $\Sigma$ , a message distribution  $\mathcal{M}$ , and a function  $\mathcal{P} : (\{0, 1\}^*)^t \rightarrow \{0, 1\}^*$  which specifies the information an adversary wants to learn given  $t$  messages, then for any adversary, the real success probability of  $\mathcal{A}$  in game  $(\Sigma, \mathcal{M}, \mathcal{P})$ -CCA [Fig. 2.2] is defined as

$$\mathbf{RealSucc}(\Sigma, \mathcal{M}, \mathcal{P}, \mathcal{A}) := \Pr[(\Sigma, \mathcal{M}, \mathcal{P})\text{-CCA}(\mathcal{A}) = 1]$$

The real success probability represent the actual likelihood that an adversary successfully executes an attack.

**Definition 2.2.** Let  $\Sigma$  be an encryption scheme,  $\mathcal{M}$  be a message distribution,  $\mathcal{P} : (\{0, 1\}^*)^t \rightarrow \{0, 1\}^*$  be a function specifying the information an adversary wants to learn, and let  $\mathcal{S}$  be an algorithm. Now let  $|c_i^*|$  be the size of the ciphertexts, where  $c_i^* \xleftarrow{\$} \text{Enc}(K, m_i^*)$  for  $i \in \{1, 2, \dots, t\}$ , and

---

<sup>1</sup>The definitions are based on the work in [12] considering multi-key security. The definitions have been modified to suit the context of this thesis considering a single key  $K$ .

<u>Initialize():</u> 1: $K \xleftarrow{\$} \Sigma.\text{KeyGen}$ 2: $\text{challenged} \leftarrow \text{false}$ 3: $L \leftarrow \emptyset$ 4: <b>return</b> $(\mathcal{M}, \mathcal{P})$  <u><math>\mathcal{E}(m)</math>:</u> 1: <b>return</b> $\Sigma.\text{Enc}(K, m)$  <u><math>\mathcal{D}(c)</math>:</u> 1: <b>if</b> $c \in L$ : 2: $m' \leftarrow \perp$ 3: <b>else</b> : 4: $m' \leftarrow \Sigma.\text{Dec}(K, c)$ 5: <b>return</b> $m'$	<u>Challenge():</u> 1: <b>if</b> $\text{challenged} = \text{true}$ : 2: <b>return</b> $\perp$ 3: $\text{challenged} \leftarrow \text{true}$ 4: $(m_1^*, m_2^*, \dots, m_t^*) \xleftarrow{\$} \mathcal{M}$ 5: <b>for</b> $i \in \{1, 2, \dots, t\}$ : 6: $c_i^* \xleftarrow{\$} \mathcal{E}(m_i^*)$ 7: $L \leftarrow L \cup \{c_i^*\}$ 8: <b>return</b> $(c_1^*, c_2^*, \dots, c_t^*)$  <u>Finalize(p):</u> 1: <b>Output</b> $(\mathcal{P}(m^*) = p)$
--	---

Figure 2.2: The  $(\Sigma, \mathcal{M}, \mathcal{P})$ -CCA security game.

$m^* = (m_1^*, m_2^* \dots, m_t^*) \xleftarrow{\$} \mathcal{M}$ . The trivial success probability with respect to  $(\Sigma, \mathcal{M}, \mathcal{P})$  is defined as

$$\text{TrivSucc}(\Sigma, \mathcal{M}, \mathcal{P}) = \max_S \Pr[\mathcal{S}(\mathcal{M}, \mathcal{P}, (|c_i^*|)_{i \in [t]}) = \mathcal{P}(m^*)]$$

The trivial success probability represents the probability that an adversary can trivially obtain information about the plaintext from the length of ciphertexts given knowledge of the message distribution, without ‘breaking’ the underlying encryption scheme. Note that the adversary does not have access to the actual ciphertexts.

**Definition 2.3.** Let  $\mathcal{M}$  be a message distribution,  $\mathcal{P} : (\{0, 1\}^*)^t \rightarrow \{0, 1\}^*$  be a function specifying the information an adversary wants to learn, and let  $\mathcal{S}$  be an algorithm. Given  $m^* = (m_1^*, m_2^* \dots, m_t^*) \xleftarrow{\$} \mathcal{M}$ , the most likely probability output of  $\mathcal{P}$  on input  $m^*$  is defined as

$$\text{PrMostLikely}(\mathcal{M}, \mathcal{P}) = \max_S \Pr[\mathcal{S}(\mathcal{M}, \mathcal{P}) = \mathcal{P}(m^*)]$$

The most likely probability is introduced to capture any prior information that an adversary might obtain through leakage by the message distribution  $\mathcal{M}$ . The probability is scaled to a value between 0 and less than 1.

**Definition 2.4.** Let  $\Sigma$  be an encryption scheme,  $M$  be a message distribution,  $\mathcal{P} : (\{0, 1\}^*)^t \rightarrow \{0, 1\}^*$  be a function specifying the information an adversary wants to learn. The trivial advantage with respect to  $(\Sigma, M, \mathcal{P})$  is defined as

$$\mathbf{TrivAdv}(\Sigma, M, \mathcal{P}) = \frac{\mathbf{TrivSucc}(\Sigma, M, \mathcal{P}) - \mathbf{PrMostLikely}(M, \mathcal{P})}{1 - \mathbf{PrMostLikely}(M, \mathcal{P})}$$

The trivial advantage is defined as the difference between the trivial success probability and the most likely probability. The advantage is scaled to a value between 0 and 1.

The  $(\Sigma, M, \mathcal{P})$ -CCA security game consists of a key  $K$  being generated where the adversary gets a fixed  $M$  and a given  $\mathcal{P}$ . The adversary observes  $t$  encrypted messages using the key  $K$ , where the messages are sampled from the message distribution  $M$ . The adversary must now determine  $\mathcal{P}(m^*)$ . In the case of website fingerprinting, an adversary wants to determine what website a user is accessing. The data exchanged between the user and the webserver would be  $m^* \xleftarrow{\$} M$  where the adversary wants to determine the website  $\mathcal{P}(m^*)$ . The authors consider  $M$  and  $\mathcal{P}$  such that  $\exists m, m' \in M$  such that  $\Pr[m^* = m] > 0$ ,  $\Pr[m^* = m'] > 0$  and  $\mathcal{P}(m) \neq \mathcal{P}(m')$ <sup>2</sup>.

The authors [12] continue to demonstrate that under certain assumptions regarding the security of the encryption scheme,  $\mathcal{A}$ 's real success probability cannot exceed the trivial success probability significantly. They use this to state the following theorem.

**Theorem 2.5** ([12], Thm 8). *Let  $M$  be a message distribution,  $\mathcal{P} : (\{0, 1\}^*)^t \rightarrow \{0, 1\}^*$  be a function specifying the information an adversary wants to learn, and  $\mathcal{A}, \mathcal{B}$  be two adversaries. Then for any encryption scheme  $\Sigma$  satisfying that for any  $m \in \{0, 1\}^*$ ,  $K, K' \in \{0, 1\}^k$ ,  $\text{Enc}(K, m)$  distributes identically to  $\text{Enc}(K', m)$ , an adversary  $\mathcal{B}$  can be constructed such that*

$$\mathbf{RealSucc}(\Sigma, M, \mathcal{P}, \mathcal{A}) \leq \mathbf{Adv}_{\Sigma}^{\text{IND\$-CCA}}(\mathcal{B}) + \mathbf{TrivSucc}(\Sigma, M, \mathcal{P})$$

Thus, the real success probability can be upper bound by the sum of the trivial success probability and the security assumption of the encryption scheme. As a result, under the assumption that the encryption scheme is conventionally secure, the authors can now focus primarily on reducing the trivial success probability through appropriate padding techniques, allowing them to quantitatively measure the effectiveness of length-hiding encryption in concealing information about the message length.

---

<sup>2</sup>This gives  $\mathbf{PrMostLikely}(M, \mathcal{P}) < 1$  and ensures that it is not trivial to predict  $\mathcal{P}$  with respect to  $M$  with probability 1.

The choice of padding to be used is not trivial. According to previous research [36], if the adversary can only observe a single ciphertext, then sampling the padding length from a uniform distribution is a nearly optimal approach, given some restrictions. However, this does not reflect a real-world scenario where a passive attacker can observe encrypted traffic. Here, the attacker would observe information that is sent in clear text or can be calculated, such as metadata. Given the existence of this, concentrating on particular features of encrypted traffic could prove beneficial if future work enhances the effectiveness of the padding used.

As stated by Degabriele [7], there are two reasons to focus solely on length-hiding security and ciphertext lengths: specific distinct attacks differ significantly from website fingerprinting attacks such as CRIME and BREACH, and the uncovering of any length-hiding mechanisms could potentially make a difference for the development of more effective website fingerprinting countermeasures. However, the former necessitates the ability to select specific messages, which is not typically required for website fingerprinting attacks.

In order to focus solely on length-hiding security, Degabriele defined a unified security model that combines the strength of previously defined security models [33, 36]. The aim is to evaluate the efficacy of different length padding strategies [Fig. B.1] while capturing potential adverse interactions with cryptographic components.

One of the most central parts of the unified security model is the definition of the composition theorem [7, Thm 3.3]. This theorem resembles [Theorem 2.5](#) and states that when considering the unified security model, the advantage of an attacker is upper bounded by the sum of two things: the advantage of an attacker under something called ‘channel simulateability,’ and the attacker’s ability in analyzing the statistical properties of a probability distribution  $\mathcal{P}$ , from which padding lengths are drawn. Consequently, as long as the encryption scheme satisfies channel simulateability, the advantage of an attacker is reduced to the amount of information that is leaked through the probability distribution. If the encryption scheme satisfies the condition, the authors can primarily focus on the padding scheme in use. All details regarding the different components of the composition theorem and technicalities are explained in detail in [7].

By combining both papers [7, 12], research indicates that under certain assumptions regarding the security of the encryption scheme, there is a direct correlation between reducing the trivial success probability [[Theorem 2.5](#)] and the encryption scheme’s ability to hide information about the message length. The composition theorem states that the ability to hide this information depends on the padding scheme in use. By the same argument by Degabriele [7], focusing

on hiding the message length could be a stepping stone towards more efficient countermeasures against website fingerprinting. However, as it is now, only focusing on the padding technique to hide the message length will not prevent website fingerprinting based on previous research.

## 2.2 IEEE 802.11

IEEE 802.11 standard is a layer two protocol primarily developed to connect devices over a wireless local area, either in home or office environment. Today, IEEE 802.11 has developed to be the most widely deployed WLAN technology, where the term IEEE 802.11 and Wi-Fi are used interchangeably in literature [3].

While different frames are being sent over the Internet for different purposes, this thesis focuses on data frames containing application-level data. It focuses specifically on the encrypted traffic itself and not the other components of Wi-Fi traffic, such as the generated and transmitted keys used for encryption. It is assumed that the CCMP-128 is utilized for encryption.

### 2.2.1 Architecture

The IEEE 802.11 defines two kinds of services [10]:

**Basic Service Set (BSS):** A basic service set consists of stationary or mobile wireless stations and an access point (AP).

**Extended Service Set (ESS):** An extended service set consists of two or more BSSs with APs. The BSSs are connected through a distribution system. The distribution system will be further discussed later on.

While the IEEE 802.11 standard supports different modes of operation depending on the network topology, the infrastructure mode is by far the most common [6]. Here, the clients only communicate through an access point. The thesis only considers infrastructure mode.

In addition to the services, IEEE 802.11 defines four main physical components in an 802.11 network [2, 11]. These are the following:

**Stations:** Stations are computing devices with wireless network interfaces, like laptops and smartphones. In wireless/wired networks, these are commonly abbreviated as STAs. Other names for stations are clients, nodes, users, originators, transmitters, and receivers. Stations will be referred to as clients in this thesis.

**Wireless medium:** Data structured at layer two are often called frames. A wireless medium transfers the frames within a network between two devices. Three physical layers were initially standardized: two radio frequencies and one infrared.

**Access points:** Devices that work as bridges between the wireless 802.11 and wired networks. An access point also manages the transmission of frames between clients within the network.

**Distribution system:** The distribution system is a wired or wireless network that forwards the frames to their destination. It works as the logical component of the 802.11 standard, ensuring the connection to the network. In a wireless network, this is used as different access points must communicate with each other to track the movements of the mobile clients. Ethernet is commonly used for these components.

### 2.2.2 802.11 MAC frame format

There are three 802.11 frame types: management, control, and data frames. Management frames are frames meant for establishing and maintaining communications. An example of a management frame is the beacon frame, used to announce the presence of a WLAN and provide a signal for synchronizing communications between devices. Control frames are frames used to ‘control’ the physical medium and assist in delivering other frames. An example of a control frame is the ACK frame, which is sent from the receiver to the sender to verify that the receiver has received the given frame. The last frame type is data frames. *Data frames* contain higher-level protocol data, such as web pages.

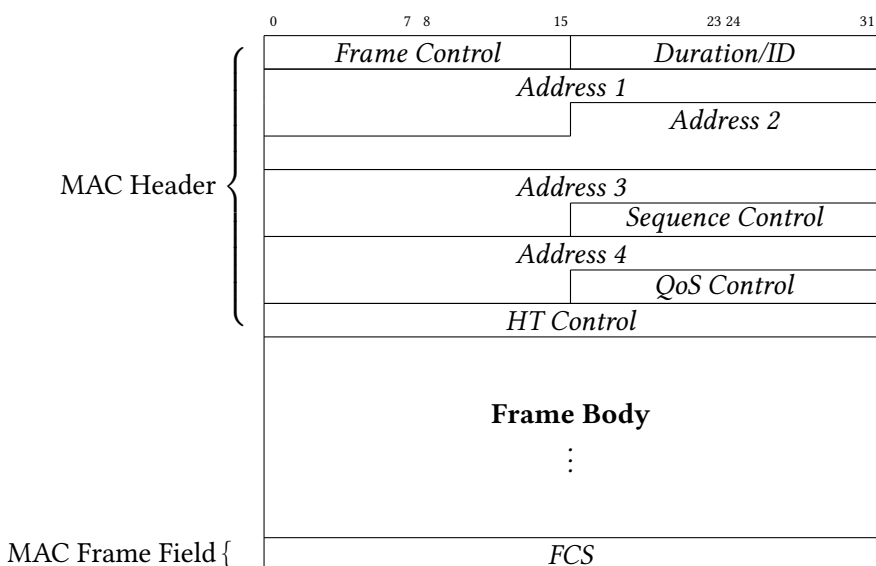
The MAC layer frame consists of 11 fields, as shown in [Fig. 2.3](#). These fields combine to form a message protocol data unit (MPDU). The 11 fields are the following [\[1, 2, 10, 23\]](#).

