Explicit and Implicit Information Leakage in Wireless Communication

Abstract

Information security vulnerabilities in wireless communication are caused by unforeseen behavior of a system as a result of flawed protocol designs or implementations. Such vulnerabilities may be exploited by an adversary in order to break the confidentiality, integrity and availability provided by a system, or compromise the privacy of its users. In this thesis, we focus on one particular class of vulnerabilities, namely information leakage, in context of two wireless protocols: Wi-Fi (802.11) and LoRa. With over 454 million Wi-Fi hotspots and 500,000 LoRa gateways forecast to be operational by 2020, these protocols are amongst the most popular wireless protocols in use at the time of writing. We will distinguish between and explore two types of information leakage, which we refer to as explicit information leakage and implicit information leakage.

In the first main part of the thesis we will examine *explicit information leakage*, which stems from unintended flaws in the design or implementation of wireless protocols. More specifically, we will reveal a vulnerability in the 802.1X PEAP protocol, which is used as an authentication method in WPA2-Enterprise networks. This vulnerability allows an adversary to relay challenge responses from a LEAP handshake as valid credentials for a PEAP handshake, thereby gaining unauthorized access to the network. We show that this attack works on all Apple devices prior to iOS 8, OS X Yosemite and Apple TV 7. Next, we look at the MAC-layer frame aggregation mechanism introduced in 802.11n, and show how an adversary can abuse the delimiter scanning algorithm to remotely inject arbitrary Wi-Fi frames into an open network, even without requiring a radio. This is achieved by crafting a specific application-layer payload that leaks to the lower layers of the network stack when the A-MPDU delimiter is corrupted by incidental noise. We then analyze the information broadcasted in Wi-Fi `Probe Request` frames, and show that such frames leak sufficient information about the transmitting device to create a unique fingerprint. We show that this can be exploited to defeat privacy-preserving measures such as MAC address randomization in a large-scale field experiment, where data was gathered at a music festival over a two-day period. Moreover, we introduce a number of techniques to increase the frequency of `Probe Request` transmissions, for example by transmitting specially crafted `GAS Request` and `ADDBA Request` frames. For each of the vulnerabilities

discussed in this part of the thesis, we propose countermeasures to mitigate their impact and improve the security and privacy of users.

The second main part of the thesis looks into *implicit information leakage*, which originates from measurable side effects of a software or hardware implementation of a protocol. As a first example, we show how frequency offset errors introduced by the hardware of LoRa devices leak sufficient information to fingerprint individual devices on the physical layer. Complementary to this finding, we release an open source implementation of a novel demodulation algorithm, based on the gradient of the instantaneous frequency of a LoRa signal, that allows to synchronize to a LoRa signal while preserving any present frequency offset errors. Using this algorithm, we capture 8 datasets of LoRa symbols using an SDR and analyze the classification accuracy under various environments when using SVM, MLP and CNN classifiers. We also perform a brief experiment with zero-shot classification techniques, where LoRa devices can be classified without having access to prior training data about these devices. Finally, we consider the electromagnetic (EM) side-channel leakage of the AES cipher, which is used by both Wi-Fi and LoRa. In particular, we examine the application of machine learning and deep learning on EM traces leaked during the execution of AES, and propose a novel methodology to find the secret key based on these traces. Our methodology requires only a few minutes of training time on commodity hardware due to a less complex architecture, while outperforming state-of-the-art deep learning algorithms on the ASCAD benchmark dataset. Additionally, we show that the requirement of having to align signals prior to performing a CEMA attack can be removed by applying our methodology in the frequency domain of the captured EM traces, and provide a practical proof-of-concept by using a USRP B210 to attack an AES implementation running on an Arduino Duemilanove.

Acknowledgements

First and foremost, I would like to thank my parents, whose continued love and support makes me feel very happy, welcome and fortunate whenever I come home. I am forever indebted to their unconditional kindness and the opportunities they gave me. Their dedication and ability to help their children is inspiring, and I hope that I may one day be able to care for others like my parents cared for me. Thanks, mom and dad.

I would like to thank Bart, my childhood mentor, for having spent many hours of his time teaching me how to program Flash games at the age of 11, and for inspiring me to study computer science. His lessons benefited me greatly, and I am very thankful for that.

I would like to thank each and every one of my friends, for their companionship in both good and bad times. Be it in Limburg, Antwerp, or in Flemish Brabant, they lift my spirits with every meeting, and they always manage to offer a listening ear whenever I wear my heart on my sleeve (or play the guitar). I especially thank Benno, who has been my friend since the day I was born, and who will always choose my side.

I would like to thank my current and former colleagues at Hasselt University, EDM, KU Leuven and Northeastern University. Thanks to you, I became passionate about information security, obtained the knowledge and means to pursue a Ph.D., and became able to do what I love most. In particular, I thank my advisor Wim and co-advisor Peter for believing in me, for their time and help, and for giving me the freedom to study and write about such a diverse spectrum of topics within the field. I would also like to thank the FWO for accepting my research proposal and for their financial support. I feel honored and very fortunate to be part of this community, and I hope to keep contributing to the field in the foreseeable future.

Finally, I would like to thank my Ph.D. jury, for the time they invested into reviewing this thesis, for providing their insightful feedback, and for their willingness to travel such a long way in order to attend my defense.

Thanks so much to all of you; without you, this thesis would not have been possible.

– Pieter Robyns

Contents

Acronyms

ACK Acknowledgement. 105–108

ADDBA Add Block Ack. 108, 117, 118, 120–122, 124, 126

AES Advanced Encryption Standard. 7, 29, 32, 179, 180, 182, 184, 186, 198, 199, 210, 224

AI Artificial Intelligence. 27

ANN Artificial Neural Network. 29, 33

ANQP Access Network Query Protocol. 105, 108, 109, 118

AP Access Point. 40–42, 47–52, 62, 66–70, 72, 73, 76, 78, 80, 86, 91–94, 98, 99, 103, 105, 108, 110, 111, 116, 120, 122, 123, 125, 126, 207–209, 223

ARP Address Resolution Protocol. 72

AS Authentication Server. 41–50, 52, 207

BA Block Acknowledgement. 88

BSS Basic Service Set. 87, 107, 109, 111, 125

BSSID Basic Service Set Identification. 73, 107, 110, 116–119, 121

BYOD Bring Your Own Device. 40, 50, 52

CCA Clear Channel Assessment. 119

CCK Complementary Code Keying. 59

CEMA Correlation Electromagnetic Analysis. 178, 179, 182, 184, 189–197, 199, 202, 203, 210, 224

CFO Carrier Frequency Offset. 144, 145, 149–151, 160, 166–168

CIA Confidentiality, Integrity, and Availability. 2, 4, 40

CMK Compound MAC Key. 51

CNN Convolutional Neural Network. 30, 31, 34, 161, 162, 168, 172, 193, 200–202, 209

CO Correlation Optimization. 184, 186, 189–197, 200–202, 210, 224

COTS Commercial Off-The-Shelf. 137, 152, 163

CPA Correlation Power Analysis. 22, 178, 182, 203

CR Coding Rate. 139, 142, 147, 149

CRC Cyclic Redundancy Check. 63, 64, 67, 141, 142

CSS Chirp Spread Spectrum. 138

CTS Clear To Send. 105, 106, 125

CV Computer Vision. 158

CVSS Common Vulnerability Scoring System. 4

DA Destination Address. 63, 73

DBPSK Differential Binary Phase Shift Keying. 59, 82

DC Direct Current. 140, 164

DEMA Differential Electromagnetic Analysis. 193

DES Data Encryption Standard. 21

DL Deep Learning. 27, 179, 183, 184, 202, 215

DOM Difference of Means. 16

DoS Denial of Service. 72

DPA Differential Power Analysis. 19, 21, 22

DSSS Direct-Sequence Spread Spectrum. 92

DTW Dynamic Time Warping. 19

EAP Extensible Authentication Protocol. 40, 41, 43, 46–48, 50, 52, 53, 211

ECDH Elliptic-curve Diffie–Hellman. 21

EM electromagnetic. 7, 8, 12, 13, 21, 29, 30, 32, 178–180, 182–184, 188, 189, 197–200, 202, 203, 210, 211, 224

FCS Frame Check Sequence. 68, 71, 74

FFT Fast Fourier Transform. 138–140, 146, 151, 168, 193–195, 199, 202, 203

FPGA Field-Programmable Gate Array. 52

GAS Generic Advertisement Service. 88, 107–110, 118, 120–122, 125, 131, 209

GE Guessing Entropy. 24, 25, 187

GI Guard Interval. 61

GPS Global Positioning System. 86

GSM Global System for Mobile Communications. 152

HD Hamming Distance. 182

HF High Frequency. 156

HN High Nibble. 142

HT High Throughput. 61, 62, 66

HW Hamming Weight. 180, 182, 183, 196, 197

ICBLBC Isolated Complementary Binary Linear Block Codes. 81

IDS Intrusion Detection System. 82

IE Information Element. 62, 66, 87, 92, 94, 98–101, 105, 111, 113, 114, 118, 119, 121, 123–125, 211

IEEE Institute of Electrical and Electronics Engineers. 40, 58, 59, 126

IoT Internet of Things. 1, 86, 136, 156, 178

ISK Inner Session Key. 51

LAN Local Area Network. 89

LDA Linear Discriminant Analysis. 24

LEAP Lightweight Extensible Authentication Protocol. 41, 42, 45–47, 49–53, 207, 208, 211

LF Location Fingerprinting. 86

LFSR Linear-Feedback Shift Register. 91, 141, 147

LLC Logical Link Control. 62

LN Low Nibble. 142

LPWAN Low-Power Wide-Area Network. 2, 3, 136, 156

LSB Least Significant Bit. 139, 140, 142

LTE Long Term Evolution. 152

M2M Machine to Machine. 1, 136

MAC Medium Access Control. viii, 5–8, 58–62, 64, 65, 68, 72, 73, 79, 81, 83, 85, 87–96, 101, 103, 105–108, 111–116, 118, 120–126, 128, 132, 141, 142, 156, 157, 208, 209, 211, 223

MCS Modulation and Coding Scheme. 60–62

MI Mutual Information. 14

MIMO Multiple Input, Multiple Output. 58

MITM Man-In-The-Middle. 40, 42, 46, 51, 52, 207

MK Master Key. 51

ML Machine Learning. 27–29, 179, 180, 183–188, 191, 199, 202, 203, 210, 211, 215

MLP Multilayer Perceptron. 30, 31, 33, 161, 168, 170, 172, 184, 190–197, 199–201, 203, 209

MPDU MAC Protocol Data Unit. 62–65, 68, 71, 75, 79, 81, 82

MS Monitoring Station. 86–90, 92, 94, 102–106, 109, 111–116, 118–122, 125, 127, 130, 208

MSDU MAC Service Data Unit. 62, 63, 70

MSK Master Session Key. 42, 44, 49–51

MTD Minimum Traces to Disclosure. 24

NAT Network Address Translation. 66

NIC Network Interface Controller. 60, 67, 80, 81, 92

NR New Radio. 2

OFDM Orthogonal Frequency-Division Multiplexing. 91, 126

OSI Open Systems Interconnection. 83, 208

OUI Organizationally Unique Identifier. 47, 93, 115

P2P Peer-To-Peer. 110

PAPR Peak-to-Average Power Ratio. 91

PCA Principal Component Analysis. 15, 16, 24

PDR Packet Delivery Ratio. 69, 77–79, 224

PEAP Protected Extensible Authentication Protocol. 41, 42, 44, 46–49, 51, 52, 207

PER Packet Error Rate. 149–152

PHY Physical. 5, 7, 8, 58–62, 64, 68, 87, 90–92, 125, 132, 136, 137, 140–143, 147, 151, 152, 156–158, 160, 161, 172, 174, 178, 208, 209, 211

PIP Packet-In-Packet. 59, 60, 65, 80, 208

PIWN Personally Identifying Wireless Network. 94, 106

PLCP Physical Layer Convergence Protocol. 59–61, 74, 75, 82, 90

PMK Pairwise Master Key. 44, 49

PNL Preferred Network List. 40, 105, 120, 121, 123

PoC Proof of Concept. 137, 207, 208

POI Point of Interest. 14–17, 23, 24, 35

PPDU PLCP Protocol Data Unit. 61

PRNG Pseudo-Random Number Generator. 91

PSD Power Spectral Density. 91

PSDU PLCP Service Data Unit. 60–62

QAM Quadrature Amplitude Modulation. 82

QoS Quality of Service. 63, 107, 108

RA Receiver Address. 63, 64, 72, 107

RADIUS Remote Authentication Dial-In User Service. 42, 50

RF Radio Frequency. 156, 157, 167, 174

RFID Radio Frequency Identification. 156, 173

RSN Robust Secure Network. 105

RSSI Received Signal Strength Indicator. 51

RTS Request To Send. 105, 106, 125

SA Source Address. 63

SCA Side-Channel Analysis. 11, 13–15, 17, 24, 34, 178, 180, 187, 199, 200, 202, 203, 214, 215

SDR Software Defined Radio. 7, 8, 13, 91, 137, 142–145, 148, 150, 152, 163, 168, 172, 179, 197–199, 203, 209, 210

SEMA Simple Electromagnetic Analysis. 21, 199

SF Spreading Factor. 138, 140, 142, 148, 149

SFO Sampling Frequency Offset. 144

SIMD Single Instruction, Multiple Data. 144

SK Session Key. 42

SNR Signal-to-Noise Ratio. 17, 18, 149, 150, 152, 174, 209, 224

SOSD Sum of Squared Pairwise Differences. 16

SOST Sum of Squared Pairwise t-Differences. 17

SPA Simple Power Analysis. 20, 21

SQL Structured Query Language. 80, 81

SSID Service Set Identifier. 40, 47, 72, 73, 94, 98, 99, 105, 106, 120, 121, 123–126

SSPN Subscription Service Provider Network. 108

STA station. 94, 98, 103, 104, 106–111, 116, 117, 119–121, 123–125

SVM Support Vector Machine. 91, 161, 163, 168, 172, 202, 209

TA Transmitter Address. 63, 72, 73, 106, 107

TA Template Attack. 22–24, 183, 185, 202, 203

TDLS Tunneled Direct-Link Setup. 107, 109–111, 119

TDOA Time Difference Of Arrival. 88

TK Tunnel Key. 51

TLS Transport Layer Security. 2, 42, 43, 46, 48, 50, 51, 207

TLV Type-Length-Value. 51, 94, 98

TSF Timing Synchronization Function. 94

TVLA Test Vector Leakage Assessment. 14, 17

UHF Ultra High Frequency. 173

USRP Universal Software Radio Peripheral. 197, 203

VHF Very High Frequency. 156

VOLK Vector Optimized Library of Kernels. 144

WAN Wide Area Network. 68

WEP Wired Equivalent Privacy. 42

WIDS Wireless Intrusion Detection System. 51, 52

WNM Wireless Network Management. 107, 110, 111, 119

WSN Wireless Sensor Network. 1, 136, 178

Complexity is the worst enemy of security.

Niels Ferguson, Bruce Schneier, Tadayoshi Kohno

1 Introduction

1.1 Motivation

Over the past decennia, the usage of wireless communication has become increasingly ingrained in our society and day-to-day lives. This is evidenced by recent market surveys, which indicate that the global wireless connectivity market is expected to continue to grow from an estimated $18.73 billion in 2016 to $34.71 billion by 2023 [254], while the total number of mobile users worldwide is forecast to reach 7.33 billion by 2023 [258]. One of the contributing factors to this growth is the rising popularity of using mobile phones to access the internet: statistics published by Eurostat indicate that between 2014 and 2018, the percentage of individuals in the European Union that used a mobile phone to access the internet has increased from 44% to 67% [84]. Another factor is the onset of the Internet of Things (IoT) trend, i.e. the provisioning of internet connectivity to everyday devices. This benefits both companies by allowing, e.g., remote updating and data mining, as well as customers through for example new features and enabling remote control. In this category we find devices ranging from wearables such as activity trackers and smart watches to "smart home" devices such as smart speakers, doorbells and thermostats. Finally, Wireless Sensor Networks (WSNs) and other Machine to Machine (M2M) applications have found their role

in optimizing various aspects of the automotive [64, 211], aviation [125, 287] and farming industries [147, 182].

A natural consequence of the continued proliferation of wirelessly connected devices is the emergence of a wide spectrum of novel use cases and hence, new and more complex (amendments to) wireless protocols had to be designed. While emergent protocols for consumer end devices remain mostly focused towards achieving higher data rates with technologies such as Wi-Fi 6 (802.11ax) [277] and 5G New Radio (NR) [1], other protocols are oriented more towards low power consumption and long-range communications. Amongst the protocols introduced specifically for the latter use case, which are coined under the umbrella term "Low-Power Wide-Area Network (LPWAN) protocols", we can find for example LoRa [156], Sigfox [241], LTE-M [2], Wi-Fi HaLow [278] and Weightless [274].

For whichever purpose a wireless protocol is designed, it remains crucial to ensure the security of the data it transports and to cater to the privacy of users. The rising popularity of mobile devices, and hence of wireless communication, results in more data being transmitted over the air. Factoring in the overall digitization of society, it is safe to assume that this data will also contain more *sensitive* information. For example, think of implanted medical devices [162] or wirelessly connected systems on an airplane [228]. Naturally, we don't want an adversary to be able to read medical data or manipulate messages sent to airplane systems; the outcome could be disastrous. The main security properties that are desired in a communication system can be summarized through the Confidentiality, Integrity, and Availability (CIA) triad. Here, "confidentiality" pertains to keeping the data secret, "integrity" is related to preventing modification and "availability" refers to making sure users can access their data at all times.

Outside of the CIA triad, another desirable property is ensuring the privacy of the user. While the term "security" refers to the technical means used to protect communication, "privacy" is a much broader concept that also concerns metadata of the communication, such as knowing who is communicating when, with whom, from what location, by which means and under what circumstances. In legal and societal contexts, privacy can furthermore relate to the nature of the communicated data itself, how this data is stored and whether and how it is shared with third parties. For example, a smartphone application can set up a secure Transport Layer Security (TLS) channel to a web server, but an adversary could still apply "traffic analysis" techniques to infer information about the user, hence compromising their privacy [77, 166, 190, 256].

1.2 Problem statement and research goals

To determine whether a wireless protocol invests sufficiently in a users' data security and privacy, it must be carefully scrutinized by security researchers. As the domain evolves, previously unknown attack surfaces or attack techniques may be discovered and precisely for this reason, the defenders (the protocol designers or implementers) are always at a disadvantage compared to the attackers (security researchers). Especially newer protocols, which have typically received only limited peer review from the information security community, are more likely to contain hidden flaws[1] (see for example [267]). Moreover, compared to wired networks, there are more challenges to be considered as adversaries are inherently more powerful: they can intercept, modify or jam any messages exchanged between the parties engaged in a wireless protocol. For LPWAN protocols, there is the additional challenge of providing security and privacy while keeping the power consumption low. Provided that power consumption is coupled with computational complexity, we logically arrive at the performance-security tradeoff, which is well-known in the information security domain. Since creating a computational advantage necessitates at least some penalty on the performance of a system, performance and security are fundamentally in conflict, and a good tradeoff that works well in practice must therefore be determined.

The question now remains of how to assess the level of security and privacy provided by a given protocol. In the field of cryptography, the security that a cryptographic primitive provides is approximated as a number of bits, which indicates the number of steps or work that an adversary has to do for an attack [85, p. 36]. For example, a secure n-bit cryptographic hash function is said to provide only $n/2$ bits of security, since an adversary would need to perform $2^{n/2}$ steps on average in order to perform a birthday attack. While the level of security that cryptographic primitives provide can be objectively measured by analyzing their mathematical properties, there is no standard objective measure for measuring the overall security and privacy of a system as a whole. An individual component of the system may be proven secure on itself, but turn out to be broken when used in conjunction with other components in a protocol. This may occur, for example, due to certain complex and unintended interactions that went unaccounted for in the protocol design or in the implementation of the protocol, ultimately leading to a *vulnerability*.

When unintended behavior of a protocol causes a vulnerability that is sufficiently

[1] Although it should be noted that by nature of the security cat-and-mouse game, even extensive peer review does not guarantee security under practical circumstances.

severe[2], it can be exploited by an adversary to mount an actual *attack* on a protocol. Examples of such attacks are spoofing, replay attacks, relay attacks, user tracking, amplification attacks, cryptographic attacks, side-channel attacks, etc. Each of these attacks is the consequence of design or implementation vulnerabilities, and affects the CIA properties of a system. A particular class of vulnerabilities that will be focused on in this thesis is the class of "*information leakage*" vulnerabilities, which have the potential to enable many of the aforementioned attacks. Information leakage refers to bits of information that are obtained and exploited by an adversary in order to gain some advantage, e.g., a computational advantage compared to performing a naive exhaustive key search (a cryptographic vulnerability) or being able to inadvertently uncover the identity of a communicating party (a privacy issue). We will furthermore differentiate between two types of information leakage: explicit and implicit information leakage.

The first type, **explicit information leakage**, is information leakage that is a direct consequence of flaws in a protocol's design or implementation. That is, given a specific input, the protocol behaves in a way that was not intended by the designer or programmer, thereby introducing an information leakage vulnerability. In Chapter 4, we will see an example of an explicit information leakage vulnerability that can be exploited to perform a relay attack on the PEAP protocol. Chapter 5 demonstrates how information can leak from higher to lower layers in the network stack due to a parsing vulnerability in 802.11. In Chapter 6 we will see an instance where explicit information leaks can lead to mass tracking of smartphone users.

The second type, **implicit information leakage**, is information leakage that manifests due to measurable *side effects* occurring during the execution of a protocol. In this case, the protocol itself behaves as intended by the programmer, but the software or hardware implementation unintentionally leaks information to an adversary. The term "implicit" should be understood as "always to be found in": the implicit leakage discussed in this thesis is inherently present, though whether it can be exploited depends on the measurement capabilities of the adversary. Intuitively, implicit information leakage can be seen as a superset of "side-channel" leakage that also includes non-cryptographic information leakage (for example, device fingerprints). The concept of side channels will be reviewed in Chapter 2. In Chapter 8 we will see an example of how implicit information leakage allows an attacker to uniquely identify a LoRa device based on its transmitted signal alone. Chapter 9 describes how electromagnetic radiation leaked by a CPU can be analyzed with machine learning methods in order to determine the key used

[2]The severity of a vulnerability can be measured using a standardized scoring system such as the Common Vulnerability Scoring System (CVSS) [86].

in a cryptographic algorithm.