**Frame control:** The frame control field is a two-byte long field defining the type of frame and control information. A specific bit will be set in this field if the frame is encrypted. The subfields under the frame control field can be viewed in [Fig. A.1](#).

**Duration/ID:** The duration ID field is a two-byte long field carrying the Network Allocation Vector (NAV) value. The value restricts the time of access to the medium.

**Addresses:** There are four address fields, each six bytes long. The meaning of these address fields depends on specific subfields in the frame control.





**Figure 2.3:** The IEEE 802.11 Generic Frame Format.

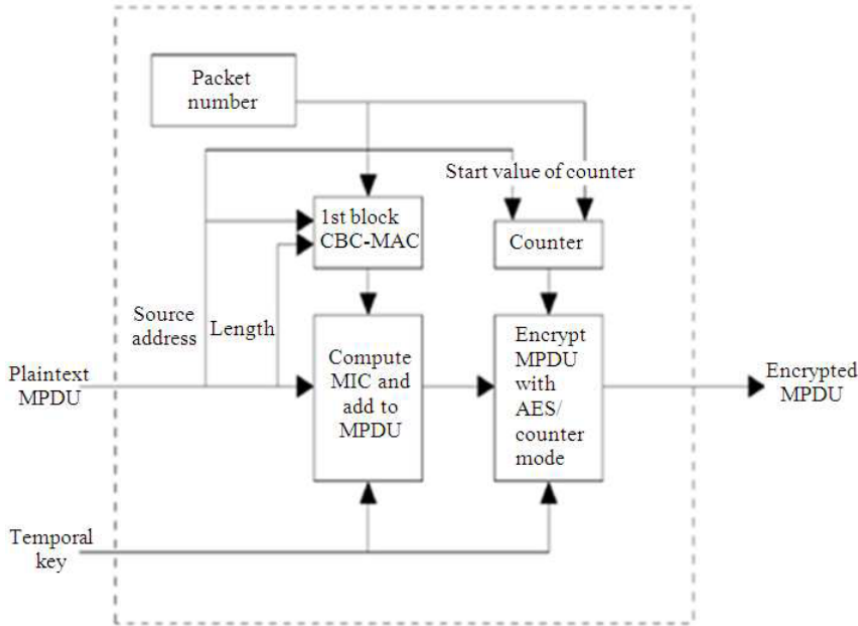
**Sequence control:** The sequence control is a two-byte long field. The first four bits define the fragment number, while the remaining 12 define the sequence number. The sequence number is the same in all fragments.

**Quality of service (QoS) control:** The presence of this field depends on the setting of the quality of service subfield found in the Subtype field. This field identifies the traffic category the frame belongs to and specifies the packet's priority. Information on whether A-MSDU is enabled is set in this field.

**HT control:** The presence of this field depends on the setting of the +HTC/Order subfield found in the Frame Control field. This field is used to support other features and extend the capabilities of the standard, such as channel bandwidth.

**Frame body:** The frame body is a variable length field containing information specific to the given frame type and subtype. While the size varies, the minimum length of the frame body is zero bytes, with the maximum 802.11 frame body length being approximately 2300 bytes.

**FCS:** The WLAN FCS field is a four-byte long field containing a four-byte long cyclic redundancy check (CRC). The CRC is an error-detecting code



**Figure 2.4:** The CCMP encapsulation of a single block. Figure from [37].

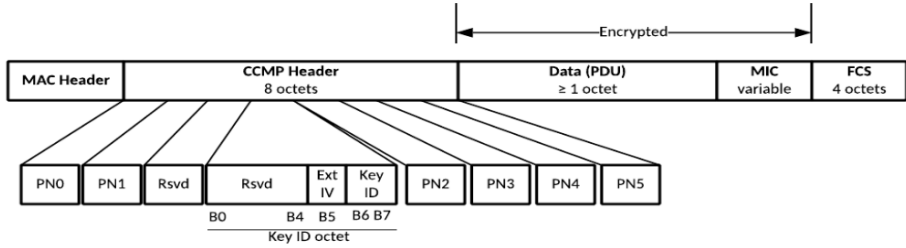
that detects corrupted data during transmission. This field will provide information on higher-level protocols, such as TCP.

The thesis will not write extensively about the different subfields, even though most frame fields are divided into new subfields. More detailed information is provided in the different fields and subfields [1].

### 2.2.3 WPA2

One of the most common wireless security standards today is the 802.11i standard, commonly referred to as WPA2, although technically incorrect [6]. The wireless security standard is now incorporated into the 802.11 standard and exists to implement security features for WLANs.

There are two types of WPA2: Enterprise and Personal. WPA2-Enterprise is designed for use in organizations where the security standard depends upon the IEEE 802.1X standard to authenticate in addition to generating the main cryptographic key used to secure wireless network traffic (4-Way-Handshake protocol) [31]. On the other hand, WPA2-Personal uses pre-shared keys (PSK) for authentication and is designed for home use. This PSK uses an 8 to 63-character



**Figure 2.5:** The expanded CCMP MPDU format. Figure from [1].

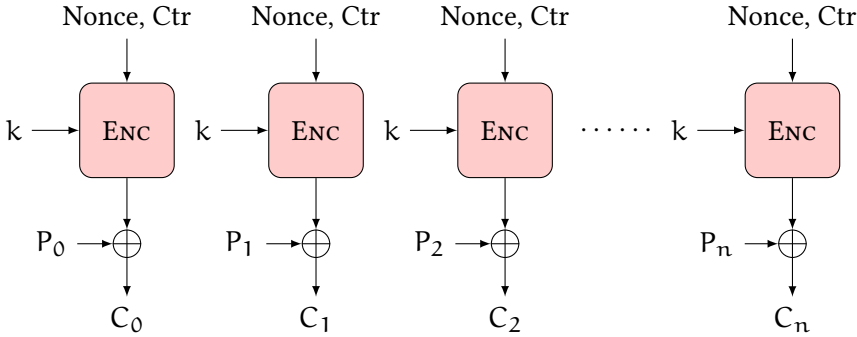
shared key. However, the temporal keys that encrypt the application data are derived from some values and the original shared key. These values are easily captured during over-the-air transmissions between the client and the access point [29], and this is a problem with WPA2-PSK.

Once the cryptographic keys are generated and transmitted, encryption is used to secure the application data during transmission. Encryption is optional in 802.11 WLANs but highly desired as a passive attacker can read the traffic within the network without it. While different protocols exist, WPA2 uses Counter Mode CBC-MAC Protocol (CCMP) [6], enabling a single key for encryption and authentication.

#### 2.2.4 CCMP

The Counter Mode CBC-MAC protocol (CCMP) uses the AES symmetric encryption algorithm with a 128-bit block size, and either a 128-bit key (CCMP-128) or a 256-bit key (CCMP-256) [1]. In addition, the protocol requires a new temporal key for every session and a unique value, a nonce, for each data frame encrypted by a given temporal key. This nonce is often called the packet number (PN), a 48-bit value [19]. The CCMP consists of two components: CBC-MAC and CTR, where cipher block chaining (CBC) is used for integrity and authenticity, and counter mode (CTR) for confidentiality. Both CBC and CTR are two AES modes of operation.

The process of encrypting MPDU using CCMP starts by using the AAD, the CCM nonce, and the temporal key [1, 19, 35, 37]. These are used to generate the encryption of the plaintext MPDU together with the encrypted message integrity check (MIC). More information regarding the encryption process will be discussed in Section 2.2.5. The MIC protects against replay attacks, as well as the CCMP header and parts of the MAC header, where the value is appended to the MPDU. For CCMP-128, this value is 8 bytes long. The process of encrypting



**Figure 2.6:** Diagram of counter mode encryption. Diagram from [17, Diana Maimut].

MPDU using CCMP can be seen in Fig. 2.4.

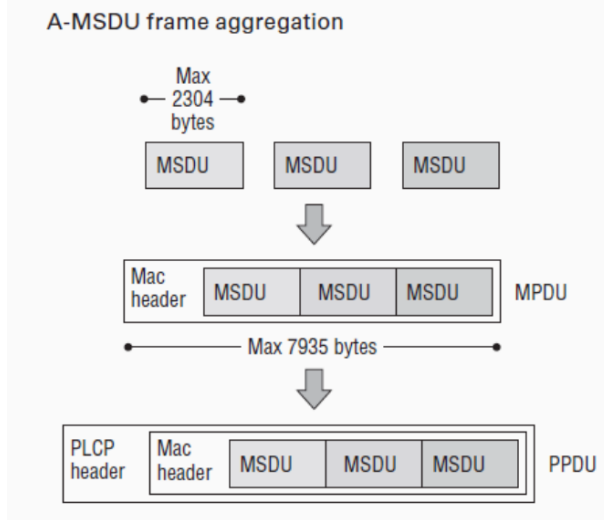
The process is finalized by appending the CCMP header in front of the encrypted MPDU and the MIC before combining this with the original MPDU header. The CCMP frame is then sent to the MAC layer, where the MAC frame will be constructed as explained in Section 2.2.2. As a result, the CCMP header is sent unencrypted to the data recipient. The final format of the header after encryption can be seen in Fig. 2.5.

### 2.2.5 Padding in WPA2 today

The use of CBC-MAC produces the MIC value, which is appended to the payload prior to encryption. The ciphertext  $C$  is the encryption of the original payload concatenated with the MIC value. However, the plaintext is encrypted using CTR as the mode of operation.

Given a plaintext message  $M$ , this message is divided into blocks  $p_1, p_2, \dots, p_l$  of 128-bits each where  $\sum_{i=1}^l p_i = |M|$ . However, depending on the message length  $|M|$ ,  $|p_l| \leq 128$ . For encryption, counter mode inputs four parameters: a nonce, a counter, a key, and a plaintext block  $p_i$ . The output is an encrypted block  $c_i$  of 128 bits. This process is performed by encrypting the nonce value and the counter value using the key  $k$  to produce a value  $t_i$ . This value is inputted to XOR with the plaintext  $p_i$ . The result of this computation is  $c_i$  [Fig. 2.6].

Since the final block  $p_l$  can be shorter than the cipher block size of 128 bits, it is unnecessary to use padding as the ciphertext has the same length as the plaintext. Even though the MIC value is appended to the payload prior to encryption, the length of the payload can easily be deduced by removing the



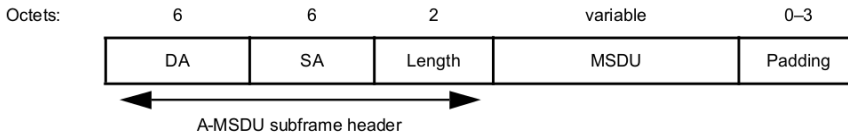
**Figure 2.7:** Visual display of A-MSDU (=MPDU). A single A-MPDU may consist of one or more MPDUs, while an A-MSDU may consist of one or more MSDUs. The figure illustrates the relationship between the two at the data link layer. Figure from <https://inet.omnetpp.org/docs/showcases/wireless/aggregation/doc/index.html>.

last 64 bits of the encrypted payload (assuming CCMP-128). Thus, the length of the payload can easily be determined.

### 2.2.6 Aggregation

The MAC layer processes each frame before sending it to the physical layer and over the medium. As a result, each frame contains a physical layer header, MAC header, and CCMP header. The overhead occurring from these headers is constant and incurs for each transmission. A solution to this is referred to as aggregation.

Aggregation was first introduced in the 802.11n [27] to reduce the overhead occurring from headers, and works by combining multiple packets to a single transmission unit that, under normal circumstances, would have been transmitted independently [5]. In Wi-Fi, modern protocols support two types of aggregation: aggregate MAC service data unit (A-MSDU) and aggregate MAC protocol data unit (A-MPDU). Fig. 2.7 shows the relation between regular frames, A-MSDU, and A-MPDU units. While the figure indicates that the maximum size of an A-MSDU frame is 7935 octets, the IEEE 802.11 standard [1] specifies a maximum size of 7951 octets for CCMP encryption. This thesis will consider



**Figure 2.8:** The IEEE 802.11 basic A-MSDU subframe format, illustrating the different fields in A-MSDU subframes. Figure from [1].

both cases, with and without using A-MSDU, as this is relevant to the data link layer.

Aggregate MAC service data unit (A-MSDU) refers to one or more MSDU frames combined into a single frame. When the MAC layer processes a packet, the packet is treated as the payload and will receive a single MAC header defined in Section 2.2.2. If encryption is used, the plaintext MPDU in Fig. 2.4 will encompass the whole A-MSDU, which consists of one or more MSDUs. Consequently, the A-MSDU will receive a single CCMP header, causing each MSDU within the A-MSDU to be encrypted with the same CCM nonce, AAD, and temporal key.

An A-MSDU frame consists of one or more subframes. Each subframe has an A-MSDU subframe header followed by the MSDU payload and 0 to 3 octets of padding to ensure the length is a multiple of 4 octets. The last subframe has no padding [1]. The MSDU can be viewed as a regular frame. The terminology (MSDU and MPDU) is introduced to be more specific regarding where the frame is located within the data link layer. The A-MSDU subframe header consists of the destination address, source address, and the length of the MSDU given in octets. The basic A-MSDU subframe structure is illustrated in Fig. 2.8. While there are different types of A-MSDU subframe formats, only the basic A-MSDU subframe format will be considered in this thesis.

There are issues with A-MSDU, though it reduces the overhead from the headers. It may introduce both delay and higher packet error rate due to the transmitter having to wait for collecting frames to aggregate, and the size of the aggregated frames increase [5]. However, using A-MSDU allows the transmission of several frames as a single unit on the physical layer and the retransmitting subframes at the A-MSDU level. Enabling aggregation improves efficiency, as fewer cyclic redundancy check calculations are needed, and it limits the number of MAC headers present within an A-MPDU.

## 2.3 Machine Learning

Machine learning is present in many different areas of our lives, such as healthcare, finance, marketing, and education. A reason for this is the development of new technology and algorithms, resulting in a rapid increase in online data and low-cost computation [18]. Machine learning has impacted empirical sciences by using algorithms to analyze data and solve problems. These could be finding an optimal approach to a problem, optimal parameters to a function, or solving a classification problem.

### 2.3.1 Classification

A classification problem can be viewed as a problem of predicting or identifying what class an object, data point, or observation belongs to based on previously observed data. *Classification* is a supervised machine learning process that understands the connection between input and output values by training on data before testing the *classifier*. It is an example of supervised learning since it tells the algorithm what to predict [14].