Both of the previously discussed leakage types are the result of behavior by a protocol that was unintended by the designer or programmer. Although it is trivial to test how a protocol *should* behave, e.g., through unit tests, it is hard to test for how a protocol *could* behave. Doing so would require checking all possible combinations of configurations and inputs, which is infeasible for many modern protocols. This is precisely why it is important for the security community to look for vulnerabilities: to make sure a protocol behaves only as intended, and not as the protocol equivalent of a "weird machine": a program that shows unintended behavior when provided with unexpected inputs. Formal verification and tools such as fuzzers can help to automatically discover flaws in respectively protocol designs and implementations, but they require an a priori definition of what unintended behavior entails exactly. Similar to how a programmer can come up with a creative solution to make a program display some intended behavior, it requires a human mind and creativity to reveal unintended behavior.

In light of these observations, this thesis aims to achieve the following research goals:

RG1 (Identification) Perform a security and privacy analysis of state-of-the-art wireless protocol designs and implementations on the Physical (PHY) and MAC layers, with a focus on identifying information leakage in wireless protocols.

RG2 (Exploration) Improve existing techniques or find new ways to extract or exploit information leaks from a practical implementation of a wireless protocol in order to get a better understanding of an adversary's capabilities.

RG3 (Remediation) Provide users and developers of wireless protocols with datasets and tools to test for and ward against attacks that result from newly discovered information leakage or extraction techniques, as well as inform them about the risks of such attacks.

Throughout most of the thesis, these research goals were strived for in a sequential manner, i.e. by first identifying a vulnerability (**RG1**), exploring the options of the adversary (**RG2**) and finally by mitigating the vulnerability and suggesting improvements (**RG3**). An exception for some chapters is that **RG1** had already been performed in related work, in which case only **RG2** and **RG3** are considered. To avoid any confusion, each chapter will individually list its intended

research goals. Finally, the focus of these research goals is primarily on two popular wireless protocols, namely Wi-Fi (802.11) and LoRa. These protocols were focused on because they are commonly deployed in practice: the number of global Wi-Fi hotspots is estimated to reach 454 million by 2020 [59], and the number of global LoRa gateways is estimated to climb to 500,000 by 2020 [237]. Nevertheless, many principles from the results of this research can be applied to wireless protocols in general.

1.3 Contributions

Pursuant to the research goals listed in the previous section, this thesis brings the following main contributions:

C1 We show that MSCHAPv1 credentials can be converted to MSCHAPv2 credentials, leading to a vulnerability in devices that support both the LEAP and PEAP authentication protocols, which allows an adversary to bypass authentication. We also provide two open-source tools on Github: `scapyfakeap`, which aims to provide a means to more easily test for security vulnerabilities in 802.11, and `peapwn`, which allows to test for the presence of the LEAP vulnerability. Finally, we suggest 5 approaches that can be implemented to prevent exploitation of the vulnerability.

C2 We expose a vulnerability in the MAC-layer frame aggregation mechanism featured in 802.11n and following amendments, which allows an adversary to remotely inject frames into local, open Wi-Fi networks. A tool named `aggrinject` is made available on Github that allows to test for this vulnerability. Finally, we propose 6 countermeasures to protect against injection attacks.

C3 We introduce a metric based on information entropy to measure the suitability of MAC-layer Wi-Fi frame bits in constructing a unique fingerprint of the transmitting device. Based on this metric, we introduce the concepts of "variability" and "stability" of a fingerprint.

C4 We conduct a large-scale analysis of 200,394 `Probe Requests` that were captured in a realistic setting. We then show that, based on the metric from **C3**, a fingerprint can be constructed that is 80.0 to 67.6 percent unique for 50 to 100 observed devices and 33.0 to 15.1 percent unique for 1,000 to 10,000 observed devices. Moreover, we indicate how this technique can be used to track devices and defeat countermeasures such as MAC address randomization, and compare with existing techniques. Furthermore, we present 8 countermeasures to mitigate user tracking.

C5 We propose and evaluate a variety of techniques to instigate 802.11 frame transmissions from a device, for example by broadcasting `GAS Request` or `ADDBA Request` frames.

C6 We provide a first full description of the LoRa PHY layer by reverse-engineering a RN2486 LoRa hardware module.

C7 We design and develop the first fully-featured implementation of a LoRa demodulator for Software Defined Radio (SDR) using the GNU Radio framework, named `gr-lora`. The demodulator implements a novel algorithm for the LoRa PHY that allows to demodulate multiple channels without requiring retuning of the SDR center frequency, and is open sourced on Github.

C8 The framework from **C7** is used to build a PHY-layer fingerprinting technique for LoRa devices based on a combination of supervised classification and unsupervised clustering methods. Our technique is able to distinguish LoRa vendor models with 99%-100% accuracy and individual devices with 59%-99% accuracy. This work was performed in collaboration with dr. Eduard Marin from KU Leuven.

C9 We propose a novel methodology to discover and exploit implicit information leakages originating from electromagnetic (EM) radiation using machine learning methods. This methodology does not require prior alignment of EM traces and outperforms previous methods. We then evaluate the methodology on Advanced Encryption Standard (AES) traces included in the ASCAD dataset and a custom dataset.

C10 All datasets procured during this research have been made available to the public domain in an anonymized format on Zenodo[3]. Furthermore, the code for all tools developed to realize the above contributions has been open sourced on Github[4]. The locations and descriptions of these datasets and tools can be found in Appendix B.

[3]The used anonymization methods are described for each dataset at `https://zenodo.org`
[4]All code contributions can be found at `https://github.com/rpp0`

These contributions were presented in 1 journal publication and 7 conference publications as the principal author. A list of all publications is given in Appendix C. Additionally, in light of raising awareness to the public, some contributions were also presented via non-academic channels, e.g., at the FOSDEM 2018 and 2019 conferences and in the general press (see Appendix D).

1.4 Thesis structure

The remainder of this thesis is subdivided into four main parts, where each of the chapters can either be read sequentially or on its own. The first part, "Preliminaries", describes a number of concepts that serve as background information for the later chapters. In Chapter 2, we will see how side-channel attacks can be used to break implementations of cryptographic primitives, while also touching upon a number of EM measurement, signal processing, and analysis principles. The purpose of this chapter is to help contextualize the contributions listed in **C9**. Then, Chapter 3 reviews a number of machine learning related concepts that are referenced in both Chapter 8 and Chapter 9 of the thesis. Readers who are already familiar with these concepts can safely skip the preliminaries.

In the second part of this thesis we focus on explicit information leakage. Particularly, in accordance with the research goals described in Section 1.2, we address a number of protocol design issues and improvements for 802.11. Chapter 4 describes a vulnerability for LEAP and PEAP authentication in accordance with **C1**. A flaw in 802.11 MAC-layer frame aggregation (**C2**) is explored in Chapter 5. Contributions **C3** – **C5** are discussed in Chapter 6.

Next, the third part of this thesis centers on implicit information leakage. Chapter 7 is a chapter that describes the analysis, design and implementation of a LoRa demodulator for SDR, yielding **C6** and **C7**. In Chapter 8, we will see how we use this demodulator to detect subtle hardware differences between LoRa devices in order to fingerprint individual LoRa devices on the PHY layer (**C8**). Chapter 9 describes a novel methodology to improve CEMA attacks using machine learning, leading to **C9**.

Lastly, in the fourth and final part of the thesis we reflect upon the obtained results to formulate a conclusion and present an outlook on the challenges ahead, which could be considered in future research.

Part I

Preliminaries

Cryptography is typically bypassed, not penetrated.
Adi Shamir

2

Side-Channel Analysis

In present-day communication systems, the confidentiality and integrity of information is primarily ensured through the use of cryptographic algorithms. At their core, these algorithms rely on secret pieces of information, i.e. keys, known only to the communicating parties in order to obtain a *computational advantage* over the adversary. That is, given that the cryptographic algorithm is not theoretically broken, it should be computationally infeasible for an adversary to modify or eavesdrop on the communication without having access to the secret information. However, as demonstrated in numerous previous works (see [290] and the references therein), an adversary can infer secret information by statistically analyzing physical properties of the hardware implementation during the execution of a cipher. These physical properties, named "side channels", can thus unintentionally "leak" information to an adversary.

Side-Channel Analysis (SCA) refers to the practice of measuring and interpreting leakages originating from a side channel in order to perform an attack on the cryptosystem.

2.1 Types of side channels

The physical phenomena that are responsible for leaking key-dependent information can take on widely different forms and characteristics. In the current literature, various types of side channels have consequently been identified as sources of potential information leakage, including:

- Seismic side-channels: Analyzing vibrations to determine keypresses, for example using a laser [19] or gyroscope [177].
- Acoustic side-channels: Inferring keypresses from stereoscopic microphones in a phone [177] or extracting RSA keys by analyzing the sound generated by a device [95].
- EM side-channels: Information leaked via the electromagnetic radiation that is emitted by various electronic components [89, 93, 94, 97, 174, 175, 201, 204, 272].
- Temperature side-channels: Information leaking through temperature fluctuations of a device [39, 123].
- Power side-channels: Patterns in a device's power consumption [37, 111, 165, 169, 181, 186].

It should be noted that the above list is non-exhaustive; new side channels may be discovered in the future (e.g., side channels based on quantum physics). Furthermore, new ways of measuring existing side channels may also be discovered, revealing novel avenues of attack that might render existing side-channel countermeasures ineffective. In this thesis, we will specifically focus on the EM side channel.

2.2 EM side-channel leakage

During the execution of a cryptographic operation, conditional jumps based on the key bits or computational intermediates, gate switching, and timing differences between operations can affect the power consumption of a device [37, 111, 137, 174, 181]. Since the current flows within a device in turn produce EM radiation [8], it follows that the EM side-channel can leak key-dependent information

as well. In fact, it is argued by Agrawal et al. that the EM side channel is more powerful than the power side channel [8].

To capture EM signals, an adversary can place one or multiple near-field probes in close proximity to a location of interest, such as the CPU voltage regulator, power-supply pin, or the CPU itself. Leakages may also be present in the far field, in which case wideband antennas can be used instead [43]. Actual measurements of the EM signal can be performed using high-end devices such as oscilloscopes and spectrum analyzers, or low-end devices such as SDRs. Once the signals have been captured in digital format, the adversary can apply signal processing and SCA techniques in order to identify the presence of implicit information leakage.

For the remainder of this thesis, we will refer to a single capture of an EM signal as a *trace*, *signal* or *measurement*. These terms will be used interchangeably. A single trace x consists of a finite number n_x of real *samples* or *points*, i.e., $x \in \mathbb{R}^{n_x}$. A collection of multiple traces will be referred to as a *trace set*. Finally, a set of instructions executed to complete a certain (cryptographic) task of interest is denoted as an *operation*.

2.3 Leakage detection and POI selection

Once the adversary has obtained a set of EM leakage traces from the targeted device, the question remains whether the quality and quantity of the measurements is sufficient to perform a successful attack. Both of these aspects indeed greatly impact the success rate of a full key recovery, which complicates the comparison of different attack methodologies. Let us now focus on what the "quality" of a leakage trace entails exactly. In general, an ideal trace should exhibit the following properties:

- **High variance between operations**: Performing different operations on the targeted device should result in large differences between trace sets pertaining to these operations. Conversely, if two operations have the same observed leakage, they cannot be differentiated.
- **Low redundancy during operations**: Traces should contain only information that is relevant to the operation itself. Any auxiliary information that does not relate to the operation only increases the storage requirements and computational complexity of the attack.

These properties are closely related to each other: if we can for example identify

a number of points in a trace set that have high variance between operations, we could discard other low-variance points to reduce the redundancy. This is the precisely goal of Point of Interest (POI) selection, which is often performed as a preprocessing step before an actual attack. In the following sections, we will discuss a number of metrics for assessing the trace quality as well as their application to POI selection.

2.3.1 Mutual information

The Mutual Information (MI) is an information-theoretic metric that was first introduced in context of SCA by Standaert et al. in order to compare the side-channel leakage between different implementations [253]. Let $x_i \in \mathbb{R}^{n_m}$ be a vector containing the i^{th} leakage point from all n_m traces in $X \in \mathbb{R}^{n_m \times n_x}$ and let y_j be the corresponding vector of the j^{th} piece of key-based information (e.g. a bit or byte of the key) from all traces. The MI then gives the mutual dependence between the key information and leakage information as:

$$I(x_i; y_j) = \sum_i \sum_j p(x_i, y_j) \log_2 \left(\frac{p(x_i, y_j)}{p(x_i)p(y_j)} \right) \tag{2.1}$$

The MI thus provides us with a means to measure how much each leakage point changes if we change the key. For example, suppose an attacker is able to measure the exact value of a CPU register where a byte of the secret key is being stored. Since each change to this byte directly affects the measured leakage, the MI between these two variables will be 8. On the other hand, a leakage point that never changes for different key byte values will result in a MI of zero.

2.3.2 TVLA

Test Vector Leakage Assessment (TVLA) is a testing methodology for measuring side-channel resistance that was introduced in [106]. Multiple variants of the methodology exist, of which the non-specific fixed-vs-random variant is currently considered the most popular [252] and most powerful [22]. All variants use the statistical t-value as a basis for determining whether a device fails the test, with a threshold value of 4.5 [22, 106].

The non-specific, fixed-vs-random TVLA takes two sets of n_m measurement

traces of length n_x, $F \in \mathbb{R}^{n_m \times n_x}$ and $R \in \mathbb{R}^{n_m \times n_x}$, where F contains n_m traces of fixed-key and fixed-data operations, while R contains n_m traces of fixed-key and random-data operations. Then, Welch's t-test is performed in a point-wise manner on the two populations as: [106]

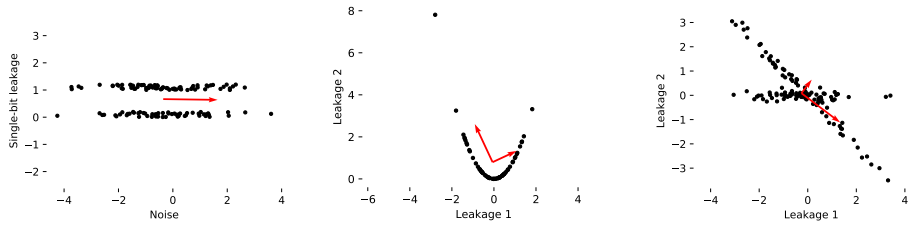
$$t = \frac{\overline{F} - \overline{R}}{\sqrt{\frac{s_R^2}{n_m} + \frac{s_F^2}{n_m}}} \tag{2.2}$$

where s_R and s_F denote the sample variances of respectively R and F, and $\overline{R}$ and $\overline{F}$ represent their point-wise means. This operation is repeated for another two sets of traces, and if both tests have $|t| > 4.5$ at the same point in time, the device is considered vulnerable [22].

2.3.3 Principal component analysis

Given a zero-mean data set $X \in \mathbb{R}^{n_m \times n_x}$ containing n_m traces of length n_x, Principal Component Analysis (PCA) performs an orthogonal change of basis such that the basis vectors point in the directions with the largest variance in the measurement space. More formally, PCA determines a linear orthogonal projection matrix P such that each i^{th} row $P^{(i)}$ is an eigenvector of $\frac{1}{n_x} X^T X$ and then performs this projection as $Y = PX^T$ [239]. The eigenvectors are in this case called the *principal components*, and they are ranked such that the first principal component corresponds to the largest variance, the second principal component to the second largest variance and so on.

In context of SCA, PCA dimensionality reduction has been utilized in previous works as a means to perform POI selection, e.g. in [32] and [11]. Indeed, the properties of POIs selected through PCA align well with our earlier-defined properties of an ideal trace: each point in the dimensionality-reduced trace has a high variance (indicated by the diagonal of $\sum$) and low redundancy (indicated by the covariance between pairs of leakage samples). However, it is important to note that we are mainly interested in points that have high variance *between operations*, rather than points with high *general* variance across a trace set. For instance, Figure 2.3.1(a) shows an example where a trace set contains one point of high-variance noise (the x-axis), and a second point with key-dependent leakage of a single bit (the y-axis). In this case, PCA will select the noisy point as the principal POI instead of the key-dependent leakage, since it has higher variance. As such, PCA is typically applied to the mean traces corresponding



(a) (b) (c)

Figure 2.3.1: Examples of two-dimensional datasets where a PCA would fail to recover the relation between two leakage points. The red arrows indicate the basis vectors of the projection.

to each operation [11]. Figure 2.3.1(b) and (c) show two other examples of a similar issue when PCA is applied to a dataset with respectively a nonlinear and non-orthogonal relation between data points.

A number of disadvantages of the technique come from its large computational time due to the calculation of the covariances between each pair of samples in the leakage trace [11, 208], its linear dimensionality reduction, and assumption of orthogonal bases [239].

2.3.4 Sum of differences

Another category of POI selection focuses on the sum of differences between traces. Here, various metrics could be used as a measure of difference.

One of the simplest POI selection methods is the Difference of Means (DOM), proposed by Chari et al. in [50]. Here, an average trace $\overline{x}^{(o)}$ is first calculated for each operation $o \in \{1, 2, \ldots, n_o\}$ with n_o the total number of operations. Then, points which exhibit large pairwise differences between $\overline{x}^{(1)} \cdots \overline{x}^{(n_o)}$ are selected as POIs. Other works have proposed to use the *sum* of all pairwise differences between operations and selects points among the highest peaks [208]. However, Gierlichs et al. noted that positive and negative differences between the averages may zeroize, which could result in information loss. They propose the Sum of

Squared Pairwise Differences (SOSD) as an improvement [100]:

$$\text{SOSD} = \sum_{i,j=1}^{n_o} (\overline{x}^{(i)} - \overline{x}^{(j)})^2, \text{ for } i \geq j \tag{2.3}$$

An issue with each of the above POI selection methods is that they only consider the mean of the signal. In case information is contained within the variance of the operations as well, a metric such as the Sum of Squared Pairwise t-Differences (SOST) can be used instead, which is defined as [100]:

$$\text{SOST} = \sum_{i,j=1}^{n_o} \left(\frac{\overline{x}^{(i)} - \overline{x}^{(j)}}{\sqrt{\frac{\sigma_i^2}{n_i} + \frac{\sigma_j^2}{n_j}}} \right)^2, \text{ for } i \geq j \tag{2.4}$$

Note that the SOST uses the t-value as its metric, similar to TVLA (see Section 2.3.2). Again, the highest values can be chosen as POIs [100, 121].

2.4 Signal alignment

For many operations pertaining to SCA, it is desired that collected traces be aligned in the time domain. For instance, when averaging a set of n_m traces of the same operation $\{x^{(1)}, x^{(2)}, \ldots, x^{(n_m)}\}$ in order to improve the Signal-to-Noise Ratio (SNR) of the operation's leakage, it is essential that $\forall j \in \{1, 2, ...n_x\}$ with n_x the number of samples per trace, the samples $x_j^{(1,2,...n_m)}$ correspond to the same leakage event (e.g., a leaking instruction). If the signals are not aligned, leakage from different parts of the operation will be added together, resulting in noise. In this section, we will briefly review a number of techniques for aligning signals.

2.4.1 Trigger signals

Trigger signals are commonly used as a coarse method for trace alignment. Here, the adversary is assumed to be in control of the device under attack and able to generate a trigger signal to mark the start and/or end of a cryptographic

operation of interest, e.g., by setting an IO pin [93, 94, 133, 174, 181]. Depending on the timing accuracy and wiring delays of the trigger, additional alignment techniques may subsequently be used to achieve a more accurate alignment [93, 174].

2.4.2 Cross-correlation

A commonly-used method to determine the relative lag between two signals $x^{(i)}, x^{(j)} \in \mathbb{R}^{n_x}$ with $i, j \in \{1, 2, \ldots, n_m\}$ and $i \neq j$ is by calculating the cross-correlation, which is defined for real signals as:

$$(x^{(i)} \star x^{(j)})_k = \sum_{m=-\infty}^{\infty} x_m^{(i)} x_{m+k}^{(j)} \tag{2.5}$$

The resulting cross-correlation signal will be high when $x^{(i)}$ and $x^{(j)}$ overlap. Hence, the index of the maximal value of this signal yields the point of maximal overlap, which is the lag between the two signals. In case more than two signals are to be aligned, one trace can be chosen as a "reference trace" to which all other signals are synchronized [43, 96, 174, 181, 265].

Note that when using this methodology, it is crucial that the reference trace is not distorted and has a high SNR. This can be ensured by discarding a bad reference trace and selecting a new one [96], or by using the mean of all currently aligned traces as the reference trace [43].

Finally, some works have applied cross-correlation as a means to deal with drift between traces. First, a set of time segments of the reference trace is chosen instead of the full trace. Then, assuming that the drift within the segments themselves is negligible, multiple cross-correlations with each of the time segments can be periodically performed to resynchronize the full trace [96, 181].

2.4.3 Autocorrelation

The autocorrelation of two real signals is defined identically to Equation 2.5, except that in this case $i = j$. As such, the autocorrelation determines the relative lag between a signal and itself, which means it can be used to detect repeating patterns within a signal. If a trace for example contains multiple identical cryp-

tographic operations, occurrences of each of these operations can be detected by observing peaks in the autocorrelation signal. It may therefore serve as a less intrusive method of segmenting traces compared to manually providing a trigger signal.

2.4.4 Dynamic time warping

Dynamic Time Warping (DTW) was originally introduced by Sakoe et al. as a technique for aligning speech signals [225]. A faster implementation, FastDTW, was proposed by Salvador et al. in [226], and later used in context of Differential Power Analysis (DPA) by Van Woudenberg et al., where it is referred to as "elastic alignment" [265].