The process starts by dividing the data set into two sets: a training set used for the training phase and a test set used for the testing phase. During the training phase, the machine learning algorithm is trained through the use of a training set consisting of  $n$  tuples on the form  $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$ . Each  $X_i$  is called a vector of features, and each  $Y_i$  is a label [9]. The algorithm is trained using the tuples to calculate statistical properties and learn the relationship between the different vectors of features and labels. The machine learning algorithm uses the testing set of  $k$  tuples in the same format as the training set. The classifier is given  $k$  vectors of features,  $X_1, X_2, \dots, X_k$ . Given these vectors, the classifier must return labels  $Y'_1, Y'_2, \dots, Y'_k$ . The accuracy of the classifier is defined as

$$\text{Acc}_{\text{ML}}^{n,k,c}(\mathcal{A}) = \frac{\sum_{i=1}^k Y'_i \stackrel{?}{=} Y_i}{k}$$

where  $n$  is the size of the training set,  $k$  is the size of the testing set,  $c$  is the configuration within the chosen machine learning algorithm ML,  $\mathcal{A}$  is the entity utilizing the machine learning algorithm and  $Y'_i \stackrel{?}{=} Y_i$  is the prediction given the vector of features  $X_i$ . Note that  $(Y'_i \stackrel{?}{=} Y_i) = 1$  if and only if the classifier predicts correctly, 0 otherwise.

### 2.3.2 The naïve Bayes classifier

The naïve Bayes classifier is a probabilistic supervised machine learning algorithm. It is based on applying Bayes' theorem with the 'naïve' assumption that having information about one feature does not offer any extra information about any other feature, given an output variable. Given a label  $Y$  where the classifier observes  $X_1, X_2, \dots, X_n$ <sup>3</sup>,

$$P(Y | X_1, X_2, \dots, X_n) = \frac{P(Y)P(X_1, X_2, \dots, X_n | Y)}{P(X_1, X_2, \dots, X_n)} = \frac{P(Y) \prod_{i=1}^n P(X_i | Y)}{P(X_1, X_2, \dots, X_n)}$$

Thus, Bayes' theorem is simplified due to the naïve assumption. The classifier is considered relatively straightforward as the intuition behind the classifier is simple, and the parameters are easier to estimate compared to other algorithms. It is considered fast and handles high-dimensional data fairly well<sup>4</sup>. On the other hand, some issues must be addressed when using the naïve Bayes classifier, such as when a given label does not exist within the training set. Issues will be further discussed in [Chapter 3](#).

---

<sup>3</sup>[https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)

<sup>4</sup><https://www.ibm.com/topics/naive-bayes>





## CHAPTER 3

# Methodology

The aim of this chapter is to describe the methodology and offer an overview of the practical aspects of the experiment conducted in this thesis. It highlights the factors that need to be considered when analyzing the results.

The chapter begins by outlining the research design in [Section 3.1](#), followed by a detailed explanation of the setup in [Section 3.2](#). The analytical approach is then addressed in [Section 3.3](#) before the chapter concludes with a justification of the different aspects of the thesis in [Section 3.4](#).

### 3.1 Research design

The study follows an experimental research design, adopting a quantitative research method approach. The choice of a quantitative research approach is determined by the requirements of using machine learning algorithms. The rationale for experimenting is to simulate a real-world scenario closely, providing insight into the success rate of a real-world attacker performing website fingerprinting using the same techniques as previous research.

The experiment is split into three key stages: data collection, feature extraction and sanitation, and machine learning classification. Data collection involves using a single computer to generate traffic using Python scripts and a single computer capturing the traffic using a packet capture tool. The data is filtered to extract specific features using Python scripts, then pre-processed and classified using a data analysis library in Python. Details regarding software and tools used in the thesis are elaborated in [Section 3.2.3](#). Finally, the classifiers are trained, and the performance of the attack is evaluated in line with the research objectives.

### 3.2 Setup

This section will cover this thesis's logical and physical components, software, and tools. The setup is covered before the method's explanation to give the reader a better understanding of the analytical approach.

#### 3.2.1 Logical components

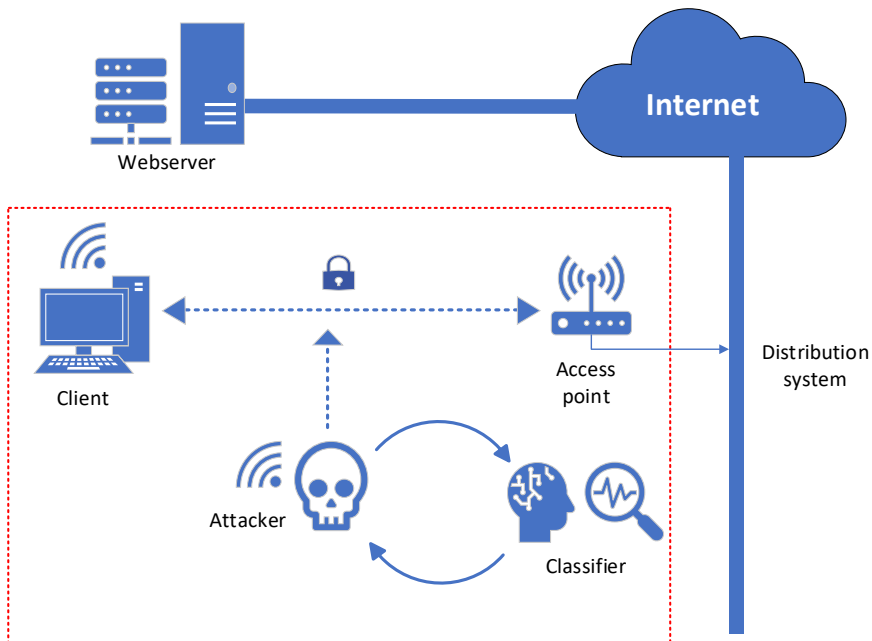
The different logical components in this thesis are a web server hosting a website, a client, a wireless access point, an attacker, and the classifier. While the distribution system is a main component of 802.11, this thesis does not consider it. [Figure 3.1](#) illustrates the different logical components and their connection within this experiment. The logical components are the following.

First, a virtual server provided by 'is\*hosting'<sup>1</sup>, a hosting company, is used to host the website. The virtual server is located in Norway and operates on Ubuntu 22 x64 as its operating system. The website is served using the open-source web server nginx<sup>2</sup>, using a PHP script that dynamically displays various PDF files stored on the server. Nginx was chosen as the web server due to its popularity and ability to serve static content quickly. The PHP script generates a webpage

---















<sup>1</sup><https://ishosting.com/en/about>

<sup>2</sup><https://nginx.org/en/>



**Figure 3.1:** The logical components of the experiment. The red box demonstrates the setting this experiment considers.

### Files

File	Upload Date	File Size (Bytes)
 <a href="#">2010_006</a>	2024-03-25 10:06:50	594571
 <a href="#">2010_019</a>	2024-03-25 10:06:07	479521
 <a href="#">2010_026</a>	2024-03-25 10:06:58	728020
 <a href="#">2010_031</a>	2024-03-25 10:06:52	218744
 <a href="#">2010_049</a>	2024-03-25 10:06:53	314501
 <a href="#">2010_053</a>	2024-03-25 10:06:58	179796
 <a href="#">2010_055</a>	2024-03-25 10:06:08	589603
 <a href="#">2010_056</a>	2024-03-25 10:06:15	554265
 <a href="#">2010_061</a>	2024-03-25 10:06:17	121525
 <a href="#">2010_065</a>	2024-03-25 10:06:08	121779
 <a href="#">2010_072</a>	2024-03-25 10:06:19	231646
 <a href="#">2010_099</a>	2024-03-25 10:06:34	358181
 <a href="#">2010_102</a>	2024-03-25 10:07:07	252723
 <a href="#">2010_117</a>	2024-03-25 10:06:28	290400

**Figure 3.2:** Screenshot of the website with domain name [uioencryptedtraffic.no](https://uioencryptedtraffic.no)

with a header and a table of rows and columns. Each row represents a single PDF file, where each row has three columns: the name of the PDF file with a small PDF icon displayed next to the name, the size of the PDF file measured in bytes, and the upload date of the PDF file. An illustration of the website's front-end is shown in Fig. 3.2. The website can be accessed at <https://uioencryptedtraffic.no>.

Second, the client is a logical component that can retrieve data from the web server by performing a GET request. It generates traffic within the network according to a specific format. This is further discussed in Section 3.3.1. During the experiment, the client is a single desktop computer associated with a wireless access point. The client is the only entity generating traffic within the network.

Third, the wireless access point is the logical component connecting the client to the web server. All data transmitted from the client is forwarded to the access point, where the payload is encrypted. The wireless access point then forwards the data to the distribution system. For the scope of this thesis, a single wireless access point is considered where the data flow between the access point and the web server is disregarded.

Fourth, the attacker is the logical component capable of monitoring all network traffic received on a wireless channel, including the observation of raw 802.11 frames, and is achieved by configuring the wireless network interface

in monitor mode. The thesis considers a single attacker capable of observing the MAC header, CCMP header, and the encrypted payload and calculating metadata from the various packets transmitted over the medium. Additionally, the attacker neither controls the wireless access point nor is associated with the wireless access point. The attacker cannot access the decrypted payload.

Lastly, the classifier is the final logical component. The attacker trains the classifier by providing data that can be split into two groups: the features and the answer. The features consist of metadata from the encrypted network traffic. The classifier predicts the answer based on the provided features. The classifier's prediction determines the type of PDF file the client fetches or the web page the client is accessing.

### 3.2.2 Physical components

This thesis utilizes various physical components: a desktop computer, a laptop computer, and an access point. The virtual server is not considered a physical component due to a lack of accessible information regarding the company's different components.

The desktop computer runs on Microsoft Windows 11 Home and has a TUF GAMING B650-PLUS Wi-Fi motherboard with an x86\_64 architecture. It uses a MediaTek Wi-Fi 6 MT7921 Wireless LAN card network adapter and an ASUS 2T2R Dual Band Wi-Fi moving antenna to connect to the wireless access point.

The laptop computer is a Huawei Notebook with a WRT-WX9-PCB M1010 motherboard with an x86\_64 architecture. Its network card is an Intel Cannon Point-LP CNVi [Wireless-AC], operating using the wireless driver 'iwlwifi' with driver version 5.4.0-174-generic.

The wireless access point is a PC Engines APU2 x86\_64 single-board computer. It operates on an OpenWrt operating system version 23.05.0. OpenWrt is an open-source project based on Linux that was developed for embedded systems and primarily used on devices tasked with routing network traffic. The access point has an AMD GX-412TC SOC chip with a 1 GHz quad Jaguar core with 64-bit architecture. It has a Qualcomm Atheros QCA9880 802.11ac/b/g/n wireless chipset and a wle600vx wireless Wi-Fi kit. The wireless access point is physically connected to a functioning router using an ethernet cable. This router is linked to the broader network through fiber optic cables and an Internet service provider. Only the PC Engines APU2 is considered when referring to the wireless access point.

### 3.2.3 Software and tools

**Aircrack-ng** : Aircrack-ng<sup>1</sup> is a suite of tools focusing on different areas of Wi-Fi security in order to assess Wi-Fi network security. These areas are monitoring, attacking, testing, and cracking. All tools are command-line tools working primarily on Linux, although they are compatible with other operating systems. In this thesis, the airmon-ng and airodump-ng command line tools are utilized. Airmon-ng<sup>2</sup> is a script allowing the user to move between monitor mode and managed mode. Airmon-ng intends primarily to allow a user to enable monitor mode on a wireless interface. Airodump-ng<sup>3</sup> is a tool for capturing raw 802.11 frames. When executed, several files containing details of the captured data are generated. Numerous options can be specified when utilizing the airodump-ng tool.

- - **channel** : This option allows the attacker to specify what channel to listen to when capturing the raw 802.11 frames.

- - **bssid** : This option allows the attacker to filter access points using the BSSID value.

- - **write** : This option is used to determine the prefix name of the output file after capturing data.

**Scikit-learn** : Scikit-learn [34] is a free open-source library for machine learning in Python. It is a simple and efficient tool built on SciPy, Matplotlib, and NumPy libraries. The Scikit-learn library contains various algorithms for implementing machine learning algorithms for supervised and unsupervised learning. Among these is the classifier Gaussian Naïve-Bayes.

**Python** : Python<sup>4</sup> is a high-level open-source language that comes with several notable features, including an extensive standard library and is easily extended by integrating new modules coded in a compiled language. An example of a built-in module is the socket module, which enables socket programming in Python. Utilizing libraries such as Scapy, NumPy, Matplotlib, and Pandas allows files to be processed for machine learning purposes and data for visual content to be plotted.

**Jupyter Notebook** : Jupyter Notebook<sup>5</sup> is a part of the project Jupyter, a free open standards software for interactive computing through a web-based

---

<sup>1</sup><https://www.aircrack-ng.org/>

<sup>2</sup><https://www.aircrack-ng.org/doku.php?id=airmon-ng>

<sup>3</sup><https://www.aircrack-ng.org/doku.php?id=airodump-ng>

<sup>4</sup><https://wiki.python.org/moin/BeginnersGuide/Overview>

<sup>5</sup><https://jupyter-notebook.readthedocs.io/en/stable/>

interface. It supports many programming languages, where users typically use Jupyter Notebook using computational notebooks. Computational notebooks are frequently used for data science, scientific computing, and machine learning. These notebooks provide a fast, interactive environment to run live code and visualize data through creating and running cells within a single notebook. Using notebooks is favorable for analyzing and sanitizing data for data preprocessing before utilizing machine learning algorithms.

### 3.3 Method

This section will address the analytical approach in the work and justify the chosen approach. The method is divided into two parts: the data collection process and the data analysis procedure. It is important to mention that all scripts used in this thesis are written in Python and executed directly in the terminal of the Linux computer or the terminal window of the Windows computer in use.

#### 3.3.1 Collection of data

Data collection occurs in three stages: preparing web pages to be accessed, the client generating network traffic, and the attacker performing the website fingerprinting attack. Both network traffic generation and the attack take place within the author's apartment. The process of the attacker performing the website fingerprinting attack remains consistent across scenarios.

In all scenarios, the client fetches data by executing a script that makes a GET request to a given web page. Given a scenario, an experiment is defined given a classifier and three values:  $q$ ,  $k$ , and  $t$ . The number  $q$  represents the number of traces performed for a given data. A *trace* is defined as the encrypted traffic of a single GET request and response to a web page. The number  $k$  decides the size of the privacy set<sup>3</sup> used for a *trial*. A single experiment consists of  $t$  number of trials. For all scenarios, the value of  $q$  is set to 20, and the value  $t$  is chosen such that  $t = \lceil \frac{4000}{k \cdot 20} \rceil$ . To prevent any caching during the session, a header  $h$  is provided as a parameter to the GET request, ensuring that the server side or any proxy does not cache each request. The result is not cached locally. Additionally, `time.sleep()` is used between GET requests, introducing a time pause of one second between fetching of data. Justification for the choice of parameters and the use of `time.sleep()` is further explained in [Section 3.4](#). Finally, the different experiments were conducted between the 26th of September

---

<sup>3</sup>Privacy set is also called anonymity-set. See [38].





**Figure 3.3:** Illustration of the simple PDF fingerprinting scenario.

and the 13th of October 2024.

The web page preparation and network traffic generation vary depending on the scenario in focus. The scenarios consist of the following.

**Simple PDF fingerprinting:** The preparation begins with creating a web page hosted on a web server storing PDF files. The PDF files stored on the web server originate from the Cryptology ePrint Archive website, accessible through the URL: <https://eprint.iacr.org>. This website provides easy access to recent research within the field of cryptology, published by researchers without undergoing any refereeing process beforehand. The PDF files displayed on the website are pre-fetched and uploaded before the experiment.

There are two reasons for choosing the ePrint archive as our data source before the experiment. First, the ePrint website is easily iterable for fetching different PDF files. Each specific PDF file can be accessed by modifying the ULR accordingly:

$$\{\text{original\_URL}\}/\{\text{published\_year}\}/\{\text{publish\_number}\}.\text{pdf}$$

Second, the website contains many PDF files of different lengths, so it stands out as a suitable candidate for our experiment. This arises from the experiment's closed setting, involving limited variables and a fixed size of the data used by the machine learning algorithms. PDF files follow a specific format, which suits the purpose of the experiment, and it allows the author to control the static data on the website.

The script for fetching PDF files from the Cryptology ePrint Archive website was executed on the 25th of Mars from 09:00 to 10:00 at the University of Oslo.

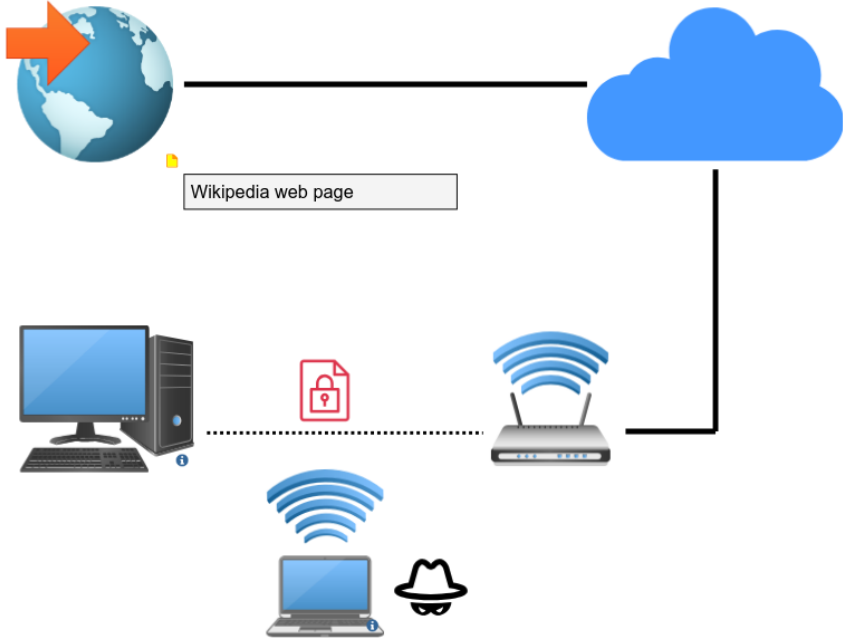
It iterates through each year from 2010 to 2018, choosing  $n$  number of files uniformly at random. Each selected file is added to a set to prevent duplicates, and all files are appended to a folder locally. The folder is uploaded to the web page after the script has terminated. The web page contains a total of 900 different PDF files, 100 PDF files for each year. The fetched files are uploaded to the web page remotely using SSH.

Before generating network traffic, a basic SQLite database is created. Each entry in this database includes a unique identifier, the name of a PDF file stored on the web page, and the PDF file size measured in bytes. A script is executed to gather and store all information about the PDF files on the web page in this database. This database is later used to calculate the trivial success probability after completing the experiment.

This thesis adopts the same message distribution as Gellert et al. [12], the uniform distribution. Let  $|\text{pad}(m, \ell)| = \left\lceil \frac{|m|}{\ell} \right\rceil \cdot \ell$ , and  $\mathcal{M} = \{m_1, m_2, \dots, m_{900}\}$  be the set of all 900 different PDF files stores on the web server. In this scenario, the adversary aims to determine which PDF file the client is accessing. Therefore, the adversary's output in the theoretical model is an index  $i \in \{1, 2, \dots, 900\}$  such that  $\mathcal{P} : \mathcal{M} \rightarrow [1, 900]$  for  $m_i \mapsto i$ . The trivial advantage [Definition 2.4] is calculated by considering different values of  $\ell$ , investigating how the trivial advantage decreases as the size of  $\ell$  increases. This thesis considers, among others, two special values of  $\ell$ : the value at which the number of uniquely identifiable PDF files is 0 and the value at which all PDF files have the same length.

There are two reasons for focusing on the theoretical setting and calculating the trivial success probability. First, the thesis assumes the use of the CCMP-128 encryption protocol. The payload length can be deduced by observing the encrypted 802.11 frames, given how the encryption protocol operates [see Section 2.2.5]. Since CCMP-128 does not include padding by default, there exists a near one-to-one correspondence between the lengths of the PDF files and the corresponding encrypted 802.11 frames in practical scenarios. As a result, focusing solely on the lengths of the PDF files allows trivial advantage (using the trivial success probability) to highlight the leakage of message length through encrypted 802.11 frames. Second, this approach enables a comparison between the success of an adversary performing a website fingerprinting attack and the theoretical trivial advantage, which could provide a better understanding of how the theoretical measure of advantage relates to the actual performance of a website fingerprinting attack.

Network traffic generation begins with the client iterating through the web page to compile a list of all available PDF files. Subsequently, the client fetches



**Figure 3.4:** Illustration of the Wikipedia fingerprinting scenario.

k PDF files uniformly at random, with each PDF file fetched 20 times. An illustration of the scenario can be viewed in Fig. 3.3.

**Wikipedia fingerprinting:** The preparation begins with running a script that performs k GET requests to <https://en.wikipedia.org/wiki/Special:Random>, a Wikipedia URL used to access a random article in the main namespace<sup>4</sup>. The URLs fetched from the GET requests are stored in a list and checked during the fetching process to prevent duplicates. The result is the privacy set used during the trial.

There are two reasons for focusing on Wikipedia articles. First, Wikipedia is a public, commonly used website, where the English Wikipedia<sup>5</sup> has approximately 6.9 million articles with more than 1.5 billion device visits monthly. Utilizing Wikipedia articles will, therefore, result in different results concerning the sizes of web pages fetched for our set within a trial. However,

---

<sup>4</sup>Techniqually the URL is pseudorandom, see <https://en.wikipedia.org/wiki/Wikipedia:FAQ/Technical#random>

<sup>5</sup><https://en.wikipedia.org/wiki/Wikipedia>

it is also convenient as a random article can be easily accessed using a single URL. Second, while the sizes of Wikipedia pages may differ, the format is more or less the same for each article. Many components found on each webpage are independent of the specific article accessed.

The difference is partially because of the number of characters displayed on the page and possible different file formats such as JPG, SVG, or tables. As a result, while many components are the same, the length of the webpage measured in bytes is different, reflecting the experiment's intention. Therefore, when changing the context from a closed limited setting (simple PDF fingerprinting scenario) to a more real-world context, Wikipedia seems a suitable candidate.

The client generates network traffic by iterating through the list from the first element to the last element within the list of  $k$  URLs. The client then performs 20 GET requests per URL chosen. Fig. 3.4 illustrates the scenario.

**Domain fingerprinting:** The preparation begins with obtaining a ranking of popular web pages frequently accessed worldwide, including subdomains. This thesis utilizes the Tranco list<sup>6</sup> generated on 11 September 2024, a research-oriented top site ranking. The Tranco list aggregates data from various rankings, including those provided by Google, Cloudflare, and Cisco. Not all GET requests for entries in the Tranco list returned a status code of 200. Consequently, the experiment uses the top 8000 entries from the Tranco list that responded with a status code 200. The list was pre-fetched before the experiment commenced.

There are two reasons for using the Tranco list in this thesis. First, the researchers behind the Tranco list [25] demonstrated that other commonly used rankings, such as the Majestic Million list and the Radar list, have notable shortcomings. The Tranco list was created to address issues such as vulnerability to manipulation and inconsistencies in determining the most popular domains. Second, the Tranco list offers improved stability over time where the methodology is fully detailed<sup>7</sup>, the properties of the source lists are explained in detail, and the researchers have published the source code for generating the list. These properties and the transparency enhance the credibility of the ranking.

The experiment starts with the client uniformly randomly picking  $k$  unique domains from the Tranco list. Each domain is stored in a local list. The client then iterates through each domain in the list and performs 20 GET requests per domain. Fig. 3.5 illustrates the domain fingerprinting scenario.

---

<sup>6</sup><https://tranco-list.eu/list/83PGV>

<sup>7</sup><https://tranco-list.eu/methodology>



**Figure 3.5:** Illustration of the domain fingerprinting scenario.

The final data collection stage involves the attacker performing the fingerprinting attack, independent of the scenario in focus. This process begins with the attacker using the Aircrack-ng suite to check and terminate interfering processes and switch the network card from managed mode to monitor mode. The attacker proceeds to execute a script establishing a connection between the client and the attacker using socket programming in Python. The script continuously listens to a signal from the client's computer. Based on the signal received, one of four actions is triggered:

**'START\_CAPTURE'** : The attacker starts capturing network traffic. This signal is before the client performs a GET request to a website.

**'STOP\_CAPTURE'** : The attacker stops capturing network traffic. This signal is sent after the GET response from a website.

**'Answer:'** : The attacker receives the name of the accessed PDF file or web page. This information is essential for training the classifier. Depending on the scenario, this signal is sent after fetching a PDF file or web page.

**'STOP\_SCRIPT'** : The attacker exists the script. The signal indicates that the experiment is finished.

Capturing encrypted 802.11 frames is done using airodump-ng within the Aircrack-ng suite, using the options outlined in [Section 3.2.3](#). After capturing the encrypted traffic, the attacker obtains PCAP files with information such as the MAC header, CCMP header, and the encrypted payload, which can be used to calculate metadata from the different packets transmitted over the medium. The adversary can distinguish multiple device connections using MAC addresses, allowing the adversary to map traffic to connections. However, this is irrelevant since this thesis considers a single client within the network.

Three main reasons exist for using the specific options outlined in [Section 3.2.3](#) when capturing encrypted 802.11 frames. First, the attacker must not be associated with the access point during the experiment. However, this also means the attacker risks capturing a large amount of data sent between different access points and clients. These options enable the attacker to filter out unwanted traffic from other networks, reducing noise and increasing the likelihood that the captured 802.11 frames correspond directly to the GET requests and responses. Second, the captured data may be saved to a file with a simpler name for later use. Finally, these options help preserve the privacy of other users, allowing the experiment to be conducted ethically by focusing solely on the private network and avoiding external traffic.

### 3.3.2 Filtering and selection of data

The attacker possesses PCAP files extracted from the captured encrypted network traffic and starts filtering by iterating through the files. For each PCAP file, the attacker filters out all packet data frames with a specific packet type, packet subtype, CCMP header, and 802.11 frame header. The filtering process ensures the packets considered are encrypted 802.11 data frames that transmit data between the client and the access point. Given a packet, a 6-tuple of information is appended to a set before processing a data frame, used to check for retransmissions.

The selection of features originates from previous research [9] considering different features for training classifiers. The thesis explores two cases: focusing solely on packet lengths with directionality or focusing on total trace time, upstream/downstream total bytes, number of traffic bursts, and bytes in traffic bursts. In both cases, the script iterates through each PCAP file present, calculates features, and writes the features to a CSV file. The resulting CSV file contains

### 3.3. METHOD

0_up	1_up	2_up	3_up	4_up	5_up	6_up	7_up	8_up	9_up	...	1542_down	1543_down	1544_down	1545_down	1546_down	1547_down	1548_down	1549_down	1550_down	Answer
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	465	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	446	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	473	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	471	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	465	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	473	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	470	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	489	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	433	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	463	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	483	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	464	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	444	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	469	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	476	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	464	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	454	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	476	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	456	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	469	0	0	2015.856
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	155	0	0	2013.084
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	156	0	0	2013.084

**Figure 3.6:** Illustration of the CSV file format considering only packet lengths with directionality as feature.

features representing PCAP files. The choice of features in focus is further discussed in [Section 3.4.5](#).

In the first case, the `pcap_to_ml_csv_length()` function is employed. The *packet lengths* are calculated by initiating columns inside a CSV file, each representing a unique packet length, taking the directionality into account. Each time a packet is observed with length  $L$  going in direction  $D$ , the value in column  $L\_D$  is incremented by one. Direction 'up' denotes packets sent from the client to the access point, whereas 'down' means packets sent from the access point to the client. The last column in the CSV file contains the answer, a necessary field for the machine learning algorithm. A visual display of the CSV file format using the `pcap_to_ml_csv_length()` function is provided in [Fig. 3.6](#).

In the second case, the `pcap_to_ml_csv_full()` function is utilized, focusing on total trace time, upstream/downstream total bytes, number of traffic bursts, and bytes in traffic bursts as features. *Total trace time* is calculated by the difference between the timestamps of the first packet observed and the last observed downstream packet in the same PCAP file. *Traffic bursts* are defined as a sequence of packets having the same directionality. For instance, if the attacker observes the following traffic:

$$(\text{up}, 200), (\text{down}, 800), (\text{down}, 1300), (\text{down}, 1450), (\text{up}, 200)$$

	Nr_packets_up	Nr_packets_down	Bytes_up	Bytes_down	Bursts	Bytes_in_bursts	Trace_time	Answer
0	73	129	7362	189152	107	79260	2.176631	2013_609
1	72	129	7243	190438	109	82138	0.388150	2013_609
2	72	121	7231	178054	91	66766	0.382455	2013_609
3	72	130	7255	193415	101	74453	0.391159	2013_609
4	71	128	7143	190344	97	71987	0.397366	2013_609
5	75	120	7304	176506	85	63451	1.328182	2013_609
6	73	123	7657	181150	105	76447	1.415735	2013_609
7	71	125	7143	184246	111	81192	0.388663	2013_609
8	72	127	7231	185888	107	79809	0.909367	2013_609
9	67	119	6791	176412	87	64948	0.382458	2013_609
10	69	129	6975	191892	109	82940	1.115194	2013_609
11	71	131	7017	193534	113	83969	0.391737	2013_609
12	73	120	7331	179202	109	82907	0.879158	2013_609
13	73	130	7350	189246	121	86651	2.961591	2013_609
14	72	128	7243	191605	105	79678	0.400886	2013_609
15	69	121	6979	179508	105	79060	0.397814	2013_609
16	72	129	7231	190438	97	71987	0.385589	2013_609
17	71	122	7029	181056	105	78891	0.390646	2013_609
18	72	128	7243	190344	111	83787	0.378361	2013_609
19	71	119	7155	176225	99	72517	0.385072	2013_609
20	162	213	16149	319927	198	156972	0.622592	2011_686
21	162	208	16227	312187	184	142845	0.535091	2011_686

**Figure 3.7:** Illustration of the CSV file format considering total trace time, upstream/downstream total bytes and number of frames, number of traffic bursts and bytes in traffic bursts as features.