Given two equal-length traces $x = \{1, 2, \dots i, \dots, T\}$ and $y = \{1, 2, \dots j, \dots, T\}$, the DTW algorithm constructs a "warp path" $F = \{c(1), c(2), \dots, c(k), \dots c(K)\}$ where K is the warp path length with $T \leq K < 2T$. A node $c(k)$ of a warp path is defined as the function $c(k) = (i(k), j(k))$, which indicates that the $i(k)^{\text{th}}$ point of x is warped to the $j(k)^{\text{th}}$ point in of y [225]. The warp path is furthermore constrained in a number of ways [225, 265]:

- *Monotonic conditions*: $i(k-1) \leq i(k)$ and $j(k-1) \leq j(k)$.
- *Continuity conditions*: $i(k) - i(k-1) \leq 1$ and $j(k) - j(k-1) \leq 1$.
- *Boundary conditions*: The warp path must start at $(1, 1)$ and end at (T, T).

The original paper proposed two possible additional constraints [225]:

- *Adjustment window condition*: A point can only be warped by a maximum distance of r, i.e., $|i(k) - j(k)| \leq r$.
- *Slope constraint condition*: The warp path is constrained by $P = n/m$, the number of diagonal steps n that must be taken per m vertical or horizontal steps.

To find the most optimal warp path, a cost matrix $d(i, j) = ||x_i - y_j||$ is first calculated. This matrix indicates the cost of a potential warp path node as the

Euclidean distance between the points x_i and y_j. The minimum-cost warp path distance is then determined by calculating:

$$D(x, y) = \min_{F} \frac{1}{N} \left[\sum_{k=1}^{K} d(c(k)) \cdot w(k) \right] \tag{2.6}$$

which can be obtained by starting from (T, T) and performing a greedy search towards $(1, 1)$. In Equation 2.6, $w(k)$ is a weighting function and $1/N$ is a normalization. In context of elastic alignment, N is chosen to be equal to $2T$ and $w(k)$ indicates the number of steps made in each dimension of the matrix d [265].

Finally, once the minimum-distance warp path F has been obtained, the principle of elastic alignment is to use this path to compress or stretch samples from y such that it is aligned with x [265].

$$\dot{x}_i = x_i \tag{2.7}$$

$$\dot{y}_j = \frac{1}{|\{k|i(k) = j\}|} \sum_{i(k)=j} y_{j(k)} \tag{2.8}$$

If the trace set contains more than two traces, one could set x to a fixed (aggregate) reference trace and perform elastic alignment with all remaining traces.

2.5 Attack methodologies

2.5.1 Simple Power Analysis

In a Simple Power Analysis (SPA), the adversary captures a single or aggregate power consumption trace of a cryptographic operation, and then directly interprets the patterns present in this trace to identify key-dependent leakage [137, 160, 174, 286]. The key-dependent leakage must therefore manifest as a discernible feature within a single (aggregate) trace. If the leakage is masked by noise such as measurement noise, noise from other operations or SCA countermeasures, a SPA is not feasible. In this case, techniques that are based on statistical measures, which involve multiple traces, could be used instead.

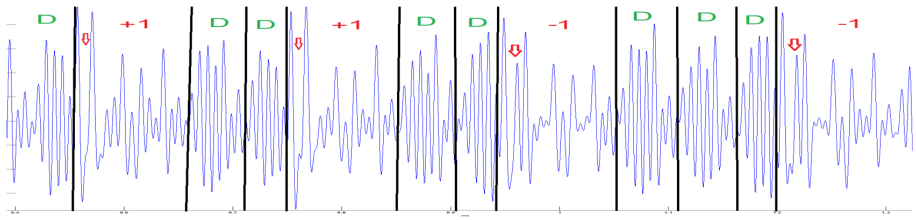


Figure 2.5.1: Figure from the work of Genkin et al. [98], showing an aggregate EM trace of GnuPG's ECDH decryption. The double (D) and add (+1/-1) operations can be distinguished with SEMA.

When a SPA relies on EM measurements rather than power consumption measurements, the analysis is referred to as Simple Electromagnetic Analysis (SEMA) [8, 204, 208, 224]. Although SPA and SEMA are conceptually identical, Agrawal et al. show that, for some instructions, the EM side channel can leak more information than the power side channel [8]. Figure 2.5.1 shows an example where a SEMA is performed on an implementation of Elliptic-curve Diffie–Hellman (ECDH). Note that the difference between double and add operations during the ECDH decryption can be visually observed from the trace. This information can subsequently be used to reveal the private key.

2.5.2 Differential Power Analysis

Classic DPA was introduced together with SPA by Kocher et al. in [137], where it was used to perform a power side-channel attack on Data Encryption Standard (DES). Similar to the case of SEMA, DPA is commonly called DEMA when applied to the EM side channel. In DPA, the adversary first captures n_m traces of the power consumption during a cryptographic operation with random-plaintext input, each consisting of n_x samples [169]. These can be arranged in a matrix $X \in \mathbb{R}^{n_m \times n_x}$. Next, the traces are partitioned according to a hypothesis of the leakage at any intermediate state of the cipher where this leakage depends on the key and the random input. For example, suppose that we have some leakage of a single bit of intermediate state $\mathcal{L}$ at time t that is a function of the trace's plaintext bit p_x and the true key bit k such that $x_t = \mathcal{L}(p_x, k)$. The adversary can then partition the traces into two sets:

$$S_0 = \{x | x \in X, \mathcal{L}(p_x, k_g) = 0\} \tag{2.9}$$
$$S_1 = \{x | x \in X, \mathcal{L}(p_x, k_g) = 1\} \tag{2.10}$$

where k_g is a guess for the value of the key bit k. The main observation of DPA is that if we make a correct guess for k_g, then the power consumptions at x_t will be correctly partitioned into S_0 and S_1, whereas if we make an incorrect guess, the partitioning will be random. If we calculate the differential trace T as:

$$T = \overline{S}_0 - \overline{S}_1 \tag{2.11}$$

where $\overline{S}_0$ and $\overline{S}_1$ are the mean traces of S_0 and S_1 respectively, then a peak should be visible at T_t when $k_g = k$, whereas for other guesses $T_t \approx 0$. In practice however, there may be leakage correlations with the true key even if k_g is an incorrect guess, leading to "ghost peaks" [37, 111]. Brier et al. introduced Correlation Power Analysis (CPA) as a methodology to mitigate some of the shortcomings of DPA, which will be discussed in Chapter 9 [37].

2.5.3 Template attacks

Classical Template Attacks (TAs) were introduced by Chari et al. in their seminal paper in 2003 [50] and consist of two stages: a *profiling* stage and an *attack* stage. In the profiling stage, the adversary is assumed to be in control of a device that is identical to the device under attack. This device is used to characterize both the leakage and noise under different operations using statistical models. Such characterizations are called *templates* of the operations. Once a template has been obtained for each operation, the adversary enters the attack stage and measures the leakage and noise of the device under attack. These measurements are then matched to the most likely template in order to determine exactly which operation was performed or to reduce the set of possibly performed operations.

Although the precise probability distribution of the leakage and noise is generally unknown, it can be approximated by the multivariate Gaussian model [11, 50, 55, 208]. To build templates under this assumption, the adversary performs the following steps:

1. The adversary captures a set of leakage traces $X_o \in \mathbb{R}^{m \times n}$, where m is the

number of traces, n is the length of each trace, and o is the operation being performed by the experimental device. We will refer to the i^{th} trace in X_o as $x^{(o,i)}$, where $x \in \mathbb{R}^n, i \in \{1, 2, \ldots m\}$.

2. For each operation, an average trace is calculated as:

$$\overline{x}^{(o)} = \frac{1}{m} \sum_{i=1}^{m} x^{(o,i)} \tag{2.12}$$

Under the multivariate Gaussian leakage model, we assume that this average trace corresponds to the leakage signal for operation o. Note that we also implicitly assume perfect alignment of the measurements.

3. Once the leakage signal has been obtained, the noise signals η are calculated by subtracting the leakage signal from each measurement:

$$\eta^{(o,i)} = x^{(o,i)} - \overline{x}^{(o)} \tag{2.13}$$

4. Finally, in order to model the noise, the covariance is computed between all pairs of components of the noise vectors, yielding the covariance matrix $\Sigma_{uv}^{(o)}$, where $u, v \in \eta^{(o,i)}$:

$$\Sigma_{uv}^{(o)} = \text{cov}(\eta_u^{(o,i)}, \eta_v^{(o,i)}) \tag{2.14}$$

The template for operation o can now be constructed as a tuple $(\overline{x}^{(o)}, \Sigma_{uv}^{(o)})$.

Given the set of templates, an adversary can now measure the leakage x from the device under attack and calculate the most likely template using the multivariate Gaussian probability density function:

$$p(x|\overline{x}^{(o)}, \Sigma^{(o)}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma^{(o)}|}} \exp\left(-\frac{1}{2}(x - \overline{x}^{(o)})^T {\Sigma^{(o)}}^{-1} (x - \overline{x}^{(o)})\right) \tag{2.15}$$

From the above description, it is clear that an important aspect of TAs is the selection of POIs. Without prior POI selection, the number of elements in the covariance matrix $\sum$ increases quadratically with the size of the leakage trace, which results in an increased computational load and storage requirements. The original paper by Chari et al. recommends to select POIs that have large pairwise differences between the average signals $\overline{x}^{(o)}$ [50] (see Section 2.3.4). However, in

the literature, several other POI selection methods have also been applied in context of TAs, including the sum-of-differences method [208], PCA [55, 208], or Linear Discriminant Analysis (LDA) [55].

2.6 Comparison of attacks

While the metrics and methodologies discussed in the previous section are useful for measuring information leakage, we also need a metric to compare the effectiveness of attacks that exploit this leakage. An intuitive approach is to simply count the number of traces required for a successful attack, known as the Minimum Traces to Disclosure (MTD) [246]. Here, a successful attack is defined as one where the adversary is able to fully recover the secret key. In practice however, it may occur that the adversary is able to obtain a small set of candidate keys using much less traces than the MTD (e.g., the attack may yield 3 candidate keys of which one is the true key). Two commonly-used metrics that allow for more flexibility regarding the definition of a successful attack are the o^{th}-order success rate and Guessing Entropy (GE), both of which were first formalized for SCA by Standaert et al. in [253].

2.6.1 Success rate

Suppose we have a part of the key, $k_s \in \mathcal{S}$, that we wish to recover using a side-channel attack, and an arbitrary[1] score vector $d = \{\text{score}(s) : s \in \mathcal{S}\}$, sorted from highest to lowest score such that $\forall i < j, d_i \geq d_j$. A successful side-channel recovery of order o is then defined as [253]:

$$\mathbf{Exp}^o(d) = \begin{cases} 1, & \text{if score}(k_s) \in [d_1, \ldots, d_o] \\ 0, & \text{otherwise} \end{cases} \tag{2.16}$$

Here, it is important to note that d depends on the algorithm under attack, its leakage function, and the adversary's capability of measuring this leakage. This assumption allows us to use a more simplified notation compared to [253]. The

[1] One could for example use the correlation with the true key k_s.

o^{th}-order success rate for an attack against k_s is now defined as:

$$\text{SR}^o_{k_s}(r_\tau, r_m, n_m) = p(\mathbf{Exp}^o(d) = 1) \tag{2.17}$$

where r_τ, r_m, and n_m are respectively the time complexity, memory complexity, and number of traces. Hence, the o^{th}-order success rate of an attack on k_s, given the computational limitations and number of measurements made by the adversary, is equal to the probability that the score for k_s is ranked amongst the top o scores in d.

2.6.2 Guessing entropy

If we modify $\mathbf{Exp}^{(o)}(d)$ such that it returns the *index* of the score of k_s, rather than a binary value that indicates its presence in the top o scores, we obtain the "rank" of k_s [253]:

$$\text{rank}(d) = \{i - 1 | d_i = \text{score}(k_s)\} \tag{2.18}$$

By convention, we choose 0 as the lowest possible rank (corresponding to the highest score in d). The GE is now defined as the expected number of guesses. If we use the optimal strategy to guess for keys with the highest score in d first, the expected number of guesses is equal to the expected value of $\text{rank}(d) + 1$:

$$\text{GE}_{k_s}(r_\tau, r_m, n_m) = \text{E}[\text{rank}(d) + 1] \tag{2.19}$$

All models are wrong, but some are useful.

George Box

3

Machine Learning and Deep Learning

In the domains of Artificial Intelligence (AI) and computer vision, techniques such as Machine Learning (ML) and Deep Learning (DL) have been shown to outperform classical approaches for numerous applications, including image classification [143], style transfer [92], face recognition [232], and image generation [132]. To perform such tasks, ML and DL algorithms learn an abstract representation of the input data by optimizing the internal parameters of a model based on a given objective function [151]. Hence, ML and DL algorithms try to solve *optimization problems*, e.g. for face recognition we might ask the question: "Which selection of facial features results in the most optimal recognition of a person's face?", whereas for image generation we may ask: "Which transformations on a random input generate an image that is indistinguishable from a non-generated image?".

Similar optimization problems can be found in the field of information security and privacy. For instance, the problem of determining whether an email contains spam can be likened to the problem of detecting whether a particular image contains a car or not. Both are *classification* problems; the only difference is in the type of input being provided, i.e. letters for an email and pixels for an image. Due to this similarity and the promising results obtained in computer vision, ML and DL techniques have also been examined in context of security-related topics

such as spam detection, intrusion detection, vulnerability discovery, forensics, fingerprinting and side-channel analysis [192, 219, 221, 250]. In Chapter 8 and Chapter 9, we will discuss the application of ML to respectively fingerprinting and side-channel analysis. This chapter introduces the used notational convention and revisits several concepts of ML that are necessary for understanding the remainder of the thesis. For a more complete overview of the current state of the art, see the works of Le Cun et al. [151], Shrestha et al. [240], and the references therein.

3.1 Notation and terminology

The ML-related notation used in this thesis is based on the Stanford notation (see for example [179] and [149]), with the exception that input examples are stored row-wise instead of column-wise. This avoids confusion with the code implementations of the presented work, where the Keras deep learning framework [53] is used. At the time of writing, Keras assumes a row-wise storage of input examples by default [54].

An *input example* is a vector of real values or *features* that is provided as an input to a ML algorithm. Depending on during which phase of the algorithm they are used (see Section 3.2), we may refer to them as either *training examples*, *cross-validation examples* or *test examples*. Formally, we denote an input example with n_x features as a tensor

$$x = \{x_1, x_2, \dots x_{n_x}\}, x \in \mathbb{R}^{n_x} \tag{3.1}$$

A *dataset* of n_m input examples can be then represented as a matrix X, where each example is stored in a row-wise manner:

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_{n_x}^{(1)} \\ x_1^{(2)} & \ddots & \ddots & x_{n_x}^{(2)} \\ \vdots & \ddots & \ddots & \vdots \\ x_1^{(n_m)} & x_2^{(n_m)} & \dots & x_{n_x}^{(n_m)} \end{bmatrix} \tag{3.2}$$

Note that individual training examples are referenced using a superscript index between parentheses, whereas a subscript indicates the index of a particular fea-

ture. For example, $x_2^{(4)}$ represents the 2^{nd} feature of the 4^{th} training example. If we only consider a subset of $n_b < n_m$ input examples of the total dataset in X, X is called a *batch* of input examples.

3.2 Supervised learning

Supervised learning is currently one of the most popular ML approaches for performing classification tasks. In this approach, a ML algorithm takes as input a tuple of tensors (x, y), where $x \in \mathbb{R}^{n_x}$ is an input example as before and $y \in \mathbb{R}^{n_y}$ is called a *label*. A label is a known numerical representation for a certain *class* that the ML algorithm must classify. For example, suppose that we would like to train a binary classifier that, given an EM trace, tries to determine whether the EM trace contains an AES encryption operation or not. In this case, we have two classes ($n_c = 2$), namely (i) the trace contains an AES encryption and (ii) the trace contains no AES encryption. The labels of these classes can be encoded as a numerical representation using *one-hot encoding*, where $y \in \mathbb{R}^{n_c}$ is a tensor of length n_c having the index of the corresponding class set to one and the remaining indices set to zero:

$$y^{(i)} = \begin{cases} \{1.0, 0.0\}, & \text{if AES in trace (class 0)} \\ \{0.0, 1.0\}, & \text{otherwise (class 1)} \end{cases} \tag{3.3}$$

During the training phase, the goal of the ML algorithm is then to learn a mapping from the training example's features $x^{(i)}$ to its corresponding label $y^{(i)}$. This is achieved by applying a sequence of parameterized non-linear functions to each input feature. In this thesis, we focus particularly on Artificial Neural Networks (ANNs), which among others use *perceptrons*, *convolution filters* and *pooling* operations to map features to labels. The following section will detail these building blocks and show how they are used in current ANN architectures to perform classification tasks.

3.3 Neural network architectures

3.3.1 Multi-Layer Perceptron

The architecture of Multilayer Perceptrons (MLPs) consists of layers of interconnected processing units, called perceptrons or neurons. Such layers are also called "fully connected" layers. Each neuron in a layer has an associated set of input values, weights, a bias term, and activation function. The output of a single neuron is determined by:

$$a_1 = g(xw + b_1) \tag{3.4}$$

In this equation, x is a tensor of input features (e.g. samples of an EM trace), w is the tensor of weights associated with each input feature, b_1 is the bias term, and $g(x)$ is the activation function. Neurons can be stacked together to form layers, such that the output becomes:

$$A^{[l]} = g(A^{[l-1]}W^{[l]} + b^{[l]}) \tag{3.5}$$

$$A^{[0]} = X \tag{3.6}$$

$$A^{[n_l]} = \hat{Y} \tag{3.7}$$

where $l \in \{0, 1, \ldots n_l\}$ indicates the layer index[1]. Note that we define the first activations $A^{[0]}$ as simply the input features X, and the final activations as the output predictions of the neural network, denoted as $\hat{Y}$.

3.3.2 Convolutional Neural Network

In Convolutional Neural Networks (CNNs), two additional types of layers are introduced: convolutional layers and pooling layers [159]. Here, the convolutional layer is a layer that performs a series of convolution operations on its inputs. The weights of the convolution kernels or "filters" are learned by the optimizer

[1]The index starts from 0 because by convention, the input layer is not counted as an actual layer.

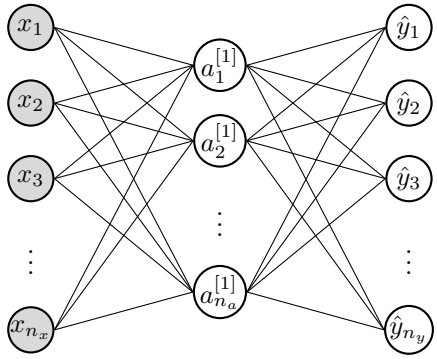


Figure 3.3.1: Example of a simple fully connected two-layer perceptron neural network architecture with n_x input features, n_a hidden layer nodes and n_y outputs.

algorithm. These filters act as feature detectors for the next layers that can be useful across the entire input [150]. At the same time, less weights need to be trained since the size of the filters is typically smaller than the number of input features itself.

Convolutional layers are usually followed by pooling layers, which reduce the resolution of the inputs by performing a local averaging (average pooling) or local maximum (max pooling) and a subsampling operation [150]. This operation makes CNNs more robust to small input shifts and deformations compared to MLPs [41, 150]. Figure 3.3.2 shows an example of a 1D CNN with multiple convolutional and pooling layers. Deeper layers are capable of detecting more complex features of the input due to their larger receptive field. At the end of the CNN, a series of fully connected layers follows, which uses the features detected by the convolution filters to determine the output of the model.

3.4 Optimization of neural networks

The values of the parameters of a neural network architecture are determined by optimizing an *objective function* with respect to the labels given during the training phase. In the testing phase, a set of inputs with unknown classes can then be provided to the trained model, yielding a prediction $\hat{y}^{(i)}$ of which class they belong to. The question now remains of how this optimization is performed.

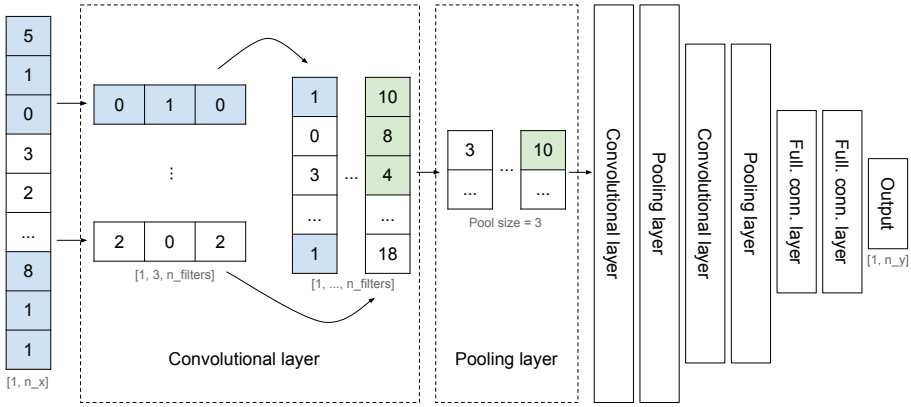


Figure 3.3.2: Example of a simple 1D convolutional neural network architecture, consisting of 3 convolutional layers, 3 pooling layers, 2 fully connected layers and an output layer.

3.4.1 Loss and cost functions

Before the parameters of a neural network can be optimized, a function $\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$ must be defined that measures, for the i^{th} training example, the fitness of a predicted label $\hat{y}^{(i)}$ compared to the known label $y^{(i)}$ during training. Returning to our example of classifying the presence of an AES encryption in an EM trace, the one-hot encoded labels can be interpreted as a probability distribution. Hence, we could use the cross-entropy loss as a similarity metric as follows:

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} log(\hat{y}^{(i)}) + (1 - y^{(i)}) log(1 - \hat{y}^{(i)})) \tag{3.8}$$

Since we are interested in minimizing the loss for our entire dataset, we can define the *cost function* as the average loss over all training examples.

$$J(\hat{y}, y) = \frac{1}{n_m} \sum_{i=1}^{n_m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \tag{3.9}$$

Note that if the predicted labels are equal to the true labels, the cost function J will be zero. It is therefore the optimizer's task to tune the model parameters such that the cost function is close to zero.