### 3.3. METHOD

```
[3]: import pandas as pd
import os
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

[4]: current_directory = os.getcwd()
folder_path = os.path.join(current_directory, 'wiki_k2_full_plain')
files_in_folder = [filename for filename in os.listdir(folder_path)]

[12]: accuracy_score_list = []
for file in files_in_folder:
    path_to_file = os.path.join(folder_path, file)
    df_k2f = pd.read_csv(path_to_file)

    X = df_k2f[df_k2f.columns[:-1]].values
    y = df_k2f[df_k2f.columns[-1]].values

    # Running the algorithm multiple times for each file
    avg_accuracy_per_file = 0
    num_iterations = 100 # Adjust as needed

    for _ in range(num_iterations):
        # Splitting the set into training and test set following an 80% - 20% split (train size=0.8)
        # Shuffle the dataset (shuffle=True) and drawing equal number of samples for each label (stratify=y)
        X_train_2l, X_test_2l, y_train_2l, y_test_2l = train_test_split(X, y, train_size=0.8, shuffle=True, stratify=y)

        # Gaussian Naive Bayes
        nb_model = GaussianNB()
        nb_model_2l = nb_model.fit(X_train_2l, y_train_2l)
        y_pred_2l = nb_model_2l.predict(X_test_2l)

        avg_accuracy_per_file += accuracy_score(y_test_2l, y_pred_2l, normalize=True)

    avg_accuracy_per_file /= num_iterations
    accuracy_score_list.append(avg_accuracy_per_file)

# Calculating the overall average accuracy across all files
overall_avg_accuracy = sum(accuracy_score_list) / len(accuracy_score_list)

print("Average accuracy for each file:")
print(accuracy_score_list)
print(".....")
print("Number of files processed:", len(accuracy_score_list))
print(".....")
print("Overall average accuracy:", overall_avg_accuracy)
```

**Figure 3.8:** Illustration of training the machine learning model inside Jupyter Notebook for the case of  $k = 2$ , considering total trace time, upstream/downstream total bytes, number of traffic bursts and bytes in traffic bursts as features.

then the number of traffic bursts is three. For each packet observed, the different values are updated until all packets are processed. Like the `pcap_to_ml_csv_length()` function, the last column in the CSV file contains the answer. A visual display of the CSV file format using the `pcap_to_ml_csv_full()` function is provided in [Fig. 3.7](#).

#### 3.3.3 Training the classifier

The attacker transitions to Jupyter Notebook [Fig. 3.8](#) after obtaining the CSV files for a specific  $k$  and features considered. Here, the necessary libraries are imported in order to train the classifier. The process begins by storing the CSV files with a list, iterating through each CSV file, and reading its content.

For each CSV file, two variables,  $X$  and  $y$ , are defined, representing the features and labels. The dataset is split into a training and test set, following an 80%-20% split. The data is shuffled using a pseudorandom generator. Stratified

sampling ensures all labels are represented during training and works by dividing the dataset into its different labels and drawing an equal number of samples from each label. Using stratified sampling eliminates sampling bias and prevents potential issues with zero division when applying Bayes' theorem within the classifier. As a result of the split, four variables are generated: the features and labels for both the training and test sets.

The classifier used is a Gaussian naïve Bayes model, which assumes that the features follow a Gaussian distribution given a particular label. Gaussian naïve Bayes is common due to its simplicity and works by estimating the mean and standard deviation of the training set. Based on the training set, the classifier predicts the labels, considering the estimation of the mean and standard deviation.

The process of splitting the dataset following an 80%-20% split and training the classifier is performed 100 times to calculate an average accuracy score; this is the classifier's accuracy. Repeating the process 100 times ensures a single split does not result in skewed results, achieving a more accurate measurement.

## 3.4 Limitations

This section will cover the limitations of data collection and analysis. The limitations are selection bias, limited scope, measurement errors, sample size, and time constraints. Lastly, the justification and mitigation of the different limitations will be addressed.

### 3.4.1 Selection bias

Choosing PDF files and web pages uniformly at random is not realistic. For the first scenario, the PDF files originate from the ePrint Archive website, where the papers do not undergo any refereeing process beforehand. Good research articles will receive more citations and attention than others. As a result, this will increase the likelihood that a client will access specific articles. Furthermore, in both years included, the PDF files were fetched between 2010 and 2018. Fetching a subset of available PDF files could also introduce size bias.

### 3.4.2 Limited scope

This thesis considers only a single classifier, limiting the scope. It could be the case that other classifiers will perform better than the classifier considered. The test set and training set used by the classifier are also fairly limited due to the experiment being performed a limited number of times for different values of

### 3.4. LIMITATIONS

---

k. The experiment's results will only indicate the accuracy of the naïve Bayes classifier. This accuracy would be more reliable if one performed the experiment a greater number of times and included bigger and more values of  $k$ . In addition, no parameter tuning is performed to optimize the classifier's performance. The adjustment of hyperparameters of a machine learning model would also increase the accuracy of the classifier considered. An example is the assumption that the features follow a Gaussian distribution when training the classifier. While this distribution assumption may work well in many cases, it is not necessarily suitable in this context.

#### 3.4.3 Measurement errors

The attacker is not guaranteed to observe traffic related to only the specific PDF file or web page fetched by the client. If the client sends other 802.11 data frames containing a CCMP header, then this will be interpreted as a packet related to the attacker's fetching of the specific PDF file or web page.

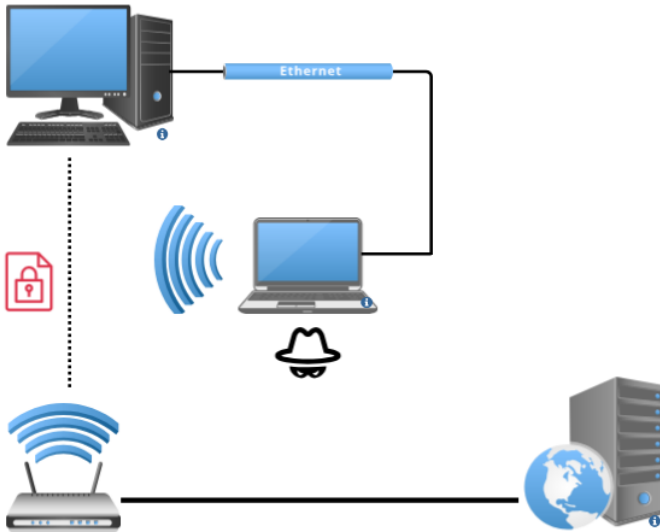
Furthermore, there will be a slight delay between a client sending a signal to the attacker and the attacker receiving and acting upon the signal. While the thesis has implemented a time delay to ensure the attacker observes whole GET requests, there is still a trade off between time delay and the amount of data the author may collect. Therefore, the choice of the time delay is relatively small, which could introduce measurement errors where the attacker does not observe some parts of the GET requests if there are delays with the signal sent between the two computers.

#### 3.4.4 Sample size and time constraints

The sample size considered is limited due to time constraints. Data for different values of  $k$  was collected over several days, unlike previous research that collected data over several months. As a consequence, the limited amount of data collected results in a limited dataset fed to the classifier. This affects the accuracy of the classifier and, thereby, the results of the experiment.

#### 3.4.5 Justification and mitigation

This thesis achieves two things, starting with a limited setting, assuming PDF files and web pages are chosen uniformly at random. Firstly, it simplifies the scenario compared to a real-world scenario, which is more complex. In a real-world context, the distribution depends on factors such as the number of users, geographical location, and time of day. In addition, a user can access multiple web pages in parallel, making it difficult for the attacker to distinguish between the



**Figure 3.9:** Illustration of the physical setup.

different web pages accessed. By simplifying the context, the evaluation of the performance of the fingerprinting attack can be used as a baseline for comparison or as an approximation of the success rate of the website fingerprinting attack in future work, allowing future research to introduce greater complexity. This is the intention of introducing the simple PDF fingerprinting scenario and the Wikipedia fingerprinting scenario before complicating the scenario to a more real-world context (domain fingerprinting scenario). Secondly, uniformly choosing PDF files and web pages removes the possibility of introducing bias into the dataset. In addition to this, it eliminates the need for any a priori information about the distribution.

The choice of fetching 900 different PDF files from the ePrint archive could introduce size bias into the dataset. This is checked by fetching all files from the website and storing them in a local database on the laptop computer before the experiment, ensuring variability in file size. While the distribution of PDF sizes may have been affected, it was verified that the selection process ensured a diverse range of file sizes across the dataset.

### 3.4. LIMITATIONS

---

A single classifier is considered in this thesis despite the existence of multiple classifiers an attacker could use to perform a website fingerprinting attack. The naïve Bayes classifier was chosen due to its simplicity, requiring minimal prior knowledge of machine learning algorithms. The attacker only needs to understand basic machine learning concepts and Bayes' theorem. An example is the use of the Gaussian distribution, which requires little knowledge by the attacker. Many datasets contain features that approximately follow a normal distribution. Thus, the use of a Gaussian distribution is logically a priori. The success rate of the naïve Bayes classifier serves as a lower bound for the attack. Configuring the parameters of the chosen classifier and having in-depth knowledge of different classifiers will only increase the attack's success rate.

Two feature sets are considered: packet lengths with directionality and total trace time, upstream/downstream total bytes, number of traffic bursts, and bytes in traffic bursts. The first considers individual frame lengths, taking the directionality into account. The other considers features looking at the traffic as a whole. Considering two different feature sets provides insight regarding what information is preferable for a classifier when distinguishing data and what information is more critical if leakage through side channels occurs. A time constraint impacted the decision not to include more feature sets.

It is uncertain whether every packet observed corresponds to a GET request. Although the attacker only processes data frames with a specific format, some packets might not be part of the response. Nonetheless, the thesis simulates a setting intended to reflect real-world conditions. The challenge of distinguishing data purely by observing encrypted 802.11 frames is also an issue for real-world attackers. During the experiment, the desktop computer fetching the data is configured to turn off any running applications that may run in the background. Nevertheless, processes starting during the experiment can potentially affect the results. However, these occurrences are expected to have a minor effect on the results.

Splitting PCAP files is essential for training the classifier, where the issue of splitting is well known, as the attacker only observes streams of encrypted data sent over the medium. It is not intuitive when the attacker observes the end of a GET response and the start of a new GET request. This thesis solves the issue of splitting the PCAP files by physically connecting the client's computer and the attacker's computer. A script interprets signals transmitted over the medium, enabling the attacker to know when to start and stop capturing data. A timeout of one second is applied between each GET request. This timeout value was determined based on empirical data from several test cases before the experiment. An illustration of the physical setup and the approach taken to split

the PCAP files can be viewed in Fig. 3.9. The used scripts and the produced CSV files can be accessed through <https://github.com/Embrikht/Thesis>.

Unlike previous studies [15, 26], this experiment was conducted over a relatively limited number of days. Although earlier research benefited from a more extended data collection period, this thesis still provides valuable insights into an attacker's potential capabilities. A total of  $\lceil \frac{4.000}{k \cdot 20} \rceil (k \cdot 20)$  data samples were gathered for each scenario given a  $k$ , which, in the author's view, indicates the attack's accuracy. While additional data would impact the results, it is expected that the change in accuracy would not be substantial. This thesis aims to demonstrate information leakage through encrypted 802.11 frames and how an attacker could exploit this information. The goal is not to determine an exact attack accuracy but to raise awareness of the attack's feasibility. The calculated accuracy should be interpreted as a baseline, not a precise measure.

### 3.4. LIMITATIONS

---

## CHAPTER 4

# Results

This chapter will present the results of the experiments described in [Chapter 3](#).

First, the results of the simple PDF fingerprinting scenario will be presented in [Section 4.1](#). This will include the calculation of the trivial success probability. Next, the Wikipedia fingerprinting scenario results will be provided in [Section 4.2](#). Finally, the results of the domain fingerprinting scenario will be revealed in [Section 4.3](#). Each scenario considers with and without A-MSDU.



## 4.1 Simple PDF fingerprinting

The first scenario considered a closed setting where the retrieved PDF files from a basic website. The attacker was aware that the user was accessing the web page and wanted to determine which PDF files the user was accessing. This scenario works as a baseline scenario, before introducing a more complex setting.

### 4.1.1 Trivial success probability

The use of the trivial success probability allows for an evaluation of different choices of padding, observing the correlation between padding and the reduction in the trivial advantage of an attacker. This provides information of message length leakage through side channels, illustrating the feasibility of an attacker if no mitigation technique is used. Accordingly, focusing on a theoretical setting can provide insights into what to expect from the 802.11 standard.

Recall that  $\mathcal{M} = \{m_1, m_2, \dots, m_{900}\}$  is the set of all 900 different PDF files stores on the web server, where the adversary's output is an index  $i \in \{1, 2, \dots, 900\}$  and  $\mathcal{P} : \mathcal{M} \rightarrow [1, 900]$  for  $m_i \mapsto i$ . The trivial success probability [Definition 2.2] is defined as

$$\mathbf{TrivSucc}(\Sigma, \mathcal{M}, \mathcal{P}) = \max_S \Pr[\mathcal{S}(\mathcal{M}, \mathcal{P}, (|c_i^*|)_{i \in [t]}) = \mathcal{P}(m^*)]$$

where this thesis considers  $\mathcal{M}$  as the uniform distribution. Now define  $\mathcal{M}_\lambda = \{m \in \mathcal{M} : |\text{Enc}(K, m)| = \lambda\}$  be the set of messages that under  $\Sigma$  encrypt to a ciphertext of length  $\lambda$ , and  $|S|$  be the number of different ciphertext lengths. This results in the following <sup>1</sup>.

$$\mathbf{TrivSucc}(\Sigma, \mathcal{M}, \mathcal{P}) = \sum_{\lambda \in S} \frac{|\mathcal{M}_\lambda|}{|\mathcal{M}|} \cdot \frac{1}{|\mathcal{M}_\lambda|} = \sum_{\lambda \in S} \frac{1}{|\mathcal{M}|} = \frac{|S|}{|\mathcal{M}|} = \frac{|S|}{900}$$

Hence the trivial advantage of an attacker is

$$\mathbf{TrivAdv}(\Sigma, \mathcal{M}, \mathcal{P}) = \left(\frac{|S|}{900} - \frac{1}{900}\right) \div \left(1 - \frac{1}{900}\right) = \frac{|S| - 1}{899}$$

The results are illustrated in Table 4.1.

---

<sup>1</sup>The computation is the same as the one presented in [12].

**Table 4.1:** Theoretical analysis of the simple PDF fingerprinting scenario. The different columns represent the mode of encryption considered *Mode*, the padding parameter that specifies the amount of padding applied to the plaintexts to ensure they are a multiple of  $\ell$  bytes, the number of uniquely identifiable PDF files  $|\text{pdf}_u|$ , the number of different ciphertext lengths possible  $|\mathcal{S}|$ , the size of the largest set of PDF files of identical size  $|\mathcal{S}_{\max}|$ , and the trivial advantage *TrivAdv*. The last column represents the overhead that occurs from using padding. The encryption protocol CCMP-128 utilizes CTR mode, and having  $\ell = 1$  is equivalent to not using padding at all. The figure illustrates the amount of padding needed to substantially reduce the trivial advantage, although this results in increased overhead.

Nr	Mode	$\ell$	$ \text{pdf}_u $	$ \mathcal{S} $	$ \mathcal{S}_{\max} $	TrivAdv	Overhead
1	CTR	1	898	899	2	0.9989	—
2	CTR	100	803	850	3	0.9444	0.01%
3	CTR	500	548	706	4	0.7842	0.04%
4	CTR	1000	336	565	6	0.6274	0.09%
5	CTR	10000	48	145	22	0.1602	0.91%
6	CTR	100000	17	37	176	0.040	9.02%
7	CTR	1000000	1	8	833	0.0078	102.10%
8	CTR	3797773	0	3	891	0.0022	582.34%
9	CTR	11393317	0	1	900	0.0000	1917.88%

#### 4.1. SIMPLE PDF FINGERPRINTING

**Table 4.2:** Classifier’s accuracy in the simple PDF fingerprinting scenario.

Classifier	Features Considered	A-MSDU disabled			A-MSDU enabled		
		k = 2	k = 10	k = 30	k = 2	k = 10	k = 30
Gaussian naïve Bayes	Packet lengths with directionality	97.9%	90.6%	84.7%	95.6%	82.6%	72.3%
Gaussian naïve Bayes	Total trace time, upstream/downstream total bytes and number of packets, number of traffic bursts, and bytes in traffic bursts	98.9%	86.3%	67.5%	98.1%	78.8%	56.1%

##### 4.1.2 The classifier’s accuracy

Table 4.2 illustrates the classifier’s accuracy in the simple PDF fingerprinting scenario using a Gaussian naïve Bayes classifier, which considered two types of features. The first type of feature is ‘packet lengths with directionality.’ Recall from Section 3.3.2 that each time a packet is observed with length  $L$  going in direction  $D$ , the value in column  $L\_D$  is incremented. For instance, if the attacker only observes the following traffic:

$$(\text{up}, 400), (\text{down}, 1500), (\text{down}, 1500), (\text{up}, 200)$$

then the columns  $400\_up$  and  $200\_up$  have the value 1, the column  $1500\_down$  has the value 2, and all other columns have the value 0.

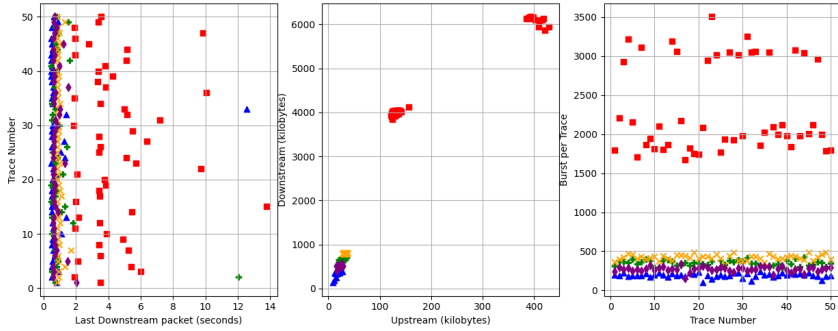
The second type of features are total trace time, upstream/downstream total bytes and number of frames, number of traffic bursts, and bytes in traffic bursts. Total trace time is calculated as the time delta between the first packet observed and the last observed downstream packet given a trace, and traffic burst is defined as a sequence of packets having the same directionality.

Given the feature set, the columns are split into two categories: A-MSDU disabled and A-MSDU enabled. The accuracy, as described in Section 3.3.3, is calculated by performing  $t = \lceil \frac{4000}{k \cdot 20} \rceil$  number of trials for a given  $k$ . The dataset, consisting of 20 traces times  $k$  data points, is split 100 times to calculate an average score for each trial. The final accuracy scores presented in Table 4.2 are the averages taken over all trials within a scenario, given a choice of  $k$  and a feature set.

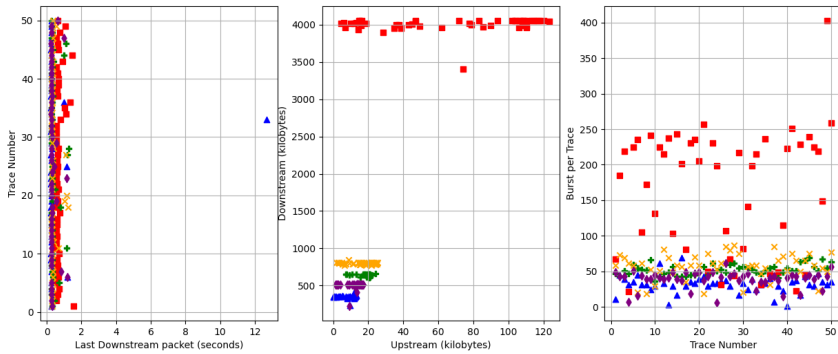
For  $k = 2$ , the lowest achieved accuracy was 95.6%, with the highest accuracy being 98.9%. The results show that the classifier’s accuracy decreased as the size  $k$  increased. In the case of  $k = 30$ , the lowest achieved accuracy was 56.1%, with the highest accuracy being 84.7%. Comparing the results from A-MSDU enabled with A-MSDU disabled shows a slightly lower accuracy score. The table



## 4.1. SIMPLE PDF FINGERPRINTING



**Figure 4.2:** Each scatterplot visualize metadata of encrypted 802.11 frames. Each colour and shape represent a specific PDF file, where each PDF file was fetched 50 times (each shape appears 50 times in each plot). Five different PDF files were fetched.



**Figure 4.3:** Each scatterplot is a visual representation of metadata of encrypted 802.11 frames related to PDF files. Figure follows the same format as Fig. 4.2, with A-MSDU enabled.

**Table 4.3:** Classifier's accuracy in the Wikipedia fingerprinting scenario.

Classifier	Features Considered	A-MSDU disabled			A-MSDU enabled		
		k = 2	k = 10	k = 30	k = 2	k = 10	k = 30
Gaussian naïve Bayes	Packet lengths with directionality	98.3%	94.1%	91.4%	97.9%	91.7%	89.1%
Gaussian naïve Bayes	Total trace time, upstream/downstream total bytes and number of packets, number of traffic bursts, and bytes in traffic bursts	86.7%	52.7%	36.9%	88%	62.3%	35.2%

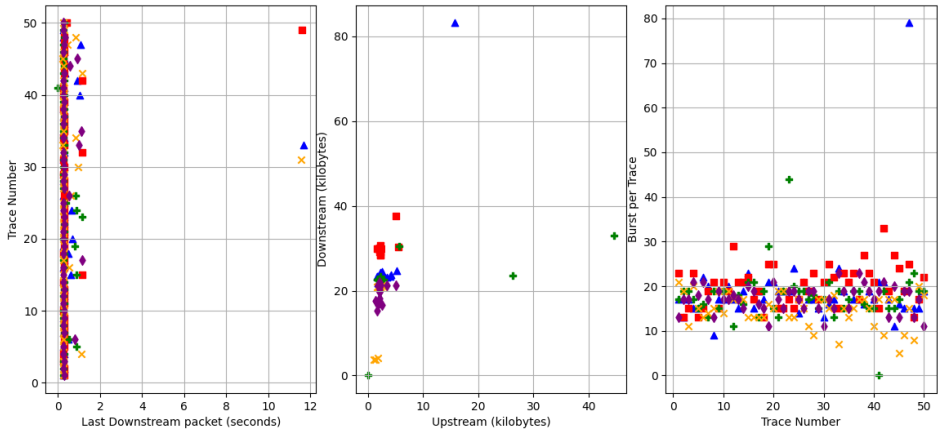
## 4.2 Wikipedia fingerprinting

The second scenario involved an attacker being aware that the user was accessing the Wikipedia domain and wanted to determine which Wikipedia pages the user was accessing.

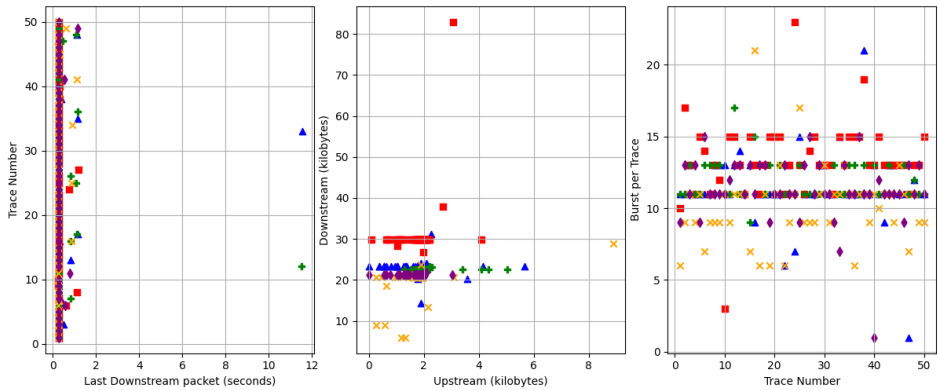
Table 4.3 illustrates the classifier's accuracy in the Wikipedia fingerprinting scenario. For  $k = 2$ , the highest accuracy achieved was 98.3%, where 86.7% was the lowest. The classifier achieved the highest accuracy of 91.4% when  $k = 30$ , where lowest achieved was 35.2%. As with the simple PDF fingerprinting scenario, the classifier's accuracy decreased as the size of the privacy set increased, where considering packet lengths with directionality achieved higher accuracy than the other feature set. The result includes an anomaly when  $k = 10$ , when comparing A-MSDU disabled with A-MSDU enabled. If only packet lengths with directionality are considered, the lowest obtained accuracy for  $k = 2$  and  $k = 30$  was 97.9% and 89.1%, respectively.

Visualizations of side channel information are illustrated in Figs. 4.4 and 4.5. As in the simple PDF fingerprinting scenario, the figures illustrate how side channel information can be visualized, comparing when A-MSDU is disabled with A-MSDU enabled. It follows the same format as the scatterplots explained in Section 4.1.2.

## 4.2. WIKIPEDIA FINGERPRINTING



**Figure 4.4:** Each scatterplot visualize representation of metadata of encrypted 802.11 frames related to Wikipedia pages. Each colour and shape represent a specific Wikipedia page, where each page was fetched 50 times (each shape appears 50 times in each plot). Five different Wikipedia pages were fetched.



**Figure 4.5:** Each scatterplot is a visual representation of metadata of encrypted 802.11 frames related to Wikipedia pages. Figure follows the same format as Fig. 4.4, with A-MSDU enabled.

**Table 4.4:** Classifier’s accuracy in the domain fingerprinting scenario.

Classifier	Features Considered	A-MSDU disabled			A-MSDU enabled		
		k = 2	k = 10	k = 30	k = 2	k = 10	k = 30
Gaussian naïve Bayes	Packet lengths with directionality	99.4%	96.5%	93.4%	99.2%	95.7%	92.3%
Gaussian naïve Bayes	Total trace time, upstream/downstream total bytes and number of packets, number of traffic bursts, and bytes in traffic bursts	91.2%	74.7%	55.8%	94.6%	66.5%	70.7%

### 4.3 Domain fingerprinting

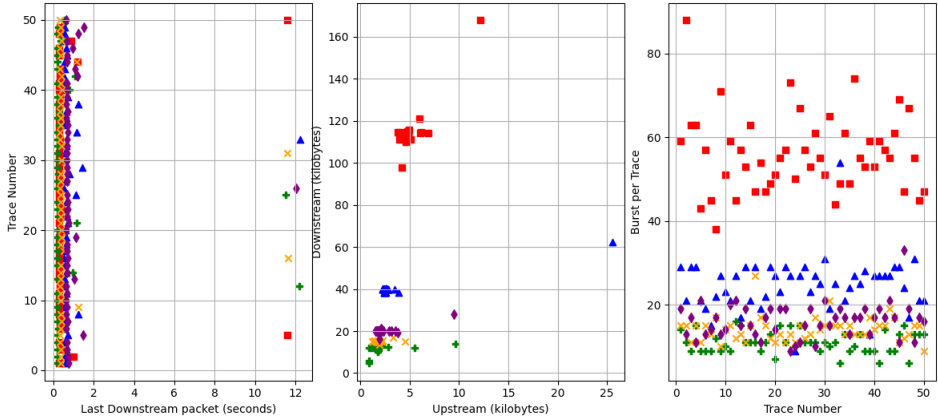
The third scenario involved a user accessing different domains. In this scenario, the attacker had no prior knowledge of what domains the user was accessing. The attacker’s goal was to identify which domains the user was visiting.

Table 4.4 illustrates the classifier’s accuracy in the domain fingerprinting scenario. For  $k = 2$ , the highest accuracy achieved was 99.4% and 91.2% the lowest. When  $k = 30$ , the classifier’s accuracy achieved a highest accuracy of 93.4%, where the lowest achieved accuracy was 55.8%. Similar to previous scenarios, the classifier’s accuracy decreased as the size of the privacy set increased and considering packet lengths with directionality achieved higher accuracy than the other feature set. However, the results have an anomaly for the case of  $k = 30$ , when comparing A-MSDU disabled with A-MSDU enabled. If only packet lengths with directionality are considered, the lowest obtained accuracy for  $k = 2$  and  $k = 30$  was 99.2% and 92.3%, respectively.