3.4.2 Parameter tuning

To produce an output prediction $\hat{Y}$ given X that results in a low cost, the parameters of the ANN must be tweaked during the training process such that $\hat{Y} \approx Y$. In order to achieve a good approximation of Y, the parameters are first initialized randomly[2]. This is to prevent the network from learning the same function for all nodes in a layer, i.e., to "break symmetry". Then, the following steps are generally repeatedly performed in order to train the network:

- Perform forward propagation to obtain $\hat{Y}$.
- Compute the cost function with respect to Y.
- Determine the gradient for the parameters with respect to the loss function using back propagation.
- Update the model parameters based on the gradient.

The back propagation algorithm works by calculating the partial derivatives of the model parameters with respect to the loss function, using the chain rule. Suppose for example that we are using the binary cross-entropy loss (see Equation 3.8) and we have the following single-layer MLP architecture:

$$\hat{Y} = \sigma(Z^{[1]}) \tag{3.10}$$

$$Z^{[1]} = XW^{[1]} + b^{[1]} \tag{3.11}$$

where σ is the sigmoid activation function. Then for one training example x we have:

$$\hat{y} = \sigma(xW^{[1]} + b^{[1]}) \tag{3.12}$$

The back propagation algorithm proceeds as follows:

[2]A popular initialization method at the time of writing is called "Glorot initialization".

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \hat{y}} &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} = \frac{\hat{y} - y}{\hat{y} - \hat{y}^2} \\
\frac{\partial \hat{y}}{\partial z^{[1]}} &= \frac{\partial \sigma(z^{[1]})}{\partial z^{[1]}} = \sigma(z^{[1]})(1 - \sigma(z^{[1]})) \\
\frac{\partial \mathcal{L}}{\partial z^{[1]}} &= \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{[1]}} \\
\frac{\partial z^{[1]}}{\partial W^{[1]}} &= \frac{\partial (x W^{[1]} + b^{[1]})}{\partial W^{[1]}} = x \\
\frac{\partial \mathcal{L}}{\partial W^{[1]}} &= \frac{\partial \mathcal{L}}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial W^{[1]}} \\
\frac{\partial \mathcal{L}}{\partial b^{[1]}} &= \frac{\partial \mathcal{L}}{\partial z^{[1]}}
\end{aligned}$$

Finally, the parameters can be updated, scaled with a given learning rate α as:

$$\begin{aligned}
W^{[1]} &= W^{[1]} - \alpha \frac{\partial \mathcal{L}}{\partial W^{[1]}} \\
b^{[1]} &= b^{[1]} - \alpha \frac{\partial \mathcal{L}}{\partial b^{[1]}}
\end{aligned}$$

3.4.3 Saliency maps

In context of computer vision, Simonyan et al. have demonstrated how saliency maps can be used as a means for visualizing the pixels of a given image that contribute the most (either positively or negatively) to a trained CNN's classification score S_c for a certain class. For an $m \times n$ grayscale image $x \in \mathbb{R}^{+m \times n}$, a saliency map M can be obtained by calculating the element-wise absolute values of the partial derivative of S_c with respect to x [244]:

$$M = \text{abs}\left(\frac{\partial S_c}{\partial x}\right) \tag{3.13}$$

The same methodology can be applied in context of SCA in order to identify

samples that are important in defining the score for a certain hypothesis key. Such points may then be used as POIs, as demonstrated in recent related works [119, 167].

Part II

Explicit Information Leakage in Wireless Communication

The enemy knows the system.

Claude Shannon

4

Exploiting WPA2-Enterprise Vendor Implementation Weaknesses through Challenge Response Oracles

4.1 Introduction

In wireless networks that implement Wi-Fi, the necessity to warrant the properties of the CIA triad that we discussed in the introduction of this thesis is perhaps even more important for enterprises, where the data that is being transmitted over the air may contain highly sensitive information such as trade secrets, medical records, or large quantities of personally identifying information. To ensure the integrity and confidentiality of this data, the 802.11 standard provides a number of security protocol variants that can be configured by network administrators.

A first variant named "WPA2-PSK", which is defined in the 802.11i amendment, is the most common choice for home users. Here, the network administrator configures a single passphrase to be used for authenticating to the Access Point (AP). This passphrase is shared with all users that require access to the network. For enterprises, such an approach would be infeasible since different users may require different access rights on the network, access may need to be revoked to former employees or the passphrase may unintentionally leak to unauthorized third parties. Therefore, the standard defines another variant that is more oriented towards enterprises, named "WPA2-Enterprise". Here, the network administrator can provide each user with their own username and password and manage these credentials from a central authentication server.

An important consideration to make, especially when the enterprise permits a Bring Your Own Device (BYOD) policy, is that most Wi-Fi-enabled devices on the market automatically join the nearest wireless network in their Preferred Network List (PNL) by default. This is convenient for the user, but it also allows an adversary to trivially perform an "Evil Twin" attack [180]. In this attack, the adversary sets up a rogue AP with a configuration identical to the targeted network (the same Service Set Identifier (SSID), capabilities and cryptographic suites) in order to trick users into connecting to their rogue AP and subsequently become a Man-In-The-Middle (MITM). To mitigate this issue, the authenticity of the AP can be verified by the device through an authentication protocol. Note that this verification transparently happens in the background, so the user fully relies on the used authentication protocol for its security. In context of WPA2-Enterprise networks, the Institute of Electrical and Electronics Engineers (IEEE) 802.1X standard specifies that the Extensible Authentication Protocol (EAP) protocol should be used for this purpose. This authentication protocol is "extensible" in the sense that it implements a wide variety of different authentication procedures, called EAP methods.

Though EAP methods are well-defined in their corresponding standards, a correct protocol implementation is the responsibility of the device vendor. Unsurprisingly, there are subtle differences between various vendor implementations. In this chapter, we will show that some of these differences can contribute to significant explicit information leakage vulnerabilities (**RG1**). We focus our analysis of EAP methods mainly on the Protected Extensible Authentication Protocol (PEAP) method because it is popular, widely supported and considered secure. We examined the PEAP implementation of some of the most popular operating systems used today, including Windows, Mac OS X, Android and iOS [113]. Our main contribution is a practical attack on Apple devices that allows an adversary to bypass authentication (**C1**). The vulnerability causing this issue was reported to Apple on February 5, 2014.

The remainder of this chapter is structured as follows. In Section 4.2, we give a theoretical description of the Lightweight Extensible Authentication Protocol (LEAP) and PEAP authentication protocols and highlight how the combined usage of these protocols can introduce vulnerabilities. Section 4.3 then illustrates a practical implementation of an attack that exploits this vulnerability. Additionally, we assess which devices were vulnerable at the time of writing (**RG2**). Then, we discuss a number of mitigation strategies in Section 4.4 (**RG3**), followed by an overview of related works and the conclusions of this chapter in Sections 4.5 and 4.6.

4.2 LEAP and PEAP vulnerabilities

Before we discuss a practical implementation of our attack in Section 4.3, let us first examine how credentials are exchanged in two EAP methods: LEAP and PEAP. The former, LEAP, is a proprietary EAP method developed by Cisco which uses the MSCHAPv1 algorithm to authenticate users. The three entities participating in the authentication are the Supplicant, the Authenticator, and the Authentication Server (AS). For simplicity, assume that Authenticator and AS reside on the same machine. The LEAP authentication procedure is performed as follows [74]:

1. The Supplicant associates with the AP and exchanges its identity with the AS. This step is identical for all EAP methods.

2. The AS sends an 8-byte challenge C_s, where C_s = Random8(seed), to the Supplicant.

3. The Supplicant generates, as described in RFC 2433 [292], a 24-byte challenge response R_p, where $R_p = \text{ChallengeResponse}(C_s, H)$, $H = \text{MD4}(\text{Unicode}(\text{PW}))$ and PW is the password of the user. R_p is then sent to the AS.

4. The AS calculates $R_{\text{check}} = \text{ChallengeResponse}(C_s, H)$. The exchange is successful if R_p and R_{check} match.

5. In case of success, an EAP-Success message is sent from Authenticator to the Supplicant. Then, AS and Supplicant switch roles and repeat steps 2 to 4. This time we denote the challenge sent by the Supplicant as C_p, and the response by the AS as R_s.

6. The AS derives the Session Key (SK) as

$$\text{SK} = \text{MD5}(\text{MD4}(\text{Unicode}(H))||C_s||R_p||C_p||R_s) \tag{4.1}$$

   where "||" is the concatenation operator. The AS encrypts this value with the Remote Authentication Dial-In User Service (RADIUS) secret and sends it to the Authenticator. The Supplicant also derives the SK, so this key can be used for Wired Equivalent Privacy (WEP) encrypted unicast communication. Finally, a random broadcast key is generated by the Authenticator and sent encrypted with the unicast key to the Supplicant.

Figure 4.2.1 shows a diagram of the previous algorithm. Note that a LEAP exchange is practically identical to performing two MSCHAPv1 authentications (steps 2 to 4): one from AS to Supplicant ($C_s \rightarrow R_p$) and one from Supplicant to AS ($C_p \rightarrow R_s$). [292].

Next, let us examine PEAP. This authentication method is significantly more complex, and among other features supports MSCHAPv2 mutual authentication to protect against MITM attacks [188, 189]. Assuming cryptographic binding is not used (see Section 4.4.3), PEAP authentication is performed as follows:

1. The Supplicant associates with the AP and exchanges its identity with the AS.

2. In Phase 1, the Supplicant and AS set up a TLS tunnel similar to the procedure described in RFC 5246 [79]. From the TLS master secret, a Master Session Key (MSK) is derived via a one-way function. This key serves a comparable purpose to the SK from LEAP.

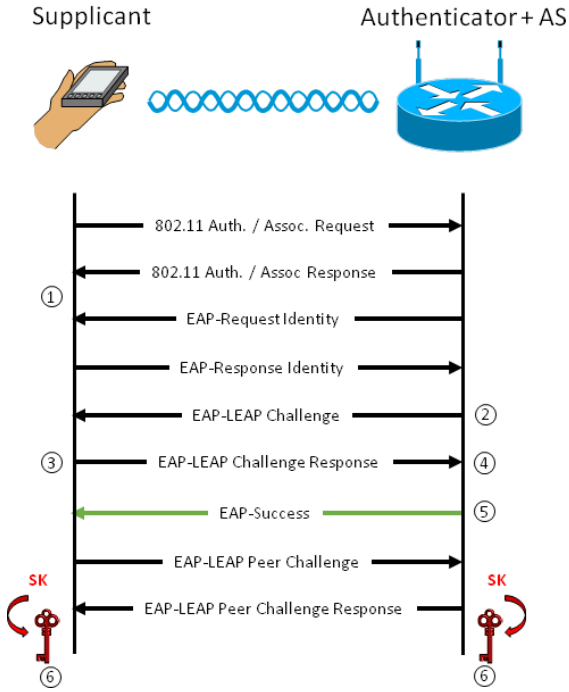


Figure 4.2.1: General LEAP authentication sequence

3. Phase 2 is performed inside the TLS tunnel and implies usage of an EAP inner authentication method. MSCHAPv2 is frequently used for this purpose. Assuming MSCHAPv2 is used, the AS starts by generating a 16-byte random server challenge $C_s = \text{Random16}(seed)$ and sends it to the Supplicant.

4. The Supplicant also generates a random 16-byte peer challenge C_p. Then the challenge response is calculated as

$$R_p = \text{ChallengeResponse}(\text{Challenge}(C_s), H) \tag{4.2}$$

where $\text{Challenge}(C_s) = \text{SHA1}(C_p||C_s||U)[0:7]$, U is the username of the user, $H = \text{MD4}(\text{Unicode}(\text{PW}))$, PW is the password of the user and $[0:7]$ means the first eight bytes of the data. This challenge response is transmitted back to the AS, along with C_p and U.

5. The AS calculates R_{check} analogous to R_p in step 4. R_{check} and R_p must match, or the authentication will fail.

6. The AS calculates a peer challenge response

$$R_s = \text{PeerResponse}(\text{MD4}(\text{Unicode}(H)), M_1, R_p, \text{Challenge}(C_p), M_2) \tag{4.3}$$

where M_1 is the string "Magic server to client signing constant" and M_2 is the string "Pad to make it do more than one iteration". This result is SHA1-hashed and sent to the Supplicant.

7. The Supplicant authenticates the server, completing the MSCHAPv2 inner authentication.

8. An EAP-Result-TLV exchange is performed between AS and Supplicant to indicate the result of the PEAP authentication. Then an EAP-Success message is sent.

9. The MSK is used to derive the WPA2 Pairwise Master Key (PMK) and subsequent keys. Secure transmission of data can begin when the 802.11i four-way handshake [117] is completed.

Figure 4.2.2 visually illustrates a PEAP exchange. When comparing the core differences between MSCHAPv1 and MSCHAPv2 credentials from RFCs 2433 and 2759, we can see that they are in fact very minor. Table 4.2.1 shows a comparison between the two methods [231, 291, 292].

Though RFC 2759 states that MSCHAPv2 is incompatible with MSCHAPv1 [291], the insignificance of the aforementioned differences led us to the conclusion that some MSCHAPv1 messages can be converted to MSCHAPv2 messages and vice versa.

We will now show that C_s from MSCHAPv1 is identical to $Challenge(C_s)$ from the MSCHAPv2 AS and that R_p from the MSCHAPv1 peer is identical to R_{check} at the MSCHAPv2 server. This way we can be sure that all messages converted from MSCHAPv1 to MSCHAPv2 or vice versa will be accepted by the destination host. For the challenges we derive:

$$\begin{aligned}
\text{Challenge}(C_s) &= \text{SHA1}(C_p||C_s||U)[0:8] && (4.4)\\
&= \text{SHA1}(x)[0:8] \ (C_p \text{ and } C_s \text{ are random}) && (4.5)\\
&= \text{Random8(seed)}, \quad \text{x is random} && (4.6)\\
&= C_s && (4.7)
\end{aligned}$$

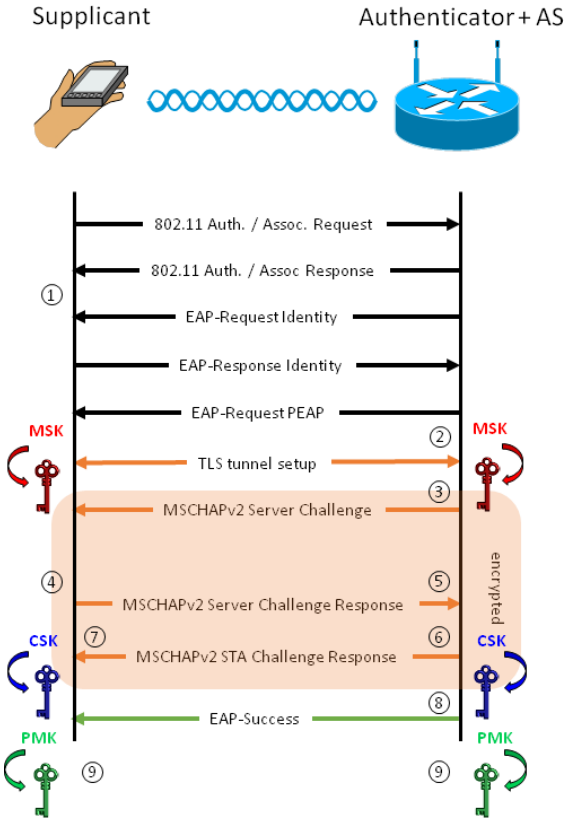


Figure 4.2.2: General PEAP authentication sequence

Given that the ChallengeResponse function is the same in MSCHAPv1 and MSCHAPv2, we derive for the challenge responses:

$$R_s = \text{ChallengeResponse}(C_s, H) \tag{4.8}$$
$$= \text{ChallengeResponse}(\text{Challenge}(C_s), H) \quad \text{(Eq. 4.7)} \tag{4.9}$$
$$= R_{\text{check}} \tag{4.10}$$

With the knowledge that the challenge we get from the PEAP MSCHAPv2 AS can be converted to an MSCHAPv1 challenge (Equation 4.7), and that the challenge response we get from our LEAP MSCHAPv1 victim can be converted to an MSCHAPv2 challenge response that matches R_{check} on the AS (Equation 4.10), we devised a relay attack that uses a vulnerable device as an MSCHAPv2 chal-

Table 4.2.1: Differences between MSCHAPv1 and MSCHAPv2 exchanges

	MSCHAPv1	**MSCHAPv2**
C_s	C_s = Random8(seed)	C_s = Random16(seed)
C_p	C_p = Random8(seed)	C_p = Random16(seed)
R_s	R_s = ChallengeResponse(C_p, H)	R_s = PeerResponse(MD4(Unicode(H)), M_1, R_p, Challenge(C_p), M_2)
R_p	R_p = ChallengeResponse(C_s, H)	R_p = ChallengeResponse(Challenge(C_s), H)

lenge response oracle in order to gain unauthorized access to PEAP networks. Figure 4.2.3 shows a schematic representation of our attack.

4.3 Practical LEAP relay attack

Our attack essentially exploits a combination of two vulnerabilities. A first vulnerability is the fact that some devices accept the LEAP method as an alternative method for authentication. Since the LEAP method does not establish a TLS tunnel from client (or "Supplicant") to AS prior to exchanging credentials, it is vulnerable to a rogue AS MITM attack [74].

The second vulnerability is that when the user configures or joins a PEAP network, some devices reuse the supplied or installed credentials for all supported EAP methods. Hence, the LEAP credentials do not have to be re-entered explicitly by the user. Existing MITM attacks try to capture these LEAP credentials using a rogue AS, and then crack them with dictionary attack tools like `asleap`[1]. In our attack, we will use the credentials to attack PEAP.

In this section we will show how the MSCHAPv1 to MSCHAPv2 conversion can be exploited in practice. First we will discuss the preconditions for the attack. Then, a practical implementation for attacking Apple devices will be demonstrated.

4.3.1 Preconditions

A device connecting to a PEAP network is considered vulnerable to our attack when all of the following preconditions are met:

[1] This tool can be downloaded from the following URL: `http://www.willhackforsushi.com/?page_id=41`

- The device supports the LEAP method.
- The device connects automatically to the PEAP network. This is the default behavior.
- The Authenticator does not require and validate *client* certificates. Server certificate validation and certificate pinning may be enabled on the client.
- The MSCHAPv2 or MSCHAPv1 inner authentication EAP method is supported and allowed on the AS.

Note that most of the preconditions listed here are commonly fulfilled by default in enterprise network setups.

4.3.2 Case study: Apple devices

We will now demonstrate how the exploit can be practically applied to Apple devices (see Figure 4.2.3). Our proof-of-concept implementation uses a simple state machine to perform the attack (Figure 4.3.1). After successful execution, an attacker gains unauthorized access to the target network by impersonating a legitimate user.

State 1: Association

Before wireless clients can begin the exchange of EAP packets in a wireless network, they require association with a wireless AP. We exploit the default auto-join behavior to have clients associate to an AP under our control. In order to accomplish this, we set up a fake wireless AP with the same SSID and parameters as the target network. This fake AP broadcasts beacon packets and replies to `Probe Request` frames from clients.

The client will associate or reassociate to our fake AP when it is closer than the target network AP, because better signal strength is preferred [90]. Alternatively, we can operate our fake AP on a different channel [266, 269] or force the client to connect to it by performing reactive jamming similar to the attacks described in [44, 266]. Since we do not want to receive requests from devices that are not vulnerable, our implementation uses the MAC address Organizationally Unique Identifier (OUI) to identify the device vendor. We can filter out all non-vulnerable devices this way.

State 2: Identification

The first step after association in WPA2-Enterprise networks is identification. The AS has to know which user wants to authenticate in order to match corresponding credentials. We can learn the identity of the vulnerable device by sending an EAP `Identity Request` frame. The device will then reply with an EAP `Identity Response` which contains the username of our victim.

At this point, data sent over the air is still not encrypted. Hence, some PEAP configurations will use anonymous identities. In this case the real username is only disclosed when a TLS tunnel has been established between the Supplicant and the AS. Nonetheless, we can still get the real username in a later phase of our attack.

Our next goal is to get the challenge value from the target AP. We created a modified version of the `wpa_supplicant`[2] tool for this purpose. At the end of this state, the binary executable of this modified version is called from our implementation.

State 3: Challenge

In State 3, we wait for the `wpa_supplicant` tool to establish a TLS tunnel with the target AS and extract an MSCHAPv2 challenge from the inner authentication. We can now see why usage of client certificates would mitigate the attack, as the client certificate validation would not be successful in this case.

When the MSCHAPv2 challenge is retrieved, we pass it on to our tool. Upon receipt, the tool will periodically send `LEAP Request` messages (containing the extracted challenge) to the Apple device in order to keep the session alive.

State 4: Response

After receiving the `LEAP Request`, our victim will reply with a `LEAP Response` which contains an MSCHAPv1 challenge response to our MSCHAPv2 challenge. Should the target PEAP network enforce anonymous identities, the real or inner identity of the victim will also be revealed to the attacker through this `LEAP Response`. Next, our implementation will forward the received MSCHAPv1

[2] `wpa_supplicant` is an open source 802.1X supplicant implementation by Jouni Malinen.

Table 4.3.1: Devices vulnerable to the LEAP relay attack

Device	Vulnerable
iPod Touch (iOS 6.1.6)	✓
iPhone 4 (iOS 7.1)	✓
iPhone 4S (iOS 7.1)	✓
Mac OS X 10.8.2 (Mountain Lion)	✓
Samsung GT-S5570 (Android 2.3.4)	✗
Google Nexus 7 (Android 4.4.2)	✗
Samsung GT-I8750 (Windows Phone 8.0)	✗
Windows 7 Desktop	✗

challenge response as an MSCHAPv2 challenge response to the modified `wpa_supplicant` tool, which will in turn forward the challenge response to the legitimate PEAP network AS.

State 5: Success

When the AS receives our modified challenge response, authentication proceeds as usual, which means the AS has to authenticate to our Supplicant. However, since we are not in possession of the NT password secret, we cannot derive H. Hence, when receiving the peer challenge response from the AS, we are forced to accept any sent value.

After this, the MSCHAPv2 inner authentication will complete successfully and the port will be authenticated. The AS and our Supplicant will derive the MSK, and from this we can derive the PMK. We now have all components required to access resources on the internal network.

4.3.3 Test results

We tested our attack on devices from multiple vendors. Table 4.3.1 shows on which devices the LEAP relay attack was successfully performed.