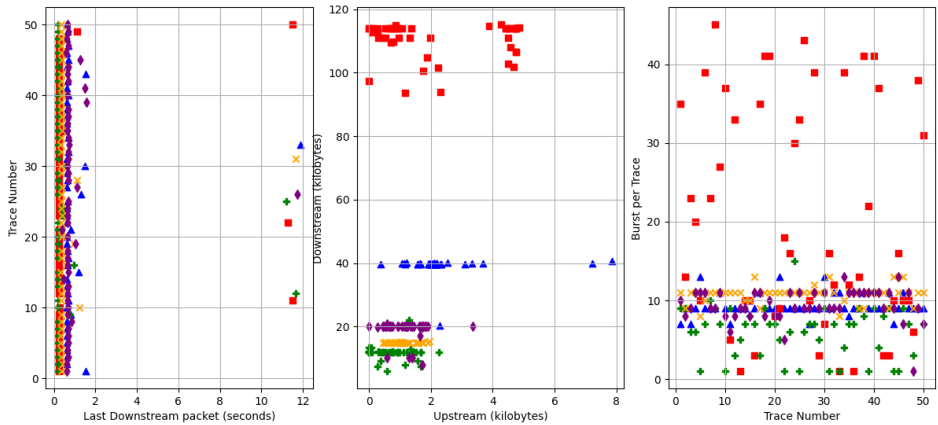
Visualizations of side channel information are illustrated in Figs. 4.6 and 4.7. The format is the same as in the previous scenarios.



### 4.3. DOMAIN FINGERPRINTING



**Figure 4.6:** Each scatterplot visualize representation of metadata of encrypted 802.11 frames related to domains. Each colour and figure represent a specific domain, where each domain was fetched 50 times (each figure appears 50 times in each plot). Five different domains fetched.



**Figure 4.7:** Each scatterplot is a visual representation of metadata of encrypted 802.11 frames related to domains. Figure follows the same format as Fig. 4.6, with A-MSDU enabled.

## CHAPTER 5

# Discussion

This chapter analyzes the results of the website fingerprinting attacks on encrypted 802.11 frames and compares the performed attack to the trivial success probability. It addresses the implications and limitations of these findings.

The chapter begins with discussing the results of the website fingerprinting attack in [Section 5.1](#) before examining the impact of aggregation in [Section 5.2](#). A comparison between the website fingerprinting attack and the trivial success probability is examined in [Section 5.3](#).

## 5.1 The accuracy of the website fingerprinting attack

The results show that the Gaussian naïve Bayes classifier achieved a minimal accuracy of 95% when distinguishing between two data types, when considering packet lengths with directionality. As the size of the privacy set increased, the accuracy decreased to a value between roughly 72% to 92%, depending on the scenario, when distinguishing 30 different data types. However, using packet lengths with directionality as features provided better accuracy than considering total trace time, upstream/downstream total bytes and number of packets, number of traffic bursts, and bytes in those bursts as the feature set. Despite the drop in accuracy with a larger privacy set, the accuracy obtained was still significantly better than selecting an element from the privacy set uniformly at random.

Previous research considered a fingerprinting attack on LTE/4G traffic [22], achieving a success rate of 92% to 95% in a controlled lab environment and around 90% success rate in a real-world scenario. The ‘Alexa top 50 websites’ list was used in these scenarios, i.e., their scenarios considered  $k = 50$ . Additional research on the failure of website fingerprinting countermeasures illustrated a 80% accuracy for  $k = 128$  [9]. In contrast, the accuracy achieved in this thesis was lower as the size of the privacy set increased when total trace time, upstream/downstream total bytes and number of packets, number of traffic bursts, and bytes in those bursts were among the features considered. This difference in accuracy is because of the smaller dataset and the simplicity of the classifier.

While the case of  $k = 2$  with  $t = 100$  trials indicates a real-world attacker’s accuracy, the dataset used in this thesis is relatively limited compared to previous studies [9, 12, 13, 16, 22, 28, 30, 39]. This limitation arises from the data being collected over several days rather than several months, resulting in a less credible accuracy, especially as the size of the privacy set increases, leading to a decrease in the number of trials  $t$ . However, the accuracy still indicates the success rate an attacker can expect. While more data would yield a more precise accuracy, it is unlikely that the results would differ significantly. In addition, this thesis considers a maximum value of  $k = 30$  per trial, which is relatively low compared to a real-world scenario. For instance, the research in [9] considered 128 and 775 different data types, which is more realistic. Moreover, the classifiers used in this research were more complex, and the authors focused more on the accuracy achieved. Fine-tuning hyperparameters and using more advanced machine learning models capable of capturing intricate patterns would likely result in higher accuracy. This thesis uses the Gaussian naïve Bayes with no hyperparameter tuning, requiring minimal expertise in classifiers. As a

consequence, the accuracy achieved in this thesis should be interpreted as a lower bound, demonstrating the potential of classifiers.

The scenarios considered in this thesis differ from previous research [9, 22], making a direct comparison of accuracy challenging despite the observed traffic sharing similar characteristics. This thesis considers total trace time, upstream/downstream total bytes and number of packets, the number of traffic bursts, and bytes in those bursts as a single feature set, which was considered individually in earlier studies [9]. Additionally, the study in [22] focused on LTE/4G traffic, which have fundamental differences compared to website fingerprinting of encrypted network traffic.

Similar to previous research on fingerprinting attacks on encrypted traffic [9, 12, 13, 16, 28, 30, 39], the results suggest that an attacker can execute a website fingerprinting attack on encrypted Wi-Fi traffic with a high success rate, particularly when using packet lengths with directionality as the feature set. As the size of the privacy set increases, the accuracy drops slower when packet lengths with directionality are considered. This suggests that with a larger dataset, the optimal strategy is considering packet lengths as the feature set.

Although there are limitations, the results indicate that the achieved accuracy aligns with findings from previous studies when focusing on the highest accuracy achieved. The results highlight the feasibility of performing a website fingerprinting attack on encrypted 802.11 frames, illustrating how information leakage undermines the security that encryption intuitively should provide. This emphasizes the importance of the underlying problem: while an encryption scheme may be cryptographically secure, it is not necessarily secure in practice if the encrypted network traffic leaks information through side channels.

Experimental results do not provide an exact accuracy measurement. They simulate a setting intended to reflect a real-world scenario. The computers in the experiments were physically connected, ensuring data quality with the client's background applications disabled. In a real-world scenario, attackers must distinguish traffic based on observed encrypted 802.11 frames, which is challenging. This issue is well known, where correctly distinguishing data is essential for obtaining quality data that attacks can exploit.

## 5.2 Impact of aggregation

The results indicate that enabling aggregation led to a general reduction in the classifier's accuracy, although some anomalies were observed. However, this reduction was minimal and did not substantially reduce the accuracy. In

the simple PDF fingerprinting scenario, the gap in classifier accuracy between A-MSDU enabled and disabled grew as the size of the privacy set increased. In the remaining scenarios, the reduction in accuracy remained relatively constant.

The impact of padding on encrypted network traffic has been previously explored in prior studies [9, 28]. One study [28] examined traffic analysis attacks on HTTPS traffic using classifiers. The study demonstrated that padding all packet sizes up to the nearest power of 2 significantly reduced the attacker's success rate from 60% to 22% for one type of attack and 89% to 59% for another. The other study [9] compared various padding techniques and found that none of the nine countermeasures effectively prevented website fingerprinting attacks, given the scenario considered. Applying the best-performing countermeasure reduced the accuracy to around only 85%. The results from both studies indicate that padding affects the classifier's accuracy, even though they do not necessarily prevent website fingerprinting attacks. Anomalies in this thesis indicated increased accuracy when aggregation was enabled. These instances occurred when total trace time, upstream/downstream total bytes and number of packets, number of traffic bursts, and bytes in those bursts were the features considered. One reason for this is the impact of aggregation on traffic patterns.

By comparing Figs. 4.3, 4.5 and 4.7 with Figs. 4.2, 4.4 and 4.6, the figures demonstrate how A-MSDU modifies specific traffic patterns of the encrypted Wi-Fi traffic, with the former showing A-MSDU enabled and the latter displaying A-MSDU disabled. While the change in traffic patterns result in reduced classifier accuracy, the results indicate that, in some instances, this action benefits the classifier. Since this effect is inconsistent across all cases, a possible conclusion is that the impact of aggregation depends on the characteristics of the different data types. If the data share similarities, aggregation might reveal distinguishing information, aiding the classifier. Conversely, the results suggest that aggregation alters the traffic patterns where A-MSDU clusters specific characteristics, making it more challenging for the classifier to differentiate.

While the results indicate that aggregation can assist the classifier in specific instances, the findings suggest that enabling A-MSDU tends to reduce the classifier's accuracy. Previous research [12] focusing on length-hiding encryption in a theoretical context has shown that even a modest amount of padding can substantially reduce the adversary's advantage, with only a slight increase in bandwidth overhead. However, the results indicate that padding introduced by aggregation does not substantially lower the success rate in practical scenarios. This implies that relying solely on aggregation as a mitigation technique against website fingerprinting attacks is insufficient, as it only marginally reduces the accuracy of the classifier. A-MSDU should still be

employed to reduce header overhead and increase bandwidth throughput.

This thesis does not precisely measure aggregation's impact or explain the observed anomalies. While the data suggests that aggregation generally lowers classifier accuracy, the effect is inconsistent. Additionally, while the figures demonstrate changes in traffic patterns when aggregation is enabled, they do not explain the improved accuracy in certain situations, such as in the Wikipedia and domain fingerprinting scenarios. A possible hypothesis is that alternation in the traffic patterns may benefit the classifier under certain conditions. Further analysis would be required to provide an accurate explanation, as time constraints limited the ability to conduct a thorough investigation.

### 5.3 Accuracy and the trivial success probability

For the simple PDF fingerprinting scenario, the trivial success probability is given by  $\frac{|S|}{900}$ , which results in a trivial advantage of  $\frac{|S|-1}{899}$ . Recall that the trivial advantage is a scaled value between 0 and 1 depending on the trivial success probability. The results show that there were 899 different ciphertext lengths when no padding was applied, resulting in a trivial advantage of  $0.9989 \approx 1$ . As the size of  $\ell$  increased, the trivial advantage decreased. However, the results indicate that a small amount of padding slightly reduced the trivial advantage. A significant amount of padding is needed to ensure no uniquely identifiable PDF files or all PDF files have the same length.

This calculation is based on previous research [12], which quantitatively measured the effectiveness of length-hiding encryption in concealing information about the message length. In one of their scenarios, they considered a simple website fingerprinting attack where a client visits a website consisting of many static HTML pages. In this case, the trivial success probability was  $\frac{|S|}{1620}$ . Their results showed that without padding, the trivial advantage was 0.74. However, when padding to a multiple of 80, the advantage decreased to 0.0327, with an overhead of only 1.158%. They concluded that a surprisingly small amount of padding may significantly reduce an adversary's advantage. This thesis suggests, in contrast, that while a small amount of padding reduces an attacker's trivial advantage, the reduction is much smaller compared to the results in previous research. Padding to a multiple of approximately 3797773 in this thesis reduces the trivial advantage, similar to the effect of padding to a multiple of 80 in the previous research. This difference is due to the type of data considered.

The previous research focused on a client visiting a website with many static HTML pages. These pages share a similar, plain text formatting, with references to external resources that need to be fetched when rendered on the

client side. This thesis, however, considers PDF files, which may include text, images, graphics, and embedded elements, all of which affects their size. PDF files are typically much larger than static HTML pages, where each PDF file ranges from a few pages to many. This causes the size range of PDF files to be much broader than that of static HTML pages, leading to more unique file lengths in the case of PDF files. This reflects the trivial advantage of an adversary, where no padding is applied, resulting in 0.9989 compared to 0.74. This explains why the effectiveness of the padding is considerably reduced.

The findings indicate that an adversary can achieve high accuracy by only focusing on the message length in the simple PDF fingerprinting scenario. Due to the significant differences in file sizes, the results show that substantial padding is required to prevent leakage of message length through side channels. This comes at a cost of considerable overhead. However, the trivial advantage is notably greater than the accuracy of the classifier. While the achieved accuracy in both cases is similar for  $\ell = 1$  and  $k = 2$ , the trivial advantage holds for all 900 PDF files hosted on the web server. This results in a much higher trivial advantage compared to the success rate of a website fingerprinting attack when packet lengths with directionality are used as features. As the classifier's accuracy declines from approximately 99% to 93% for  $k = 30$ , the results suggest that for  $k = 900$  is significantly worse than the trivial advantage in the theoretical model. These findings demonstrate that focusing solely on message length can provide insights into information leakage through side channels and how trivial success probability serves as a proof-of-concept. However, the comparison shows that the theoretical model does not accurately represent the success rate of a conducted website fingerprinting attack.

The results do not accurately measure of how well the trivial advantage reflects a real-world adversary. The proof-of-concept aims to demonstrate the leakage of message length through 802.11 frames in an idealized scenario, assuming a one-to-one correspondence between the length of PDF files and their corresponding encrypted 802.11 frames. This comparison can provide a clearer understanding of how theoretical measures relate to actual website fingerprinting attacks. However, as it stands, the results can only indicate an attacker's success probability, not an accurate measure.

## CHAPTER 6

# Conclusion

This chapter will provide a summary and a conclusion to the research questions presented in [Chapter 1 Section 1.2.2](#) in [Section 6.1](#). The chapter concludes with potential directions for future work in [Section 6.2](#).



## 6.1 Summary

Previous research [9, 12, 13, 16, 28, 30, 39] has demonstrated the feasibility of performing a fingerprinting attack on encrypted network traffic. However, it is not obvious that such an attack would be effective at the data link layer.

This thesis aims to demonstrate the impact of a website fingerprinting attack on encrypted 802.11 frames and evaluate the information leakage in encrypted wireless network traffic. An experiment was designed with three different scenarios, starting with a controlled scenario with limited variables before gradually moving toward a more realistic one. In each scenario, an attacker used a classifier to execute a website fingerprinting attack, focusing on specific features observed through side channels. The first scenario examined the correlation between the attack's performance and the information leaked through the lengths of encrypted messages in a theoretical setting. The effectiveness of aggregation as a mitigation technique was also assessed, with each scenario considered with and without aggregation. The results answer the following research questions.

### **Q.1 What is the baseline accuracy of an attacker performing a website fingerprinting attack on encrypted Wi-Fi traffic, considering different traffic features and varying conditions?**

When distinguishing between two data types, an attacker can achieve a minimum accuracy of 95%, which drops to at least 72% when distinguishing between 30 different types of data. If the attacker's aim is distinguishing different domains, the accuracy increases to approximately 99% and 92%, respectively. While the classifier's accuracy varies depending on the scenario considered, the achieved accuracy is still significantly higher than guessing uniformly at random. The results highlight the feasibility of performing website fingerprinting attacks on encrypted 802.11 frames, illustrating how information leakage undermines the security that encryption intuitively should provide.

### **Q.2 How does A-MSDU frame aggregation impact the accuracy of a website fingerprinting attack on encrypted Wi-Fi traffic?**

The results suggest that aggregation reduces the classifier's accuracy, even though the reduction is minimal and does not prevent an attacker from successfully performing a website fingerprinting attack. Findings suggest that aggregation can even aid the classifier in distinguishing types of traffic, such as when the classifier is trained on traffic patterns like traffic bursts. Despite aggregation being unable to protect against website fingerprinting attacks, it still reduces the header overhead and improves bandwidth throughput.