Assuming that the same network protocol stack is used on all Apple operating systems, we concluded from these results that all Apple devices are vulnerable. The attack was executed analogously on each device for multiple APs, using differ-

ent AS implementations. These included a TP-Link WN422G using `hostapd` and the latest `freeradius` implementation on the same machine, a Linksys WRT54G AP using the latest `freeradius` implementation on a dedicated machine, and a Ubiquity UniFi AC 3.x AP using Windows RADIUS server on a dedicated machine.

4.4 Mitigation

The attack we described in this chapter can be mitigated in various ways. We will discuss five methods in this section.

4.4.1 Client certificates

In State 3 of our attack, a TLS tunnel has to be established between the attacker and the target network AS. When using client certificates similarly to EAP-TLS, each client's certificate must be provided in the "Client Hello" phase of the TLS tunnel setup. When this verification fails, the TLS setup will be aborted and hence, our attack will fail because the MSK cannot be derived from the TLS master secret.

This countermeasure is very effective and by far the most secure. However, it would require a lot of administration effort for enterprises. Especially in enterprises with a BYOD policy, because a signed certificate for every device allowed on the network must be installed on the AS.

4.4.2 iPhone Configuration Utility

iPhone configuration profiles allow the network administrator to choose which EAP methods clients must use (see Figure 4.4.1). They are the only way in which LEAP can be disabled on the examined Apple devices. If this method is chosen to mitigate the attack, care must be taken in BYOD environments: if one user does not install the network profile, the attack can nevertheless be executed. Furthermore, network profiles can be accidentally removed by the user. For these reasons, security is put in the hands of the end user and therefore this method is not as secure as using client certificates.

4.4.3 Cryptobinding

An optional feature described in the PEAP version 2 internet draft is cryptographic binding [189]. This feature introduces the use of a new Type-Length-Value (TLV), the CryptoBinding TLV, to address MITM attacks. A two-way handshake containing a Compound MAC Key (CMK) proves that the two authentications terminate at the same PEAP peer and PEAP server [173].

To calculate the CMK, the Supplicant is required to use keying material from both Phase 1 and Phase 2 of the PEAP exchange. In practical terms this involves the calculation of the Tunnel Key (TK) and the Inner Session Key (ISK). These keys are combined in the cryptobinding algorithm to form the CMK.

The TK is calculated similarly to the MSK from the TLS master secret, and would be available to an attacker. The ISK however, is calculated at the Supplicant as ISK = InnerMPPESendKey||InnerMPPERecvKey. The InnerMPPESendKey and InnerMPPERecvKey are both derived from the inner MSCHAPv2 Master Key (MK), which is derived as

$$\mathrm{MK} = \mathrm{GetMasterKey}(\mathrm{MD4}(\mathrm{Unicode}(H))[0:16], R_s) \tag{4.11}$$

Since H is unknown to the attacker, the ISK cannot be derived and authentication will fail.

If all consumer devices would support cryptobinding, this method would probably be the best way to mitigate our attack. However, from our experiments we concluded that Apple devices did not support cryptographic binding at the time the vulnerability was discovered.

4.4.4 Intrusion detection

A signature based Wireless Intrusion Detection System (WIDS) might be able to detect our attack by passively scanning for LEAP requests. Since these packets will never be sent by a legitimate AP, WIDS sensor nodes have a clear indication that the network is under attack. Analytic approaches to detect our attack may include station counts, association counts, OS fingerprinting and Received Signal Strength Indicator (RSSI) value analysis, though these methods often lead to false positives [57, 122].

As a final note, we would like to indicate that care must be taken when relying on a WIDS for detection of an attack, as we believe that in many cases the WIDS may be bypassed. For example, a victim may be in range of the rogue AP, while the latter is out of range from a WIDS sensor. In Figure 4.4.2, an example scenario is shown where the relay attack is executed over the internet.

4.4.5 Rogue AP mitigation

If one can prevent the attacker from setting up a rogue AP, the LEAP relay attack cannot be performed. Several methods have already been developed to mitigate the rogue AP attack [21, 222, 284]. However, we believe not all of these mitigation strategies will work. Context leashing will only work when creating the rogue AP in a different context, EAP-SWAT will have the same problems as PEAP, and other mitigation strategies in these works rely partly on the awareness and expertise of the user. We believe link layer protection mechanisms would be the most effective in this case.

4.5 Related work

A similar, generalized MITM attack on tunneled authentication protocols was demonstrated by Asokan et al. in 2002 [12]. Related attacks on PEAP vendor implementations such as the EAP dumb-down attack were introduced by Raul Siles in 2013. This attack exploits the default lack of certificate validation in mobile devices. However, for Apple devices, the dumb-down attack requires user intervention whereas our attack is automatic [243]. Furthermore, a correct configuration of authentication server certificates does not mitigate our attack for Apple devices.

Schneier et al. have highlighted a number of weaknesses in MSCHAPv1 and MSCHAPv2 [231]. In 2008, Joshua Wright and Brad Antoniewicz demonstrated how EAP credentials such as MSCHAPv2 exchanges can be collected using `freeradiuswpe`, a rogue AS implementation [282]. By using the `asleap` tool, these credentials can then be cracked with a dictionary attack [58]. Then in 2012, Moxie Marlinspike showed how MSCHAPv2 credentials can be cracked in less than 24 hours using cloud-based Field-Programmable Gate Array (FPGA) nodes [163]. Finally, Josh Yavor indicated the dangers of BYOD and default certificate validation behavior of mobile devices in 2013 [285].

4.6 Chapter conclusions

We demonstrated how MSCHAPv1 challenges and challenge responses can be converted to MSCHAPv2 challenges and challenge responses. Then, we indicated how this can be exploited in practice when a Supplicant supports the insecure LEAP method and when credentials are reused between EAP methods.

From our experiments we concluded that all Apple devices were vulnerable to our attack at the time of the discovery of the vulnerability. We discussed various mitigation strategies and their applicability to enterprise wireless networks, and suggest that cryptobinding should be enabled as an effective countermeasure. For devices that satisfy all mentioned vulnerability preconditions but do not support cryptobinding, we instead suggest to disable LEAP entirely, for example by installing configuration profiles in the case of older Apple devices.

The vulnerability was assigned CVE-2014-4364 after being reported to Apple through a responsible disclosure procedure, along with our recommendations. It was subsequently mitigated by Apple by disabling LEAP by default on all devices. However, LEAP can be enabled manually through configuration profiles, meaning an attack may still be possible if cryptobinding is disabled as well.

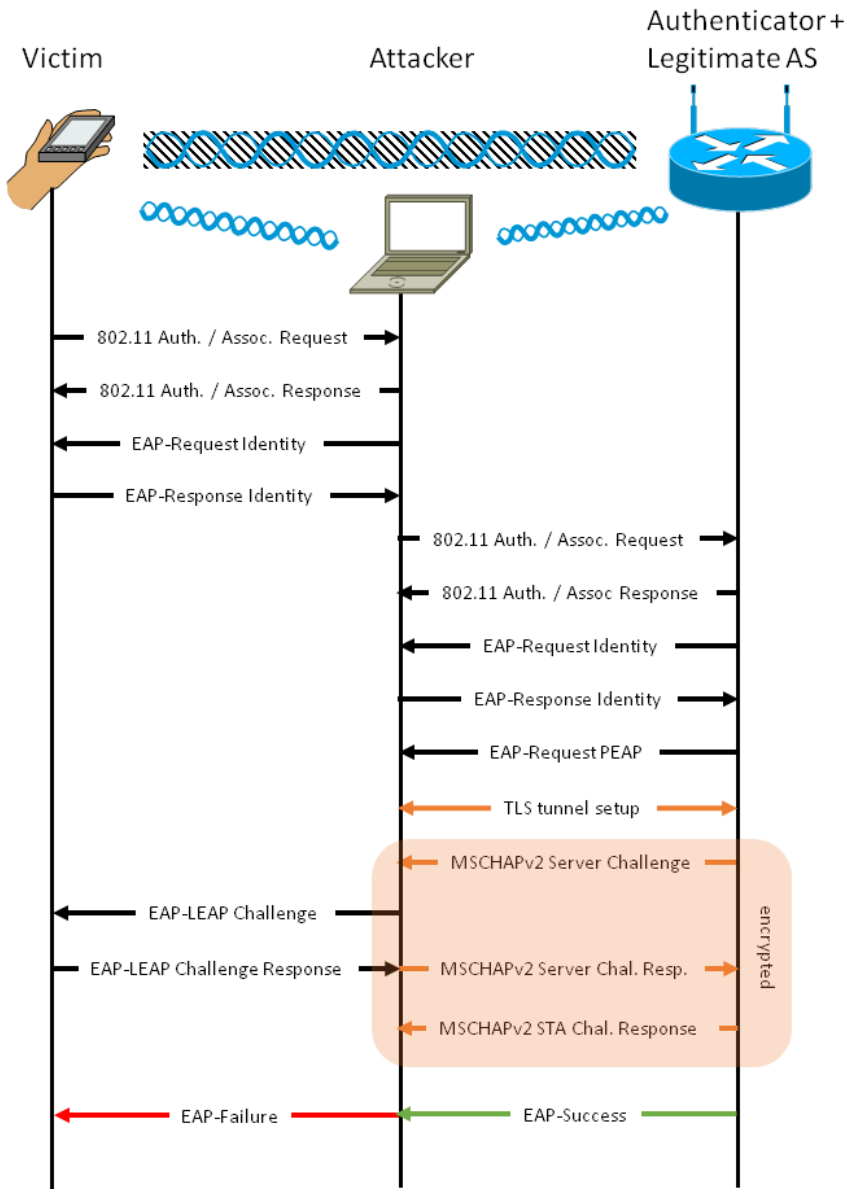


Figure 4.2.3: Schematic representation of the Apple LEAP attack

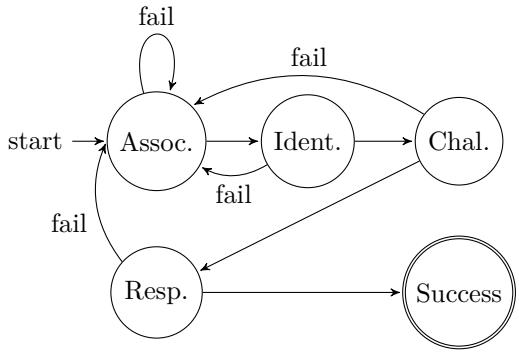


Figure 4.3.1: State machine of our attack

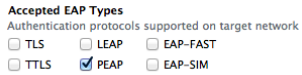


Figure 4.4.1: Disabling LEAP on Apple devices through configuration profiles

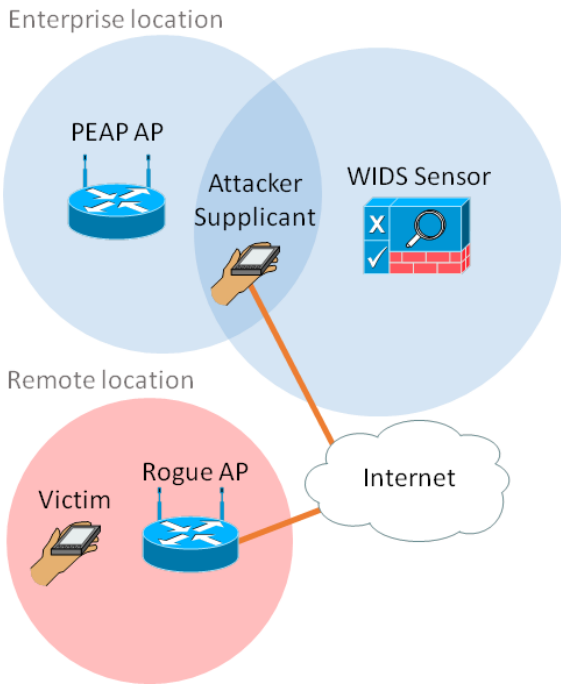


Figure 4.4.2: Remote LEAP relay attack

5

Injection Attacks on 802.11n MAC Frame Aggregation

5.1 Introduction

Throughout the years, several amendments have been made to the original IEEE 802.11 standard in order to meet the increasing throughput demands of wireless networks. The first amendment to increase the data rate as the result of improvements on both the PHY layer and the MAC layer is 802.11n [120, 194]. Here, a key improvement on the PHY layer is the introduction of Multiple Input, Multiple Output (MIMO) technology, where multiple spatially separated antennas are used on both the receiver and transmitter to create multiple spatial streams [127] and reduce multipath interference. Another addition is the optional usage of an increased channel bandwidth of 40 MHz instead of 20 MHz [120, 194].

Besides these improvements on the PHY layer, the efficiency of the MAC layer needed to be improved as well in order to go beyond speeds of 100 Mbps. Here, the most notable efficiency improvement is frame aggregation [194], which allows a station to transmit or receive multiple MAC frames in a single PHY frame, reducing overhead due to headers and interframe spacing. All of the improvements on the PHY and MAC layer combined increase the theoretical maximum raw data rate of 802.11n to 600 Mbps [120, 248], which ultimately contributed to 802.11n devices dominating the Wi-Fi market in 2013 [6].

Despite the beneficial effect of PHY and MAC layer improvements on the data rate, these additions introduce new opportunities for security issues to arise. In the case of 802.11n, frame aggregation is especially interesting to look at. Although the standard specifies aggregation principles, frame structures and the general mechanisms, it provides mere guidelines for the actual implementation [127]. It is up to developers to implement their own aggregation schemes that determine when, why and even how to aggregate individual frames [91, 257]. Such schemes are typically implemented in the device driver or on the device firmware.

In this chapter, we will explore the frame aggregation mechanism introduced in the 802.11n standard. We will show how the frame aggregation algorithm provided by the 802.11n standard introduces a remote arbitrary frame injection vulnerability on MAC hardware that implements this algorithm, thereby leaking information from higher layers (e.g. the Application layer) into the MAC layer. In Section 5.2, we will detail the contributions of this work, and describe several related works. Section 5.3 describes the properties and implementation of the 802.11n frame aggregation mechanism, which is required knowledge for understanding our attack. Next, in Section 5.4, we describe the attack itself, along with a feasibility study and a discussion of the results and potential impact. In

Section 5.5 we propose several measures that can be put in place to defend against our attack. Finally, Section 5.6 describes our conclusions.

5.2 Related work and contributions

Goodspeed et al. introduced the Packet-In-Packet (PIP) technique [105] in 2011, and applied it to IEEE 802.15.4 [103, 105]. Additionally, they used this technique to inject raw PHY layer frames into 802.11b networks for data rates of up to 2 Mbps [104]. Barisani et al. demonstrated in 2013 how, in rare cases, the PIP technique can be applied to 802.3 wired links [20]. In 2014, Jenkins et al. used the PIP technique for fingerprinting 802.14.5 and ZigBee receivers [131]. Similar PIP principles have been used by Dabrowski et al. in order to embed barcodes inside other barcodes [65]. Ossmann et al. propose a defense against PIP in the field of radio communications by using error correcting codes [185]. Finally, Sassaman et al. describe how formal language theoretic methods can be used as a defense against injection attacks in general [227].

Our main contribution in this chapter (**C2**) is a methodology for remote arbitrary frame injection into wireless networks that implement the 802.11n standard or newer standards such as 802.11ac. This is accomplished by applying PIP principles to the MAC frame aggregation mechanism. To the best of our knowledge, this approach has not been demonstrated before. Previously, the PIP technique could only be used to perform frame injection attacks on 802.11b networks [104] by tricking the radio into interpreting a raw PHY layer packet which is embedded in the payload of another packet. As Goodspeed et al. mentioned in their work, several complications and challenges exist that limit the practical feasibility of their approach:

- The symbol set used in the Physical Layer Convergence Protocol (PLCP) Protocol Data Unit (PPDU) must be included in the symbol set used for the injected payload. Otherwise it will not be possible to inject a valid PHY layer header. For example: if the data portion of the frame is modulated using Complementary Code Keying (CCK), it will be difficult for an attacker to inject a valid PLCP preamble, which is modulated using Differential Binary Phase Shift Keying (DBPSK).
- Even if a favorable symbol set is used, the data rates must be compensated for. The 802.11 standard specifies different data rates for different sections of the frame. For example, the PLCP preamble must always be transmitted using the 1 Mbps DBPSK modulation. The `SIGNAL` field of a PPDU

however, can then indicate to use a higher data rate for the remainder of the frame [127].

- 802.11 PLCP Service Data Units (PSDUs) are scrambled using a 127-bit sequence, which must be accounted for by the attacker.

- If the modulation technique uses differential signaling, the attacker must account for this as well.

The methodology presented in our work removes all of these complications by operating on the MAC layer. Consequently, our injection method can be applied regardless of the chosen data rate, symbol set, signal whitening, or modulation technique. Hence, the practical exploitability and attack surface of frame injection attacks is increased. To justify our claim of practical feasibility, we present two realistic proof-of-concept attacks and estimate their success rate. Since this success rate depends on the aggregation behavior of the wireless Network Interface Controller (NIC), this aspect is briefly discussed as well. Finally, we propose several defensive measures which can be applied to prevent exploitation of our attack.

5.3 Background

Before we discuss our injection attack, we will give a brief overview of some of the relevant PHY and MAC layer features introduced in 802.11n. These features have notable implications for Goodspeed's PIP attack, as well as the injection attack that we will discuss in Section 5.4.

5.3.1 PHY features

On the PHY layer, two features added in 802.11n have some interesting properties for this research. First, there are the added PLCP frame formats, which have additional fields and hence additional capabilities in comparison to previous versions of the standard. Secondly, there is a set of new Modulation and Coding Schemes (MCSs) for higher data rates which we used throughout our experiments.

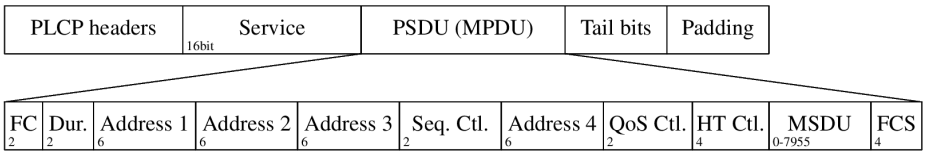


Figure 5.3.1: PLCP and MAC frame structures

PLCP frame format

The PLCP acts as an interface between the PHY and MAC layers. It defines a frame format named the PLCP Protocol Data Unit (PPDU), which consists of a PLCP Preamble, PLCP Header, and a PSDU. In 802.11n, the PPDU can have several formats depending on the capabilities of the transmitting device:

- Non-High Throughput (HT) format: legacy frame format as specified by previous versions of the standard.
- Mixed format: format that is backwards compatible with the 802.11 a/g format.
- Greenfield format: HT frame format that can only be used by devices that are 802.11n compatible.

The fields specified in the PPDU contain the parameters that determine how the device hardware should transmit a packet. Examples of such parameters are the modulation scheme, transmission length, and aggregation flag. Figure 5.3.1 shows the generic PLCP frame format. Naturally, a frame transmitted using a certain set of parameters can only be received by a device that supports said parameters. This set of supported parameters is also referred to as the capability set of the device.

Modulation and Coding Scheme

The MCS is determined by the index specified in the `MCS` field of the PPDU. The mapping between indices and the corresponding mandatory rates is given in Table 5.3.1. The last column of the table indicates the data rate when a short Guard Interval (GI) is used.

Table 5.3.1: MCS parameters for mandatory 20 MHz [127].

MCS	Modulation	Coding	Data rate (Mb/s) GI	SGI
0	BPSK	1/2	6.5	7.2
1	QPSK	1/2	13.0	14.4
2	QPSK	3/4	19.5	21.7
3	16-QAM	1/2	26.0	28.9
4	16-QAM	3/4	39.0	43.3
5	64-QAM	2/3	52.0	57.8
6	64-QAM	3/4	58.5	65.0
7	64-QAM	5/6	65.0	72.2

5.3.2 MAC features

802.11n related extensions of the MAC layer include changes in the frame format and the addition of two types of frame aggregation that were introduced in order to reduce MAC layer overhead: MAC Service Data Unit (MSDU) aggregation or A-MSDU, and MAC Protocol Data Unit (MPDU) aggregation or A-MPDU.

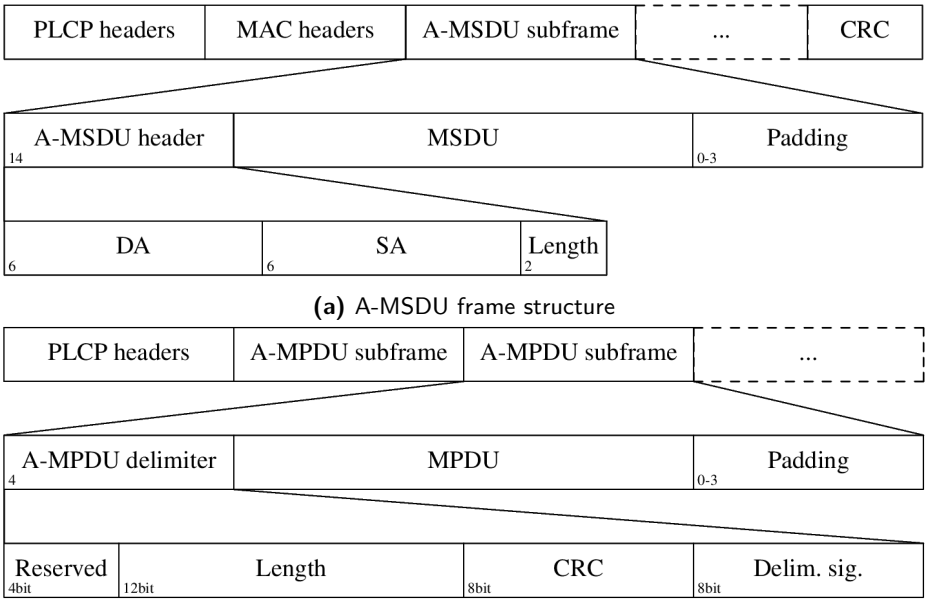
MAC frame format

The MAC frame format is defined in the MPDU. Its original frame structure is extended with an optional `HT Control` header field as shown in Figure 5.3.1. The presence of this field is indicated by the `Order` subfield of the `Frame Control` field [127]. Other new additions are the HT Capabilities and HT Operation Information Elements (IEs), which are included in the Beacon frames of the AP, and in the Probe Requests and (Re)Association Requests of stations [91].