**Q.3 How does the accuracy of a website fingerprinting attack on encrypted Wi-Fi traffic compare to the trivial success probability when distinguishing between different data types within a single domain?**

By comparing the trivial advantage, based on the trivial success probability, with the accuracy of the website fingerprinting attack, the results show that the trivial success probability does not accurately reflect the success rate of an attacker performing the attack. However, using the theoretical model as a starting point still offers insights into how message length leaks through encrypted 802.11 frames via side channels, indicating what to expect from the 802.11 standard in the absence of mitigation techniques.

## 6.2 Future work

This thesis's data collection and analysis have demonstrated the feasibility of an attacker executing a website fingerprinting attack on encrypted 802.11 frames. The results illustrate the effectiveness of using machine learning algorithms to achieve high accuracy and demonstrate the difficulty of hiding information leakage of encrypted traffic.

Based on the findings and limitations of this thesis, future work should consider the following.

- **Obtaining quality data:** Two computers physically connected are considered in this thesis, ensuring the attacker can obtain quality data to train the classifier. However, in a real-world scenario, a challenge is how an attacker can observe encrypted traffic and confidently split it to distinguish between data from different applications. Solving this well-known issue could add to the attacker's capability and enable them to perform fingerprinting attacks in parallel, targeting multiple clients simultaneously. Future research could focus on how an adversary could confidently differentiate data types by purely focusing on encrypted 802.11 frames when executing a website fingerprinting attack.
- **Aggregation:** While this thesis explores aggregation as a padding technique to reduce the success rate of an attacker performing a website fingerprinting attack, future research could examine two aspects. First, it could investigate whether A-MSDU can improve the accuracy in specific situations and identify the threshold at which it begins to either increase or decrease the accuracy of the classifier. Second, it could examine how reducing the size of MSDUs within A-MSDU impacts the accuracy of website fingerprinting attacks in various scenarios. However, one should

not rely on aggregation as the only defense against such attacks. Future research should not expect A-MSDU to prevent such attacks.

- **Theoretical security:** This thesis draws on the theoretical model in [12]; future work could extend this model to capture other features leaked through side channels, examining how they impact the trivial success probability of an attacker. This would help bridge the gap between theoretical and practical settings. If the theoretical model could capture different features, it would provide deeper insights into an attacker's success rate within a theoretical framework.

# Bibliography

- [1] Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)* (2021), 1–4379. 21, 23, 24, 26, 27
- [2] ABDELALIM, K. *Study and optimisation of IEEE 802.11 PHY and MAC protocols towards a new generation integrated in 5G*. PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire, 2019. 20, 21
- [3] BANERJI, S., AND CHOWDHURY, R. S. On ieee 802.11: wireless lan technology. *arXiv preprint arXiv:1307.2661* (2013). 4, 20
- [4] BELLARE, M., AND ROGAWAY, P. Code-based game-playing proofs and the security of triple encryption. *Cryptology ePrint Archive* (2004). 13
- [5] BHANAGE, G. Amsdu vs ampdu: A brief tutorial on wifi aggregation support. *arXiv preprint arXiv:1704.07015* (2017). 26, 27
- [6] BRZUSKA, C., AND JACOBSEN, H. A modular security analysis of eap and ieee 802.11. In *IACR International Workshop on Public Key Cryptography* (2017), Springer, pp. 335–365. 20, 23, 24
- [7] DEGABRIELE, J. P. Hiding the lengths of encrypted messages via gaussian padding. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19,*

- 2021 (2021), Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds., ACM, pp. 1549–1565. 2, 3, 8, 19
- [8] DEGABRIELE, J. P., AND PATERSON, K. G. Attacking the ipsec standards in encryption-only configurations. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA* (2007), IEEE Computer Society, pp. 335–349. 7
- [9] DYER, K. P., COULL, S. E., RISTENPART, T., AND SHRIMPTON, T. Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA* (2012), IEEE Computer Society, pp. 332–346. 3, 4, 7, 8, 28, 43, 64, 65, 66, 70
- [10] FOROUZAN, B. A. *Data communications and networking*. Huga Media, 2007. 20, 21
- [11] GAST, M. *802.11 wireless networks: The definitive guide*. Southeast University Press Nanjing, JiangSu, China, 2006. 20
- [12] GELLERT, K., JAGER, T., LYU, L., AND NEUSCHULTEN, T. On fingerprinting attacks and length-hiding encryption. In *Topics in Cryptology - CT-RSA 2022 - Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1-2, 2022, Proceedings* (2022), S. D. Galbraith, Ed., vol. 13161 of *Lecture Notes in Computer Science*, Springer, pp. 345–369. 2, 3, 4, 8, 15, 16, 18, 19, 39, 54, 64, 65, 66, 67, 70, 72
- [13] GONG, X., BORISOV, N., KİYAVASH, N., AND SCHEAR, N. Website detection using remote traffic analysis. In *Privacy Enhancing Technologies - 12th International Symposium, PETS, 2012, Vigo, Spain, July 11-13, 2012. Proceedings* (2012), S. Fischer-Hübner and M. K. Wright, Eds., vol. 7384 of *Lecture Notes in Computer Science*, Springer, pp. 58–78. 3, 64, 65, 70
- [14] HARRINGTON, P. *Machine learning in action*. Simon and Schuster, 2012. 28
- [15] HERRMANN, D., WENDOLSKY, R., AND FEDERRATH, H. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the first ACM Cloud Computing Security Workshop, CCSW 2009, Chicago, IL, USA, November 13, 2009* (2009), R. Sion and D. Song, Eds., ACM, pp. 31–42. 51
- [16] HINTZ, A. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies, Second International Workshop, PET, 2002, San*

- Francisco, CA, USA, April 14-15, 2002, *Revised Papers* (2002), R. Dingledine and P. F. Syverson, Eds., vol. 2482 of *Lecture Notes in Computer Science*, Springer, pp. 171–178. 3, 64, 65, 70
- [17] JEAN, J. TikZ for Cryptographers. <https://www.iacr.org/authors/tikz/>, 2016. 25
- [18] JORDAN, M. I., AND MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science* 349, 6245 (2015), 255–260. 28
- [19] JUNAID, M., MUFTI, M., AND ILYAS, M. U. Vulnerabilities of ieee 802.11 i wireless lan ccmp protocol. *White Paper, electronically available at: http://whitepapers.techrepublic.com.com/whitepaper.aspx* (2006). 24
- [20] KELSEY, J. Compression and information leakage of plaintext. In *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers* (2002), J. Daemen and V. Rijmen, Eds., vol. 2365 of *Lecture Notes in Computer Science*, Springer, pp. 263–276. 7
- [21] KENT, S. Ip encapsulating security payload (esp). RFC 4303, RFC Editor, December 2005. <http://www.rfc-editor.org/rfc/rfc4303.txt>. 7
- [22] KOHLS, K., RUPPRECHT, D., HOLZ, T., AND PÖPPER, C. Lost traffic encryption: fingerprinting lte/4g traffic on layer two. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks* (2019), pp. 249–260. 4, 8, 9, 64, 65
- [23] KUMAR GUPTA, A., AND VENKATESH, T. Design and analysis of ieee 802.11 based full duplex wlan mac protocol. *Computer Networks* 210 (2022), 108933. 21
- [24] LANGLEY, A. A transport layer security (tls) clienthello padding extension. RFC 7685, RFC Editor, October 2015. 7
- [25] LE POCHAT, V., VAN GOETHEM, T., TAJALIZADEHKHOOB, S., KORCZYŃSKI, M., AND JOOSEN, W. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium* (Feb. 2019), NDSS 2019. 41
- [26] LIBERATORE, M., AND LEVINE, B. N. Inferring the source of encrypted HTTP connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006* (2006), A. Juels, R. N. Wright, and S. D. C. di Vimercati, Eds., ACM, pp. 255–263. 51

- [27] MANSOUR, K., JABRI, I., AND EZZEDINE, T. Revisiting the ieee 802.11n a-mpdu retransmission scheme. *IEEE Communications Letters* 23, 6 (2019), 1097–1100. 26
- [28] MILLER, B., HUANG, L., JOSEPH, A. D., AND TYGAR, J. D. I know why you went to the clinic: Risks and realization of https traffic analysis. In *Privacy Enhancing Technologies - 14th International Symposium, PETS, 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings* (2014), E. D. Cristofaro and S. J. Murdoch, Eds., vol. 8555 of *Lecture Notes in Computer Science*, Springer, pp. 143–163. 3, 64, 65, 66, 70
- [29] MOISSINAC, K., RAMOS, D., RENDON, G., AND ELLEITHY, A. Wireless encryption and wpa2 weaknesses. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)* (2021), pp. 1007–1015. 24
- [30] MUEHLSTEIN, J., ZION, Y., BAHUMI, M., KIRSHENBOIM, I., DUBIN, R., DVIR, A., AND PELE, O. Analyzing HTTPS encrypted traffic to identify user’s operating system, browser and application. In *14th IEEE Annual Consumer Communications & Networking Conference, CCNC 2017, Las Vegas, NV, USA, January 8-11, 2017* (2017), IEEE, pp. 1–6. 3, 64, 65, 70
- [31] NAKHILA, O., AND ZOU, C. Parallel active dictionary attack on ieee 802.11 enterprise networks. In *MILCOM 2016-2016 IEEE Military Communications Conference* (2016), IEEE, pp. 265–270. 23
- [32] PATERSON, K. G., AND ALFARDAN, N. J. Plaintext-recovery attacks against datagram TLS. In *19th Annual Network and Distributed System Security Symposium, NDSS, 2012, San Diego, California, USA, February 5-8, 2012* (2012), The Internet Society. 7
- [33] PATERSON, K. G., RISTENPART, T., AND SHRIMPTON, T. Tag size does matter: Attacks and proofs for the TLS record protocol. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings* (2011), D. H. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Computer Science*, Springer, pp. 372–389. 2, 3, 15, 19
- [34] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M.,

- AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830. 36
- [35] ROSS, D. A. *Securing IEEE 802.11 Wireless LANs*. PhD thesis, Queensland University of Technology, 2010. 24
- [36] TEZCAN, C., AND VAUDENAY, S. On hiding a plaintext length by preencryption. In *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings* (2011), J. López and G. Tsudik, Eds., vol. 6715 of *Lecture Notes in Computer Science*, pp. 345–358. 2, 3, 15, 19
- [37] VELAYUTHAM, R., AND MANIMEGALAI, D. Ccmp advanced encryption standard cipher for wireless local area network (ieee 802.11i): A comparison with des and rsa. *Journal of Computer Science* 11, 2 (Aug 2014), 283–290. 23, 24
- [38] WANG, T., CAI, X., NITHYANAND, R., JOHNSON, R., AND GOLDBERG, I. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)* (2014), pp. 143–157. 37
- [39] WANG, T., AND GOLDBERG, I. On realistically attacking tor with website fingerprinting. *Proc. Priv. Enhancing Technol.* 2016, 4 (2016), 21–36. 3, 64, 65, 70
- [40] YLONEN, T., AND LONVICK, C. The secure shell (ssh) transport layer protocol. RFC 4253, RFC Editor, January 2006. <http://www.rfc-editor.org/rfc/rfc4253.txt>. 7



## BIBLIOGRAPHY

---

## APPENDIX A

# Frame control subfields

---

<----- <i>Frame Control</i> ----->										
<i>Protocol Version</i>	<i>Type</i>	<i>Subtype</i>	<i>To DS</i>	<i>From DS</i>	<i>More Frag.</i>	<i>Retry</i>	<i>Power Manag.</i>	<i>More Data</i>	<i>Protected Frame</i>	<i>Order / +HTC</i>

<b>Name</b>	<b>Size</b>	<b>Description</b>
Protocol Version	2 bits	Information about what specific protocol from the 802.11 standard is used.
Type	2 bits	Defines what type of frame it is.
Subtype	4 bits	Defines what kind of subtype the frame is.
To DS	1 bit	Information about the direction of the frame.
From DS	1 bit	Information about the direction of the frame.
More Fragments	1 bit	Indicates if more fragments are to follow or not.
Retry	1 bit	Information whether the frame is retransmitted from the sender or not.
Power Management	1 bit	Tells the receiver if the sender is in active mode or in power-save mode.
More Data	1 bit	Indicates if there are further data frames to follow.
Protected Frame	1 bit	Informs if encryption and authentication is used in the frame. Can be set for data frames, if authentication is set in the subtype field.
Order / +HTC	1 bit	Informs if the frames must be processed in a ordered sequence or not.

**Figure A.1:** The Frame Control Subfields.

## APPENDIX B

# Padding strategies

<b>Type-1: SSH/TLS/IPsec-Motivated countermeasures</b>	
<b>Session Random 255</b>	Apply a single amount of padding to all plaintexts in the session. Sample uniformly at random a value $r$ such that $r \in \{x   x = 8n \text{ for } n \in \mathbb{N} \text{ and } 0 \leq x \leq 248^1\}$ . For every packet in the trace, increase the packet length by $r$ or up to the MTU <sup>2</sup> .
<b>Packet Random 255</b>	Sample uniformly at random a value $r$ for each single packet $P$ , where $r \in \{x   x = 8n \text{ for } n \in \mathbb{N} \text{ and } 0 \leq x \leq 248\}$ . Increase the padding length by $r$ up to the MTU.
<b>Type-2: Other Padding-based countermeasures</b>	
<b>Linear Padding:</b>	Increase the packet length to the nearest multiple of 128 or the MTU, whichever is smaller.
<b>Exponential Padding:</b>	Increase the packet length to the nearest power of 2 or the MTU, whichever is smaller.
<b>Mice-Elephants Padding:</b>	For a packet $P$ with length $ P $ do the following: If $ P  \leq 128$ , then pad the packet $P$ to 128. If not, then pad the packet $P$ to the MTU.
<b>Pad To MTU:</b>	All packets are padded to the MTU.
<b>Packet Random MTU Padding:</b>	Sample uniformly at random a value $r$ for each packet $P$ , where $r \in \{x   x = 8n \text{ for } n \in \mathbb{N} \text{ and } 0 \leq x \leq \text{MTU} -  P \}$ . Increase the packet length by $r$ .
<b>Type-3: Distribution-based countermeasures</b>	
<b>Direct Target Sampling:</b>	Two different padding countermeasures. Both enlarge a protocol's packet by padding and chopping, with the intention of deceiving an attacker. This is done by making it look like the enlarged packets come from a different web pages which is pre-defined.
<b>Traffic Morphing:</b>	

**Figure B.1:** Table of various padding strategies categorized.