The MPDU of the MAC layer is equivalent to the PSDU of the PHY layer. The higher layer payload of the frame is included in the `MSDU` field of the MPDU.

Aggregate MSDU

With A-MSDU aggregation, the transmitter collects multiple MSDU subframes from the Logical Link Control (LLC) sublayer and prepends them with an A-MSDU subframe header, which is structurally equivalent to an 802.3 header: it



(a) A-MSDU frame structure

(b) A-MPDU frame structure

Figure 5.3.2: Frame structure of A-MSDUs (a) and A-MPDUs (b).

contains the Destination Address (DA), Source Address (SA), and length of the MSDU subframe as shown in Figure 5.3.2a.

The MSDU subframes are aggregated and transmitted in a single MPDU when either the maximum A-MSDU size (7935 bytes) is reached, or when the transmission delay reaches a predefined threshold. Each A-MSDU subframe must be followed by a number of zero padding bytes, so that the length is a multiple of 4 bytes. The final A-MSDU subframe has the `A-MSDU Present` flag in the Quality of Service (QoS) header set to true. Furthermore, the `DA` and `SA` fields of the MSDUs must be identical to respectively the `Receiver Address (RA)` and `Transmitter Address (TA)` fields of the MPDU [127].

A disadvantage of this aggregation method is that it performs poorly in situations where the packet error rate is high [101]. This is due to the fact that there is only a single Cyclic Redundancy Check (CRC) for the entire aggregate frame.

Aggregate MPDU

A-MPDU aggregation collects multiple frames from the MAC sublayer and aggregates them in a single PHY frame of maximum 65,535 bytes. Here, each subframe is prepended with an A-MPDU delimiter as shown in Figure 5.3.2b. As with A-MSDU aggregation, each subframe is padded with zero bytes so that its length is a multiple of 4.

For the subframes to be valid, they must have the same `RA` and `Duration` fields[1], i.e. the receiving host and frame lengths must be the same, and the same `KeyID` field must be provided in case encryption is enabled. The maximum subframe size is 4095 bytes [127].

The A-MPDU delimiter of a subframe is shown in Figure 5.3.2b, and contains the following fields:

- Reserved: Unused bits with a possible future application.
- Length: Length of the A-MPDU subframe in bytes. This length can be 0 bytes, in which case the A-MPDU delimiter is used for padding purposes.
- CRC: 8-bit CRC of the `Reserved` and `Length` fields.
- Delimiter signature: Pattern that indicates the start of an A-MPDU subframe. This pattern is defined as the ASCII value for the character 'N'.

Note that contrary to A-MSDU aggregation, a packet error in one subframe does not result in the entire aggregate frame being dropped. Instead, only the erroneous subframe is dropped. If the subframe error occurred in the A-MPDU delimiter, the 802.11n specification suggests that the next `Delimiter signature` should be searched for according to the algorithm in Listing 5.1.

In essence, this algorithm searches for any A-MPDU delimiter signature on a 4-byte boundary. If a valid delimiter signature has been found, the `recv_mpdu` function will be called. This function will check whether the value specified in the `CRC` field of the A-MPDU delimiter is correct, and if so, strips the delimiter and sends the number of bytes specified in the `Length` field of the A-MPDU delimiter as an individual MPDU up to the MAC protocol driver for further processing.

[1] It should be noted that whether this requirement is enforced depends on the vendor's implementation.

```
void parse_mpdu (int length)
{
  int offset = 0;
  while (offset+4 < length)
  {
    if (valid_mpdu_delimiter(offset) &&
      mpdu_len(offset) <= (length - (offset+4)))
    {
      recv_mpdu(offset+4, mpdu_len(offset));
      offset += 4 + 4 * ((mpdu_len(offset)+3)/4);
    }
    else
    {
      offset += 4;
    }
  }
}
```

Listing 5.1: A-MPDU scanning and parsing algorithm [127, p. 2661]

Note that because this algorithm is performed by the hardware MAC component of the chip, it is not possible to observe the A-MPDU in its entirety on the host. Only the individual MPDUs will be visible to the firmware and driver of the device.

Our hypothesis is that the above algorithm can be exploited by an attacker. Observe that since the symbol set and data rate used by the payload data is the same as the symbol set and data rate used by the A-MPDU delimiter, a vulnerability is introduced: if one subframe has an incorrect delimiter, the scanning algorithm will overflow into the payload and parse this payload as if it were header data. An attacker can therefore define their own subframe boundaries by using a specially crafted payload.

5.4 Frame injection attack

We have investigated whether our hypothesis is correct and whether the A-MPDU frame aggregation mechanism can be exploited in practice. We found that this is indeed the case and demonstrate a proof-of-concept remote frame injection attack. This attack allows an attacker to inject arbitrary frames into 802.11n networks from any location, by leveraging the PIP technique [105] at the data link layer. For this attack to succeed in practice, a number of conditions must be true:

- The last hop between the attacker and the victim transmits packets wirelessly.
- Encryption is disabled. In other words, the AP is an open network (e.g. hotspot, internet cafe, airport, or other open AP).
- The AP and associated victim are configured for 802.11n operation.

In order to determine whether the network is vulnerable, the first two conditions are trivial to determine. The last condition can be evaluated based on the presence of the `HT Capabilities` IE (see Section 5.3.2). In all of the following experiments we assume that these conditions are true.

5.4.1 Experimental setup

While testing the applications and success rate of our frame injection attack, we used the setup depicted in Figure 5.4.1. Here, the attacker's Linux machine is connected to the internet via an Ethernet port. An AP is used to provide internet access for the wireless network to which the victim is connected. For our experiments we used four different models: a MikroTik CRS109 (AR9344 chip), a Linksys E1200 (BCM5357C0 chip), a Sitecom WLR-3100 (MediaTek MT7620N chip), and a Linux machine running `hostapd` (AR9271 chip). Each AP uses Network Address Translation (NAT) to translate between internal addresses and its external address, and is protected by a firewall.

Our victim's Linux machine is associated to the AP with a TP-Link TL-WN722N USB dongle, which uses the Atheros AR9271 wireless chip with the default open source `ath9k_htc` firmware[2]. The setup was placed in an office environment without any actively interfering stations.

5.4.2 Injection method

We now present our method for performing the frame injection attack. As a first step, the attacker is required to craft a valid A-MPDU delimiter, subframe, and padding as shown in the bottom part of Figure 5.4.2. The `Padding` fields are necessary to align the current and any subsequent A-MPDU delimiters to a

[2]This firmware is available for download at `https://github.com/qca/open-ath9k-htc-firmware`.

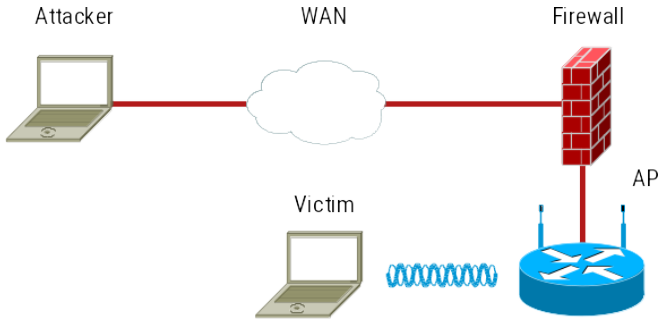


Figure 5.4.1: Experimental setup

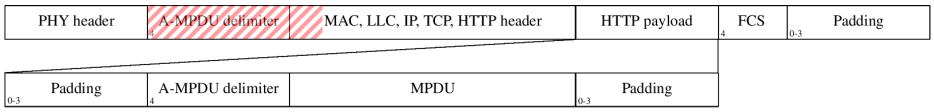


Figure 5.4.2: A-MPDU subframe injection

4-byte boundary as specified by the standard. Note that the length of the first `Padding` field is dependent on the length of all previous bytes of the frame, and hence on the link layer protocol used from the AP to the victim's machine as well. Based on the knowledge that the used link layer protocol will be 802.11 on the target network, the attacker can calculate the correct padding size. The second `Padding` field can simply be calculated as $4 - (\texttt{mpdu\_len} \mod 4)$.

Our crafted A-MPDU subframe can now be transmitted towards the victim. The attacker can embed this subframe in any higher layer protocol payload, such as an HTTP request. Ideally, we would like to trigger frame aggregation from the AP to our victim, and inject as many frames as possible. Therefore, a possible approach is to rapidly transmit repeated sequences of the crafted subframe to the victim.

When the payload is transmitted over the wireless link at the last hop, there is a non-negligible probability that part of the aggregated frame becomes corrupted, for example due to frame collisions, transmission errors, or interference from other radio protocols. If the corruption of any subframe is limited to only the A-MPDU delimiter, its `CRC` field will be invalid, and the algorithm from Listing 5.1 will be performed by the hardware of the wireless NIC in order to recover any following uncorrupted subframes of the aggregated frame. Hence, if a valid A-

MPDU subframe delimiter crafted by the attacker is present in the payload at a 4-byte boundary, this data will be interpreted as an A-MPDU delimiter instead of payload data. Figure 5.4.2 (top frame) demonstrates an aggregated frame where the A-MPDU delimiter of a subframe is corrupted.

After parsing the attacker's A-MPDU delimiter, the hardware will pass the malicious subframe to the host or device firmware. It should be mentioned that the Frame Check Sequence (FCS) of the injected frame must be correctly calculated by the attacker, or the frame will be dropped due to an invalid FCS at the MAC layer. However, this is trivial since the attacker has complete control over the injected MPDU. A library can be used to calculate the correct CRC over the injected MPDU.

In the work presented by Goodspeed et al.[104], frame injection in 802.11b networks is achieved by embedding an entire PHY frame in a higher layer protocol. Using our methodology however, it is not required to include PHY layer headers in the payload, because the injection happens at the MAC layer. The data rate and scrambler state do not have to be guessed, and the same symbol set is used for the distinction between frame header and frame data in our case. This increases the attack surface and chance of success from a practical point of view. The attacker only needs to make sure that A-MPDU aggregation is performed when the AP transmits the frames to the victim. Exactly when or why frames should be aggregated is not defined in the standard, and therefore depends on the device driver or firmware implementation [91, 257]. In our experience, an efficient method to trigger A-MPDU aggregation is to rapidly transmit packets of the same size to the AP. This was determined by means of a number of experiments that we will discuss in Section 5.4.4.

5.4.3 Applicability

In Section 5.4 we mentioned that only open wireless networks, such as hotspots and internet cafes, are vulnerable to our injection attack. At the time of writing, this is about 10.61% of the total number of known wireless networks [281]. An attacker would therefore have a reasonable chance of being able to inject packets into a random network if they would probe a random IP address range. Note that this may happen from any location or via a Wide Area Network (WAN) such as the internet.

Besides targeting an open network, a second requirement is that the victim implements A-MPDU frame aggregation in a 802.11n standard compliant manner.

Table 5.4.1: Tested devices vulnerable to A-MPDU subframe injection. The top 7 devices are client devices, whereas the bottom 3 are APs.

Device name	Chipset
Intel Dual Band Wireless-AC 7260	7260HMW
TP-Link TL-WN722N	AR9271
Netgear WNA1100	AR9271
CastleNet RTL8188CTV	RTL8188CTV
K11 Mini	RT5370
TL-WDN3200	RT5572
Nexus 5	BCM4339
MikroTik CRS109	AR9344
Linksys E1200	BCM5357C0
Sitecom WLR-3100	MT7620N

Since according to a research article by ABI Research 802.11n devices hold the largest market share of the consumer Wi-Fi device shipments in 2013 [6], and since A-MPDU reception support is a mandatory requirement in 802.11n [91], we believe a significant number of devices will be vulnerable to our demonstrated attacks. It should be noted that newer 802.11 standards, such as 802.11ac, implement A-MPDU frame aggregation as well. Therefore, devices that implement these standards will also be vulnerable. Table 5.4.1 shows the devices we tested. All of them are indeed vulnerable to our attack.

5.4.4 Optimal aggregation triggering

The 802.11n standard specifies that the frame size or a timer could be used for determining how many frames should be aggregated [127]. However, in practice it is very difficult to determine exactly which packets will be aggregated and which packets will be sent in the regular fashion, as this depends on the vendor's implementation. Moreover, the attacker has no means of measuring the properties of the wireless link at the remote location, such as the Packet Delivery Ratio (PDR), which can affect aggregation behavior as well [153].

From the attacker's point of view it would be interesting to see whether a generic, optimal transmission speed and frame size can be selected in order to maximize the probability of aggregation at the remote AP, as this will increase the probability of successful injection. We experimentally compared the aggregation behavior

of the four APs from our setup with respect to these parameters. In each experiment, we transmitted 250,000 frames from the attacker to the victim, and checked how many of them were forwarded by the AP as A-MPDU subframes.

The MikroTik AP and `hostapd` machine respectively use the AR9344 and AR9271 Atheros Wi-Fi chipsets, which have a number of hardware queues that receive frames from the firmware (AR9271) or driver (AR9344). Both chips aggregate frames depending on the total number of frames currently in these queues [46, 203]. Note that neither a timer is used nor are frames aggregated based on their size. Figure 5.4.3a shows the percentage of A-MPDU subframes transmitted by the MikroTik AP, which appears to favor small frames for aggregation.

Figure 5.4.3b shows the percentage of A-MPDU subframes received by the `hostapd` AP using the same setup. Despite the fact that this device implements the same aggregation strategy, it is much slower at emptying the transmission queues, which leads to increased aggregation – especially for larger frames. Therefore, the performance of the device itself impacts aggregation as well, and is an additional unknown to the attacker.

Finally, the Linksys AP (Figure 5.4.3c) and Sitecom AP (Figure 5.4.3d) respectively use the Broadcom and MediaTek chipsets, which appear to use frame aggregation more frequently. Since we observed that the aggregation rate decreases if the delay between successive transmissions increases, we can conclude that transmitting frames rapidly is a good strategy.

5.4.5 A-MSDU injection

In Section 5.3.2, A-MSDU aggregation was briefly discussed. A similar vulnerability in these kind of aggregated frames would allow an attacker to craft their own MSDUs. However, we determined that this aggregation method is not vulnerable to our injection attack.

To see why this is the case, let us consider the A-MSDU header from Figure 5.3.2a, which is different from the A-MPDU delimiter. For the attack to succeed, a random bit error must cause the `Length` field to become smaller, so that the attacker's payload is interpreted as the next A-MSDU header. This random `Length` value cannot be predicted by the attacker, which is a first issue that complicates exploitation.

Another consequence of not knowing the corrupted values from the previous fields

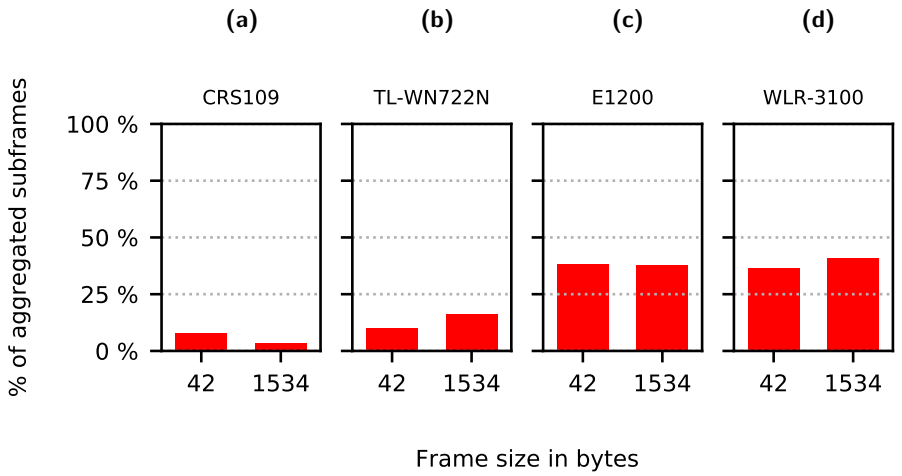


Figure 5.4.3: The percentage of aggregated subframes received per 250,000 transmitted frames from attacker to victim in our test setup shows that for the CRS109 and E1200, using small frames results in a slightly higher aggregation rate, whereas for the TL-WN722N and WLR-3100, using large frames results in a slightly higher aggregation rate.

is that the FCS of the MPDU cannot be calculated beforehand. An incorrect FCS would then automatically lead to the entire frame being dropped, resulting in a failed injection attack.

It should be noted that although the injection of A-MSDUs is unlikely to work in practice because of the above reasons, there is a possible use case for them in context of A-MPDU injection: the 802.11n standard specifies that an A-MSDU can be embedded inside an A-MPDU subframe [127] in order to increase the maximum size of the subframe. An attacker can therefore use this in order to increase the injection payload of a single subframe.

5.4.6 Attack scenarios

We have proposed a methodology for frame injection that can be used to inject arbitrary frames into a remote network. In this section, we will discuss two practical attacks that can be performed using this methodology. Our implementation is written in Python, uses the Scapy library for crafting the payload, and is open source [196].

Host scan

In our first attack, we apply our frame injection methodology to perform a remote scan for all active hosts on an internal network. We assume that the target network to scan is behind a remote AP, and that at least one service is accessible through this AP. This ensures that our payload will be sent wirelessly over the air. In our case, we configured a web server on the victim of our experimental setup. In practice this can be any service or open port, as long as the last hop between the attacker and the victim is a wireless link. For this experiment, the Sitecom AP was used, having its firewall configured to explicitly only allow HTTP packets and drop ingress ICMP requests.

In the above scenario, an attacker can craft HTTP POST requests containing A-MPDU subframes with ICMP echo requests as the payload, and transmit them to our victim machine. Note that since we are injecting at the MAC layer, the MAC address of the AP must be known[3] and used as the TA, or else the packet will be dropped by the victim. For the RA, the broadcast MAC address can be used.

After several repeated transmissions, we observed that frame corruption indeed caused the inner ICMP packet to be successfully received and replied to by the victim instead of the original payload, successfully bypassing the configured firewall rule. By checking which clients reply, we can then iterate over each destination IP address to perform a full host scan on the target network. A packet trace of this experiment is available at [198]. Here, we were able to scan 52% of a /24 network in 122 seconds using 665 MB of data.

When testing this attack on our Linksys and Mikrotik APs, we noticed that the ICMP replies by the victim were not forwarded to the attacker. The reason is that these APs use stateful firewalls, and hence block outgoing ICMP replies if there is no associated request. Nevertheless, even if stateful firewalls are used, the ICMP requests were successfully injected and interpreted by the victim for all tested APs and receivers.

Naturally, the injection of data frames is not limited to ICMP requests. An attacker can choose any frame in order to perform other attacks such as a Denial of Service (DoS) attack, Address Resolution Protocol (ARP) poisoning, or the injection of 802.11 management frames such as `Beacons` and `Deauthentication` frames. For each of the *data frame* related attacks to succeed, the attacker

[3]This could be done by looking up the SSID or location of the AP in a wardriving database such as WiGLE.net [281]

would need to know the Basic Service Set Identification (BSSID) to which the targeted host is associated. This is essentially the MAC address of the AP's wireless interface. If the BSSID is incorrect, the targeted host will drop the frame, which makes random attacks against an IP range infeasible. However, a determined attacker could look up the BSSID in a wardriving database such as WiGLE.net [281], and perform a targeted attack against a specific BSSID.

Beacon injection

Section 5.4.6 gave an example of how our frame injection attack can be applied if the attacker has access to a service on the internal wireless network. However, in practice this scenario is not very prevalent. It would therefore be more interesting for an attacker to set up their own service, and lure the victim into accessing it. As an example, the attacker can set up a web server and serve a binary file containing malicious frames, which would be automatically downloaded by anyone who visits the web page. As with our previous attack, these frames will occasionally be interpreted by the victim due to interference on the wireless link.

The above scenario was tested using the same experimental setup from Figure 5.4.1. We set up a web server on the attacker's machine and created a `jpg` image containing `Beacon` frames with a specific SSID as the A-MPDU subframe payload, though any other type of payload could have been used. Despite the fact that sending management frames inside an A-MPDU is not standard compliant [127], the frames were accepted by the victim machine upon injection. A packet trace of this experiment is available at [197]. The trace shows a first injected beacon frame after 47 seconds and 16 MB of transmitted data.

Contrary to our previous attack from Section 5.4.6, the attacker would need no prior knowledge of the AP's MAC address, since for Beacon frames the DA is always `ff:ff:ff:ff:ff:ff` and the TA can be spoofed. The impact of such attack can range from rather harmless, such as displaying a message in a remote victim's SSID list, to injecting a beacon frame with a malformed SSID. Such malformed frames could trigger a buffer overflow vulnerability on the receiving host[4].

[4] An example is the heap-based buffer-overflow vulnerability found in NetGear WG311v1 wireless devices, which allows attackers to execute arbitrary code in kernel mode on the host [148].

5.4.7 Success rate

To determine whether our attack would be feasible in practice, we would like to calculate the probability of successful injection. An injection is considered successful when at least one subframe crafted by the attacker is received without errors by our victim. Here we present both an analytical approximation and experimental measurement of this probability.

Analytical approximation

Observe that the event of a successful injection occurs when any previous A-MPDU subframe delimiter of the aggregate becomes corrupted due to interference, as this will trigger the A-MPDU parsing algorithm. We assume that both the probability of aggregation p_a and probability of a single frame corruption p_c are known constants. After all, a significant amount of related work for analytically determining the probability p_c already exists [26, 51, 153, 288], which is outside the scope of this thesis.

Both of the input variables p_c and p_a can be modeled using Bernouilli trials. The probability that a received frame is corrupted by interference can be written as $p_c = 1 - q_c$, where q_c is the probability that a single frame is received correctly. If the frame is part of an aggregate, and becomes corrupted after transmission, we can distinguish between three cases:

1. The PLCP header of the A-MPDU was corrupted. In this case the entire A-MPDU would be dropped and the receiver would interpret neither legitimate nor injected frames.

2. Any of the A-MPDU delimiter bytes were corrupted. This means that the `Length` field of the current subframe will be ignored, and that the A-MPDU parsing algorithm from Listing 5.1 will be performed to search for the next valid A-MPDU delimiter.

3. Part of the subframe MPDU was corrupted. Here, the subframe in question will be passed to the MAC layer, where it will be dropped because of an incorrect FCS.

Let us now assume for the sake of clarity that each byte in the A-MPDU of length L_a has the same probability of becoming corrupted. Given a corrupted

frame and that the PLCP header is 6 bytes in length, the probability that the corruption occurred in the PLCP header is $\frac{6}{L_a}$. By applying Bayes' rule we get $p_c \cdot \frac{6}{L_a}$ as the unconditional probability. We would like to express this probability per subframe instead of per aggregate, so the probability should be divided by the number of subframes. This results in the following final probability $\frac{p_c \cdot \frac{6}{L_a}}{L_a/L_s}$.

In our second case, for any corrupt subframe of length L_s where $4 \leq L_s \leq L_a$, the probability that the A-MPDU subframe delimiter is corrupted is $\frac{4}{L_s}$. Analogous to our previous case, the unconditional probability becomes $p_c \cdot \frac{4}{L_s}$. For an entire aggregate, the probability that at least one delimiter becomes corrupted is $p_c \cdot \frac{L_a}{L_s} \cdot \frac{4}{L_s}$.

Finally, we do not need to consider the third case, since it does not matter whether part of the MPDU is corrupted or not: corruption of the A-MPDU delimiter is sufficient to trigger the subframe delimiter signature scanning algorithm and consequently, inject a subframe. For the victim to interpret the payload provided by an attacker, the injected subframe itself must not be corrupted, which has the probability $1 - p_c$.

Putting everything together, the probability of injecting a frame p_i and the number of successful injections per A-MPDU n_i become:

$$p_i \approx p_a \cdot (p_c \cdot \frac{4}{L_s}) \cdot (1 - \frac{p_c \cdot \frac{6}{L_a}}{L_a/L_s}) \cdot (1 - p_c) \tag{5.1}$$

$$n_i \approx p_a \cdot (p_c \cdot \frac{4L_a}{L_s^2}) \cdot (1 - \frac{p_c \cdot \frac{6}{L_a}}{L_a/L_s}) \cdot (1 - p_c) \tag{5.2}$$

A network administrator can use this model to approximate the success rate of our attack on their network. The accuracy of the model depends on the choice of the parameters p_a and p_c. As an example, these input variables will be approximated for our experimental setup in the following section.

Experimental measurement

To experimentally determine the success rate for our setup, we repeatedly performed our Beacon frame injection attack from Section 5.4.6 and compared the number of successful injections with the number of acknowledged A-MPDUs.

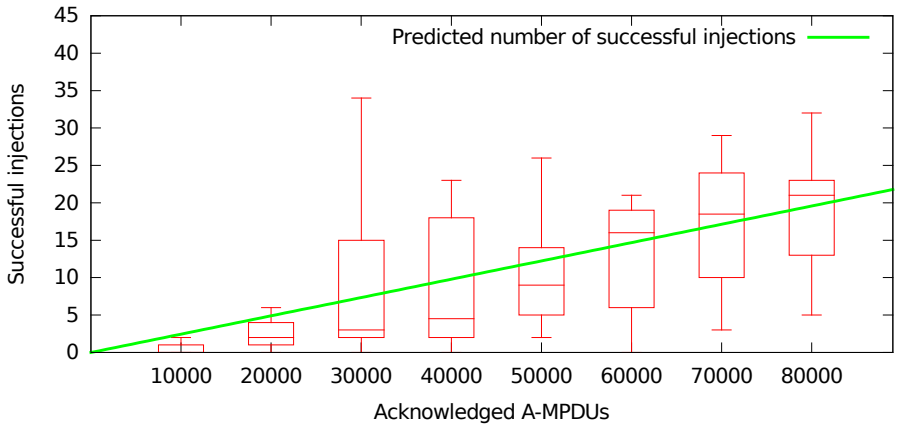


Figure 5.4.4: Repeated measurements of the malicious download attack with different file sizes in our experimental setup show that the number of successful injections versus the number of A-MPDUs acknowledged by the receiver is overestimated by our analytical model when using the parameters $p_c = 0.08$, $p_a = 0.03$, $L_a = 65535$, and $L_s = 1538$, as indicated by the green line.

Figure 5.4.4 shows the results of these measurements for 285 malicious `jpg` file downloads of sizes between 30 MB and 500 MB via the MikroTik AP.

In our analytical approach from Section 5.4.7, we defined the frame aggregation and corruption probability as two input variables for our model. If we would like to predict the number of successful injections in a realistic environment, these probabilities need to be measured, which is not trivial as they can vary significantly between different trials of the same experiment.

As an example, Figure 5.4.5 shows the variations of the aggregation rate over different trials, for 300 MB of transmitted data frames via the MikroTik AP. It is this variation in the aggregation rate which "smears out" the x-axis of our measurement of the number of successful injections from Figure 5.4.4. Thus, for a fixed number of transmitted data frames, the number of received A-MPDUs can vary greatly between different measurements.

The second input variable, the frame corruption probability, naturally varies between different trials due to external factors such as contending stations and other radio sources. Instances of poor channel conditions can lead to spikes in the number of successful injections, such as the outliers that can be seen in the

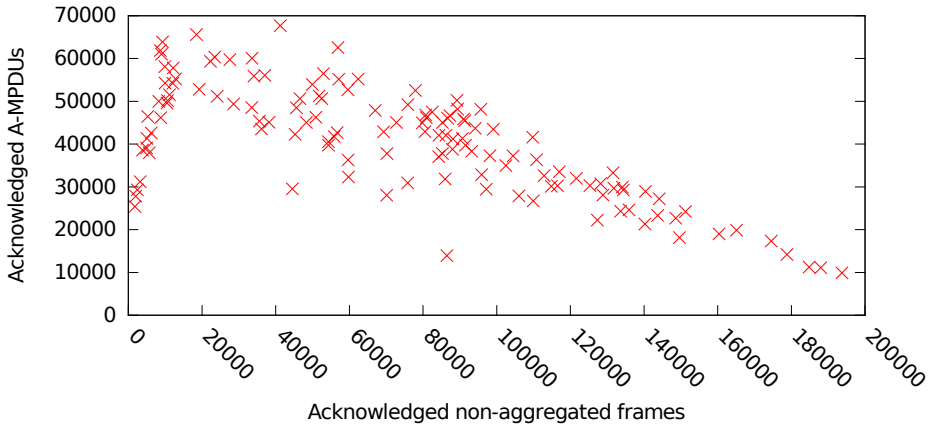


Figure 5.4.5: Experimental measurement of the number of acknowledged A-MPDUs versus the number of acknowledged regular frames during a malicious download attack. Each point in the data set represents a download of a 300 MB malicious `jpg` file.

boxplot of Figure 5.4.4 for 30,000 acknowledged A-MPDUs on the x-axis.

Despite the aforementioned random fluctuations, we can see that the median number of successful injections increases with the number of received A-MPDUs as expected. If we would like to predict the number of successful injections per A-MPDU, our analytical approximation can be used to fit the experimental data.

Suppose we want to estimate the number of A-MPDUs to transmit in order to have at least one successful injection in our experimental setup, given $L_a = 65535$ and $L_s = 1538$. The first step is then to determine the input variables for our model, p_c and p_a.

For determining p_c, it is easier to calculate $1 - q_c$ instead, where q_c is the probability of receiving a packet *without* errors. This probability can be determined by measuring the PDR at the receiver for a given time t. Previous work by Vlavianos et al. has shown that this is a reliable metric to measure the link quality of a wireless network [270]. De Couto et al. define this metric as:

$$\mathrm{PDR}(t) = \frac{\mathrm{count}(t - w, w)}{w/\tau} \tag{5.3}$$

where count$(t - w, w)$ is the number of correctly received frames, excluding retransmissions, at the receiver and w/τ the total number of transmitted packets [72].

The PDR depends on several variables such as the amount of interference, transmit power of the sender, distance to the receiver, the data rate, and the frame size. Since the attacker only has control over the frame size, we would like to measure the impact of this variable on the PDR.

Figure 5.4.6a shows the mean PDR for 50 repeated measurements per given frame size, with 50,000 packets sent to the receiver in each iteration and a distance of 1 meter between the AP (MikroTik) and the receiver.

From previous work, we know that decreasing the packet size increases the PDR [270]. In practice, the results can vary between measurements. For example, observe in Figure 5.4.6a that the measurements with a frame size of 1534 bytes have about the same mean PDR as the measurements with a frame size of 42. Such variations can be caused by the rate controller of the AP, which lowers the data rate in case of a suboptimal PDR. Figure 5.4.6b shows the impact of the rate controller. Here, the mean PDR was calculated based on 50 repeated measurements with 10,000 frames of size 1534, transmitted at a fixed data rate. Indeed, the PDR is higher when the data rate is low.

Since the mean PDR was equal to 0.92 with the rate controller enabled during our attack, we will use the value 0.08 as p_c in our model. Next, we need to determine the aggregation probability p_a. We measured that this probability is 0.03 for $L_s = 1534$.

We now have all required inputs for our model. For an A-MPDU length of 65535 bytes, a subframe length of 1538 bytes including padding and A-MPDU delimiter, aggregation probability of 0.03 and PDR of 0.92, $n_i \approx 0.00025$. This means that on average, one injection will be successful per 4065 transmitted A-MPDUs. The prediction is plotted as a green line in Figure 5.4.4, and appears to slightly overestimate the actual measured number of successful injections; given our assumptions and selected parameters.

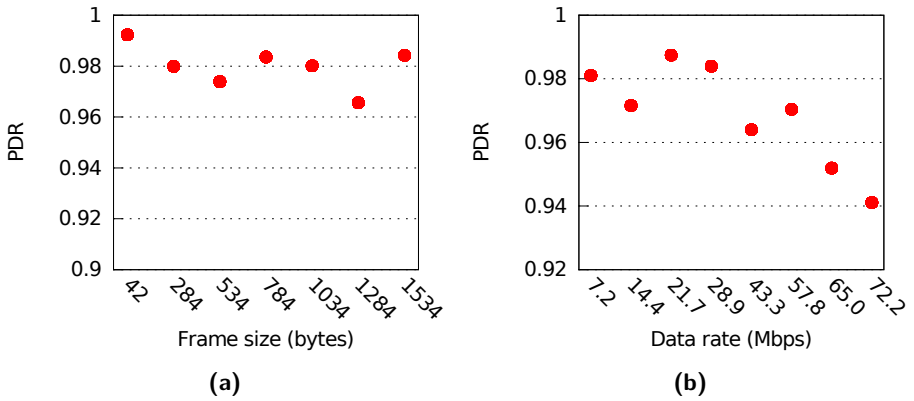


Figure 5.4.6: The mean PDR for our test setup generally decreases as the frame size increases (a), but displays some variations. Such variations can be introduced by the rate controller, which can improve the PDR by lowering the data rate (b).

5.5 Defensive measures

We have demonstrated in Section 5.4.6 how an attacker can use our injection methodology to perform several attacks. We will now suggest a number of defensive measures that network administrators and vendors can put in place in order to mitigate these kinds of injection attacks. Since not all of these measures are equally feasible, we will provide a comparison as well in Section 5.5.7.

5.5.1 Encryption

A simple solution for defending against our injection attack is to make use of MAC layer encryption such as WPA2-AES. This method does not require modifications to the firmware, driver or 802.11n standard, and is easy to implement.

When both encryption and A-MPDU aggregation are used, each MPDU of the aggregate frame will be encrypted separately. Therefore, if A-MPDU delimiter corruption occurs, the delimiter signature scanning algorithm will only see the encrypted payload, which is evidently not equal to the payload created by the attacker. Note that even if the attacker knows the master encryption key, it will not be possible to inject frames from a remote location, since a unique key pair is derived from the master key for each individual session. This session key is not

known to the attacker.

5.5.2 Disable A-MPDU frame aggregation

Another effective defensive measure that can be implemented by network administrators is disabling A-MPDU frame aggregation. However, there are several disadvantages that should be considered. Firstly, connected clients can no longer benefit from the increased data rate provided by A-MPDU frame aggregation. A-MSDU aggregation could be used as a less efficient [101, 248] alternative.

A second disadvantage is that, depending on the device, this method may require modifications to the firmware or driver of the NIC. For example, on Linux systems the `mac80211` protocol driver needs to be modified to clear the `IEEE80211_HW_AMPDU_AGGREGATION` flag. Performing such changes on all APs of a network is practically infeasible, especially if the drivers or firmware are proprietary.

5.5.3 Drop corrupted A-MPDUs

Some chipsets, such as the AR9271, set a certain register flag when a subframe delimiter error has occurred. The firmware or device driver can be modified to drop the entire A-MPDU in case this flag is set, similar to how an A-MSDU would be dropped in case of an error. However, this would cause an impact on performance that depends on the probability that an error will occur in any A-MPDU delimiter.

5.5.4 LangSec stacks

As mentioned by [103], LangSec stacks can provide a robust defence against PIP attacks. These network stacks essentially use formal language theoretic methods for input validation. Previous research by Sassaman et al. [227] shows that by treating inputs to network protocol stacks as simple to parse input languages, the security can be improved or even guaranteed. An example that is frequently referred to in this research is Structured Query Language (SQL) injection, where treating the user input as executable code can completely mitigate injection attacks.

In context of LangSec, we use the notation M for a message, $\mathcal{D}$ for a decoding function, and $\mathcal{E}$ for an encoding function. The 802.11n aggregation mech-

anism can now be described as $\mathcal{D}(\mathcal{E}(M))$, where M is a list of MPDUs, $\mathcal{E}$ is the aggregation process or the addition of the A-MPDU delimiters, and $\mathcal{D}$ is the deaggregation process or the removal of the A-MPDU delimiters. Naturally, the intended behaviour by the designers of the frame aggregation mechanism is that $\mathcal{D}(\mathcal{E}(M)) = M$, or in other words that list of MPDUs is not altered after the aggregation process. However, the introduction of random noise on the wireless channel then gives us the probability that $\mathcal{E}(M)$ is altered and hence, that $\mathcal{D}(\mathcal{E}(M)) \neq M$ which introduces a frame injection vulnerability as we saw earlier.

From a language theoretic point of view, an effective defensive measure against our injection method would be to design a recognizer that can parse a frame unambiguously with minimal computational overhead. For MAC frames sent over a wireless link, this is harder to accomplish compared to SQL query input, because if regarded a language, a frame has a much more complex grammar than a SQL query. Furthermore, SQL query input is usually already protected from transmission errors by underlying layers such as TCP. Conversely, MAC frames are sent over unreliable channels. In case of corruption, a frame header or delimiter can become indistinguishable from the frame data, since the same modulation and encoding is used for the entire aggregate, and since boundaries are defined by `Length` fields which may become corrupted as well. A valid MPDU M_2 can then exist so that $M_2 = \mathcal{E}(M_1)$, and consequently, $\mathcal{D}(M_2)$ becomes a valid operation.

One solution to this problem comes in the form of an encoding technique that facilitates unambiguous encapsulation, which was proposed by Ossmann et al. This technique, named Isolated Complementary Binary Linear Block Codes (ICBLBC), uses codes of a certain Hamming distance, for example (5,2,2) codes, with an additional unique "isolation" property. Such codes can be divided into two groups, where each codeword in one group is isolated from any codeword of the other group by a Hamming distance of 3. The first group of codewords can then be used for the header, and the second for the payload. In this example, the Hamming distance between the two groups ensures that up to two bit errors can exist without breaking the isolation between header and payload [103, 184, 185]. Though we believe this method is an effective defence, it is difficult to implement, as the hardware of the wireless NICs needs to be modified to implement the new coding scheme. Such modifications also need to be uniformly implemented by all 802.11 devices, and therefore a standard amendment is required as well. Finally, the overhead of the extra isolation bit in the code would lead to a slight impact on the performance.

5.5.5 Modulation switch

A different technique that can be used to achieve isolation between header and payload is switching between modulation schemes. Similar to how the modulation scheme switches from DBPSK in the PLCP preamble to Quadrature Amplitude Modulation (QAM) in the MPDU, the A-MPDU delimiter could be transmitted using a different modulation scheme than the subframe itself. Given that no symbols from the header modulation scheme can be created by using the payload modulation scheme, it will be impossible to inject payload data that can be interpreted as a header.

This approach is the least practical, since it would require changes in the standard and costly hardware modifications. Additionally, backwards compatibility with existing devices would be lost.

5.5.6 Deep packet inspection

Since a large number of packets typically need to be transmitted by the attacker before injection can be successful, a spike in network activity might be flagged by an Intrusion Detection System (IDS) as unusual, and the attacker could be blocked consequently. However, in our experience, transmitting a large number of frames is not *always* necessary for the attack to succeed. Moreover, in some networks, high loads of traffic might be commonplace.

A more effective approach would be to perform deep packet inspection. The IDS can look for payloads that resemble 802.11 headers and drop those packets. Disadvantages of this method are that expensive hardware is often required, and that the processing required to validate packets could introduce latency.

5.5.7 Comparison

Table 5.5.1 summarizes and compares all of the proposed defensive measures in terms of advantages. In this table, the filled portion of a circle denotes the presence of the advantage. A transparent circle means that the advantage is absent.

From the table we can derive that using encryption is the simplest and most effective measure. The only disadvantage is that users will have to provide some

Table 5.5.1: Comparison between the proposed defensive measures.

	Standard compliant	No hw. mod. required	No driver or fw. mod. required	Negligible throughput loss	Low cost	Supports open networks	Backwards compatible
Encryption	●	●	●	●	●	○	●
Disable A-MPDU aggregation	●	●	○	○	●	●	●
Drop corrupted A-MPDUs	●	●	○	○	●	●	●
Langsec (ICBLBCs)	○	○	●	●	◗	●	○
Modulation switch	○	○	●	●	○	●	○
Deep packet inspection	●	●	●	●	○	●	●

type of credential, for example a username / password combination or secret key, before the network can be joined. It should be mentioned however, that this disadvantage can be removed by deploying a technology such as Wi-Fi Passpoint [275].

5.6 Chapter conclusions

We identified a vulnerability in the A-MPDU frame aggregation mechanism of the 802.11n standard (**RG1**), which allows information to leak from higher layers of the Open Systems Interconnection (OSI) stack to the MAC layer. Based on this vulnerability, we demonstrated a novel and practical frame injection attack that can be remotely performed against open Wi-Fi networks which support 802.11n (**RG2**), yielding **C2**. Additionally, we have shown two example attack scenarios using our injection method. We then demonstrated that the success rate of this attack depends on the frame corruption probability or link quality of the target network, and on the probability that a frame is aggregated between the last hop and the victim host. We have experimentally determined that for maximizing the probability that a frame becomes corrupted, large frames should be transmitted

at a high data rate, whereas for maximizing the aggregation probability, the ideal frame size depends on the device model. Finally, with regard to **RG3**, several defensive measures that can be applied to mitigate the attack were described and compared.

6

Non-cooperative 802.11 MAC-layer Fingerprinting and Tracking of Mobile Devices

6.1 Introduction

Techniques for tracking the whereabouts of IEEE 802.11 standard-compliant mobile devices can be employed for several benign use cases, such as positioning [135, 155, 199, 283], asset tracking [82, 289], travel time estimation [5] and behavioral modeling [33]. These use cases inspired the creation of numerous startups and commercial products such as Skyhook[1], Wifarer[2], YFind and Nomi[3], as well as a substantial amount of academic research. Wi-Fi based tracking is furthermore often chosen in favor of other technologies because of low battery consumption, low cost, passive monitoring capabilities of personal devices, and widespread presence of Wi-Fi chipsets, which is likely to further increase due to the current IoT trend. However, tracking systems can unfortunately also be used *maliciously* to non-cooperatively or involuntarily disclose the location of mobile device users to an adversary, compromising their privacy.

From the perspective of the tracked device, location tracking systems can be divided into two categories: cooperative and non-cooperative location tracking systems. In cooperative tracking, the mobile device is aware that it is being tracked, and actively cooperates with the location tracking system to determine its correct location. In Android phones for example, when the "high accuracy" location mode is enabled, the device will determine its own location by combining information from nearby Wi-Fi APs, cellular networks, or the Global Positioning System (GPS) chipset.

On the other hand, in non-cooperative tracking systems, the mobile device is unaware of the fact that it is being tracked, no existing infrastructure is modified, and the mobile device does not cooperate with the location tracking system[4]. Instead, the location tracking system relies on information leaks from radio transmissions captured by one or more Monitoring Stations (MSs) to track the location of a mobile device, thereby creating the opportunity for malicious exploitation by an adversary. A remote positioning system topology as described by Liu et al. [155] can for example be used for this purpose [5, 33, 62, 176].

In this work, we will study two aspects of non-cooperative tracking: how mobile devices can be *fingerprinted* and *deanonymized* by a MS based on information

[1] `http://www.skyhookwireless.com/`

[2] `http://www.wifarer.com/`

[3] `http://www.nomi.com/`

[4] This type of tracking is referred to as infrastructure-less terminal-based Location Fingerprinting (LF) in Kjærgaard's taxonomy of location tracking technologies [134], or as "involuntary tracking" or "non-collaborative tracking" in other works.

leaked from a single observation of a Wi-Fi frame, and how their *transmission frequency* can be artificially increased so that the number of observations by the MS is increased. We demonstrate a variety of techniques that exploit protocol design flaws and chipset implementation vulnerabilities to achieve these goals. Furthermore, we focus on tracking systems that operate on the data link layer (specifically, the MAC sublayer). These systems have the advantage of being deployable in software on commodity hardware as opposed to PHY layer based tracking (see [71] and the references therein), where specialized equipment is needed. For higher layer (e.g. transport or application layer) based tracking, the device that is being tracked must be associated to a Basic Service Set (BSS), and the frame payload must be transmitted in plain text. This was rarely the case anymore at the time we conducted this study, as according to WiGLE only 7.95% of wireless networks did not use encryption [280].

The structure of this chapter is as follows. Section 6.2 entails general background information about the principles and topology used in non-cooperative tracking. The role of fingerprinting in non-cooperative Wi-Fi tracking is discussed as well. In Section 6.3, we examine the types of device identifiers present on the PHY and MAC layer. Furthermore, we introduce a novel technique that can be used to defeat MAC address randomization based on a single frame observation from the transmitting device (**RG1**). Section 6.4 will then discuss both novel and existing techniques for increasing the frame transmission frequency from tracked devices. The effectiveness of the discussed techniques is evaluated and compared against methodologies from earlier works in Section 6.5 (**RG2**). A discussion of which countermeasures should be implemented by vendors is detailed in Section 6.6 (**RG3**). In Section 6.7, we will discuss related work, and finally, the conclusions are discussed in Section 6.8.

Our contributions

Considering research goals **RG1** and **RG2**, our first contribution is an entropy-based metric for measuring information leakage in 802.11 MAC-layer frame bits, specifically of the IE bits contained within `Probe Request` frames (**C3**). We then use this metric in a detailed analysis of 200,394 `Probe Requests` to identify the frame bits that are the most useful for constructing a per-device fingerprint (**C4**), and quantify the uniqueness of this fingerprint. To the best of our knowledge, this approach has not been considered before in previous works. Next, we demonstrate how an adversary can exploit this information leakage in practice, in order to track users involuntarily without changing the existing wireless network infrastructure, while using only commodity hardware. We further demon-

strate how this technique can be used to defeat MAC address randomization and deanonymize users.

Following our per-bit analysis, a second novel technique that can be applied to instigate extra transmissions from a tracked device is detailed (**C5**). More specifically, we show that "Hotspot 2.0" devices supporting the 802.11u amendment's Generic Advertisement Service (GAS) or devices that incorrectly handle 802.11e Block Acknowledgement (BA) frames can be actively probed, so that their presence is unwittingly exposed to a tracking system.

The aforementioned techniques are experimentally evaluated on two datasets, of which the first was recorded at Glimps 2015, a Belgian music festival in the city of Ghent, and the other at our research lab. Both datasets and the code are made available to the public domain in an anonymized format such that the true identity of the devices or persons involved in the experiment is not revealed (**C10**). Finally, considering **RG3**, we also present several countermeasures against these non-cooperative tracking techniques in order to minimize information leakage and thus improve the privacy of the user.

6.2 Background

Recall that in non-cooperative tracking, the mobile device is unaware that it is being tracked, and does not actively assist the location tracking system in determining its location. In this case, a topology as shown in Figure 6.2.1 can be used.

Here, the tracking system consists of several MSs spread out over a number of known and fixed geographical locations. Since the locations of these MSs are known, radio signals transmitted by a target device can be captured by the MS and roughly mapped to the MS's location [33, 176, 191]. In this scenario, the accuracy of the exact location of the tracked device depends on several factors, such as the gain of the receiving MS's antenna, carrier frequency, the transmit power of the tracked device, and environmental factors. A high range results in a higher probability to detect a device's transmission, but will decrease the location accuracy and vice versa. If location accuracy is crucial for the application, other techniques such as Time Difference Of Arrival (TDOA) with multiple synchronized MSs could be used to improve the location accuracy [17, 102, 283].

The effectiveness of a non-cooperative remote positioning topology depends on

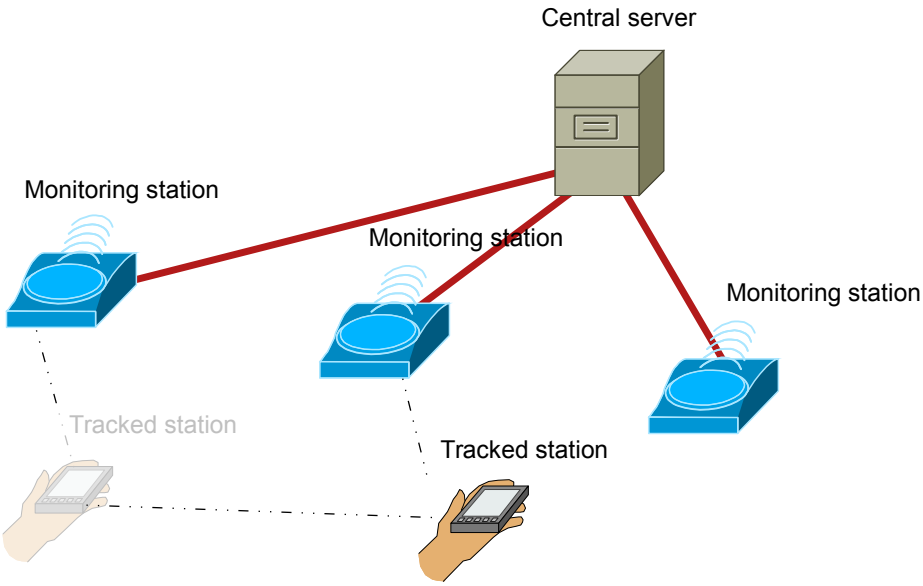


Figure 6.2.1: An example of a non-cooperative position tracking topology. Several low to medium range Wi-Fi MSs with known positions are connected to a central server via a Local Area Network (LAN) or the internet. The system can track a roaming target by intercepting signals transmitted by the device and mapping those detections to the location of the corresponding MS.

three criteria: a monitored device must be *uniquely identifiable*, its identifier must *remain stable over time*, and the identifier must be *transmitted at a minimal frequency*, preferably as often as possible.

Firstly, the unique identification requirement follows from the fact that the location tracking system must be able to differentiate between multiple observed devices. Here we can make the distinction between globally unique identifiers, which are permanently unique versus local identifiers, which are only unique for a limited duration or session. In the literature, the study of uniquely identifying a device is commonly referred to as "fingerprinting". We will use this terminology from now on. In order to create a unique fingerprint, MSs must analyze externally observable properties from frames that are transmitted by the target device. When these properties are intended or designed to uniquely identify a device, they are called *explicit identifiers*. An example of an explicit identifier is the MAC address. On the other hand, if the ability to fingerprint a device from

this property is unintentional, it is an *implicit identifier*[5] [191]. Identifiers are present in many shapes and forms on both the PHY and MAC layers of 802.11 frames, as will be shown in Section 6.3.

Secondly, stability of the identifier over time means that the identifier must remain either identical or correlatable to previous values for a sufficiently long duration. Indeed, the same device identifier must be detected by at least two different MSs before assumptions about the path traversed by the device can be made. A globally unique identifier for example is always stable.

Finally, the identifier must be transmitted as often as possible in order to increase the probability of detection. This is especially important if the target device is moving fast, since the device might move out of range even before it had the chance to transmit its identifier. We will henceforth refer to this property as the transmission frequency of the device.

6.3 Identifiers and fingerprinting

Identifiers are important in fulfilling the requirement that a MS must be able to differentiate between multiple observed devices. Moreover, when explicit identifiers are unavailable (e.g. because of MAC address randomization), the MS can only rely on implicit information to construct a fingerprint for the observed devices. If these fingerprints are unique to such a degree that frames transmitted by a particular device can be linked together without explicit identifiers, the device is said to be deanonymized. In this section, we will explore earlier approaches to PHY and MAC layer fingerprinting. In Section 6.3.3, we will discuss our own methodology for constructing a fingerprint of a device without using explicit identifiers. Later, in Section 6.5.2, we will see how this approach can be used to deanonymize devices in practice.

6.3.1 PHY-layer fingerprinting

Identifying a device based on PHY-layer properties typically relies on the analysis of either a raw radio signal transmitted by the device or the baseband signal (e.g. for Wi-Fi, the PLCP data). Unfortunately for the tracker, these PHY-layer

[5]Note however that despite the term "implicit", these identifiers are still a form of *explicit* information leakage according to our definition, since they are not the result of inherently measurable side effects of a (hardware) implementation.

properties cannot be easily extracted from a frame, since in most conventional chipsets the PHY layer is implemented in hardware for performance reasons. Therefore, this type of data is typically not accessible via the driver or firmware of the chipset. A signal analyzer or SDR can be used instead, in order to capture the raw radio signals and demodulate them in software [38]. Although this approach can yield highly unique fingerprints (depending on the measurement capabilities of the tracker), the required hardware is typically more expensive, requires more compute power compared to using off-the-shelf devices, is more sensitive to channel noise and interference, and requires more complex pattern matching algorithms to derive a meaningful fingerprint from the radio signal. A more detailed description of how a device can be fingerprinted using PHY-layer properties will be given in Chapter 8, where we discuss a machine learning based approach to fingerprint LoRa signals captured by an SDR. Other approaches have been proposed several earlier works, where we can distinguish a variety of PHY-layer properties from which a fingerprint can be derived:

Time domain analysis In a time domain analysis of a transmitted signal, devices are classified using differences between the amplitude, phase or frequency of the wave transients. This technique can identify a device given that its fingerprint is known to the classifier [115, 263].

Frequency domain analysis Brik et al. compare small frame imperfections in the modulation domain with an ideal PHY frame modulation. However, a Support Vector Machine (SVM) classifier must be trained to recognize the device beforehand, using a set of 20 frame – MAC address pairs [38]. In the work presented by Corbett et al., a Power Spectral Density (PSD) analysis is performed, which captures the power or spectral density that a signal has over a range of frequencies. The PSD is calculated for each of the devices to be fingerprinted [61].

Clock skew Inherent drifts in the hardware clock of a device, caused by variations in the manufacturing process, can be measured and then utilized as a fingerprint [10, 130, 139]. However, in context of 802.11 this technique relies on timestamps extracted from `Beacon` and `Probe Response` frames, which are only transmitted by APs.

Scrambler seed The 802.11 scrambler is a Pseudo-Random Number Generator (PRNG), implemented as a 7-bit Linear-Feedback Shift Register (LFSR), that is XORed with the frame payload in order to obtain a uniform distribution of bit values. It is used in Orthogonal Frequency-Division Multiplexing (OFDM) modulation to improve the Peak-to-Average Power Ratio (PAPR), and in turn to reduce the packet error rate. Bloessl et al. and

Vanhoef et al. show that predictable scrambler states can be used to track mobile devices [29, 268]. However, this assumes that the scrambler seed is variable[6], and that the chipset vendor has implemented the scrambler so that it behaves in a deterministic, predictable manner. Furthermore, if the tracked device cannot be observed continuously, linking the scrambler states of two transmissions originating from the same device becomes increasingly difficult.

Observe that for each of these PHY-based approaches, the implicit identifiers of a device need to be determined in some preprocessing step [71], before the device can be uniquely detected reliably at a later stage. Furthermore, multiple observations are often required before a device can be correctly classified. These properties render PHY-based approaches impractical for non-cooperatively tracking devices at a large scale.

6.3.2 MAC layer fingerprinting

In contrast to PHY layer fingerprinting, MAC layer fingerprinting does not require raw radio signals to be analyzed in order to uniquely identify a device. Instead, one can use off-the-shelf Wi-Fi hardware such as a USB dongle to process the radio signal and analyze the resulting MAC layer data. For this reason, MAC layer tracking is particularly attractive to the industry, which according to Kjærgaard et al. desires systems that "have low maintenance, allowing positioning of all user devices, regardless of platform and form factor" [135]. The MSs used in MAC layer tracking systems are typically configured in *monitor mode* so that frames are forwarded to user space regardless of their destination MAC, and a Radiotap header with frame metadata is prepended to the packet.

MAC address

A common identifier for fingerprinting a device on the MAC layer is the MAC address, which is disclosed by devices on a regular basis via transmitted frames, even when the device is not associated to an AP. Examples are `Probe Request` frames, which are transmitted by a client device in order to exchange IEs with the AP and detect their presence. The advantage of tracking a device based on the MAC address is that it is inherently a globally unique identifier per NIC,

[6] This is not the case when Direct-Sequence Spread Spectrum (DSSS) modulation is used, e.g. when transmitting Probe Requests in the 2.4 GHz band [268].

which explains why it is so commonly used in existing tracking systems [5, 33, 34, 62, 63, 176, 191].

However, tracking systems that solely rely on the MAC address as an identifier can be easily thwarted by employing MAC address randomization. With this feature, the device temporarily sets a random, locally unique MAC address before the active scanning procedure (i.e., before transmitting `Probe Requests`) [110]. This may be implemented by the device vendor in several ways, since the procedure is not explicitly stated in the 802.11 standard [127, p. 980]. For example, the MAC address can be randomized for each `Probe Request`, or only once before association with an AP. Moreover, some implementations randomize the entire MAC address, whereas others keep the OUI part of the MAC address identical[7]. Random MACs should have their "locally administered address" bit set to one [127]. After the scanning procedure, i.e. when the device is associating with an AP, the non-random MAC is typically reused in order to prevent MAC address collisions or network disruptions when roaming [110].

MAC header information

Despite the fact that MAC address randomization mitigates the issue that the MAC address can be used as a unique identifier, some devices can still be uniquely identified even when MAC address randomization is enabled. This can be accomplished by exploiting implementation quirks, for example by correlating the 802.11 frame `Sequence Number` [47, 112, 268] or `Duration` field [40] if these fields are not randomized.

Several other fields and properties from the MAC frame header have been proposed in previous works as implicit identifiers, such as combinations of the frame size [178, 191], "`more fragments`", "`retry`", "`power management`" and "`order`" bits in the header, the authentication algorithms offered, and the used transmission rates [191]. However, the per-bit uniqueness and stability of these identifiers in context of devices in the unassociated state has not been studied before. In Section 6.3.3, we will demonstrate our methodology for determining the suitability of an identifier in an automated fashion.

[7] In `wpa_supplicant`, the most popular 802.11 supplicant for Linux and the default supplicant for Android devices, this randomization behavior can be configured through the parameters `mac_addr`, `rand_addr_lifetime`, and `preassoc_mac_addr` [13].

Timing variations

Differences between implementations or hardware can cause slight timing variations that can in turn serve as useful implicit identifiers. For example, the Timing Synchronization Function (TSF) in `Beacon` frames from an AP can be used to measure the hardware clock skew, which is different for each device due to variations in the manufacturing process [10, 130]. However, this timing information cannot be extracted from non-AP stations (STAs), since they do not transmit the TSF.

For non-AP STAs, timing differences between the observations of `Probe Request` frames have been considered to differentiate between implementations [62, 76, 87, 168]. Even so, these timings are usually not unique per device, and they can be influenced by other factors such as whether the screen is on or whether the Wi-Fi chipset is in sleep mode [88]. A more generalised time-based fingerprinting approach where timing information is extracted from the Radiotap headers of the MS was proposed by Neumann et al. [178]. Here, the medium access time, transmission time and frame inter-arrival time are derived from the Radiotap header timing information in order to construct an implicit identifier.

Information elements (IEs)

IEs are TLV fields that are embedded in `Probe Requests` and `Beacon frames`. They contain information about the STA such as the supported data rates, SSID, capabilities, vendor-specific data, etc. and are therefore a plentiful source of implicit identifiers [268].

One example is the SSID IE, which is used in a `Probe Request` to indicate the SSID that is addressed. An SSID of length 0 can be used to indicate the wildcard (any) SSID [127, p. 478]. When a rare SSID, denominated in some works as a Personally Identifying Wireless Network (PIWN) [62], is observed or when multiple common SSIDs are observed in a particular order for a single device, the MS can build a fingerprint based on these observations [63, 191].

6.3.3 Per-bit MAC header analysis

In each of the approaches from Section 6.3.2, a relatively small number of bits is chosen as the uniquely identifying information, and the remainder of the frame is discarded. Instead of manually defining which bits are suitable for fingerprinting

a device and which are not, it would be interesting to *automatically* learn the suitability of a bit from a set of observations. We will now present our methodology, which aims to accomplish this goal.

In our methodology, a score is assigned to each bit of the MAC frame based on its potential as an identifier. The most suitable bits are then combined to construct a bitmask for each frame field. As such, new or rare fields used by only a handful of devices are automatically incorporated into the resulting fingerprint, independently of the header field order or underlying protocol semantics. If performance is critical, it is possible to perform the per-bit analysis on a small "training" subset, and utilize the resulting bitmask to fingerprint the remainder of the dataset (see Section 6.5.2).

Our approach originates from the observations that we made in Section 6.2, i.e. that any set of field bits from an 802.11 frame could potentially be exploited to form a unique identifier, as long as the bits differ sufficiently across devices to become locally unique (bit variability) and remain consistent or at least related[8] over the duration required by the tracking system (bit stability).

For the tracked device, we assume that it may randomize its MAC address for every transmitted frame, i.e. MAC address randomization is enabled and correctly implemented. Therefore, our analysis requires only a single frame from the tracked device in order to construct a fingerprint. Moreover, we assume that the tracked device is in the unassociated state, causing it to only receive and transmit Class 1 frames. This class of frames will be discussed in detail in Section 6.4.1.

Let us now define what a "uniquely identifying bit" entails exactly. We define three metrics that will henceforth be used in our analysis: the *variability* or uniqueness of a bit, the *stability* of a bit, and the *suitability* of a bit.

Calculating the variability

The variability of a bit is the entropy of that bit measured over multiple MAC frames, each of which is transmitted by a different device. Only one frame should be considered per device in order to prevent a frequently transmitting device from biasing the result, and only non-random MACs should be considered in this learning phase for the same reason. The goal of the variability metric is to indicate which bits provide unique contributions to the fingerprint.

[8]For example, when using the sequence number from 802.11 frames, the value will not be identical in each transmission but it will be related to the previously transmitted frame.

To calculate the variability of the bit at position i in a frame, we will first calculate the discrete probability density function $P(X_i)$ for each bit value at position i. Let X_i be the random variable that represents a bit value. Then $X_i \in \{0, 1, U\}$, where a bit value of U denotes the absence of a bit. The probability density function can subsequently be used to calculate the Shannon (information) entropy:

$$H(X_i) = -\sum_{x \in \{0,1,U\}} P(X_{ix}) \log_3 P(X_{ix}) \tag{6.1}$$

Note that we use a base 3 logarithm since we have a tri-state bit instead of a conventional bit. The tri-state bit is used to ensure that the absence of a bit is also counted as a different bit value. Now $H(X_i)$ will be a value between 0 and 1 where 0 indicates no entropy and 1 indicates maximum entropy. The result can be represented as a variability vector $\boldsymbol{v}$ per bit:

$$\boldsymbol{v} = [\ H_1\ \ H_2\ \ \cdots\ \ H_{n-1}\ \ H_n\ ] \tag{6.2}$$

where n is the number of bits in the frame and H_n is the entropy of bit n.

Calculating the stability

We define the stability of a bit as one minus the entropy of that bit measured over multiple MAC frames transmitted by *the same* device. The stability gives an indication of how likely it is that a bit will stay the same over multiple transmissions by the same device. The goal of the stability metric is to exclude those bits that are associated with the same device but change frequently.

The stability for a certain bit of the frame transmitted by a device is calculated analogous to Equation 6.1. The result can be represented as a stability vector $\boldsymbol{s}$ per device d:

$$\boldsymbol{s_d} = [\ 1 - H_{d1}\ \ 1 - H_{d2}\ \ \cdots\ \ 1 - H_{dn-1}\ \ 1 - H_{dn}\ ] \tag{6.3}$$

As opposed to the variability, a higher entropy is unfavorable as it means a lower bit stability. Hence, we measure the stability of bit i as $1 - H(X_i)$. Finally, recall that contrary to the variability, we calculated the stability *per device*. Hence, the stability vectors $\boldsymbol{s_d}$ for each device are averaged into a vector $\boldsymbol{s}$, resulting in the

final average stability vector per bit.

Combining variability and stability to form a fingerprint

Ideally, the bits used in our fingerprint should both be highly variable and highly stable. We will now discuss two possible approaches to combine these metrics into a new metric which we will define as the *suitability*, denoted as the vector $\boldsymbol{u}$. Both approaches can be used in practice, as we will discuss in the coming sections.

PROBABILISTIC APPROACH Since both the variability and stability are values in the interval $[0, 1]$, we can interpret them as probabilities of the bit being suitable for use in a fingerprint. Assuming that variability and stability are independent variables, we can then multiply the variability and stability to obtain the final suitability:

$$\boldsymbol{u} = \boldsymbol{v} \odot \boldsymbol{s} \tag{6.4}$$

For example, a bit with 1.0 variability and 1.0 stability is ideal and therefore has maximum suitability, whereas a bit with 1.0 variability and 0.0 stability is completely unsuitable for fingerprinting a device.

FILTERING APPROACH Considering that stability is a desired feature, another possibility is to interpret the bit variability itself as the suitability, on the condition that the stability is better than a user specified threshold λ.

$$\boldsymbol{u}_i = \begin{cases} \boldsymbol{v}_i, & \text{if } \boldsymbol{s}_i \geq \lambda \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

For example, if $\lambda = 1$, this approach ensures that the suitable bits that are combined in the fingerprint will always remain stable over time.

The threshold λ essentially determines the tradeoff between uniqueness of the fingerprint and stability of the fingerprint, which will be discussed in Section 6.3.3. Bits with a suitability greater than zero are used in a per-IE bitmask, which is

applied to each frame received from a device. The result of the mask operation is then concatenated and used as the final fingerprint.

Information Element analysis

In Section 6.3.2 we briefly mentioned that IEs are exchanged between the AP and client STAs via `Probe Requests` and `Probe Responses`. Due to the large quantity of potential implicit identifiers in IEs (e.g. in the supported rates, SSID, capabilities, vendor specific data, etc. IEs), it would be interesting to analyze which bits of which IEs are *the most* variable and stable. For that purpose, we can use the metrics discussed in the previous section.

An issue that should be considered beforehand is that although IE TLVs are usually transmitted in ascending order of the IE type, we have observed that some implementations transmit different orderings of IEs. If the order is disregarded, this would consequently cause the bit variability and stability to be derived from different IE types in some cases.

Therefore, to get an idea of exactly which bits per IE type contain uniquely identifying information, we need to compute their suitability *separately* per IE type. That is, we split the `Probe Request` into its IEs and perform our analysis separately for each IE type. Later, when constructing the fingerprint for a device, the order in which the IEs appeared should still be considered, because this order itself is a piece of uniquely identifying information [268]. We achieve this by introducing a "dummy" IE type to the fingerprinting algorithm, which contains a concatenation of the IE type bytes in the order in which they appeared.

To get an idea of which IEs are the most suitable to be used in a fingerprint, we have performed our bit entropy analysis using a dataset of 200,394 `Probe Requests` collected at our research lab. Table 6.3.1 shows the total variability and instability for each IE type. Figure 6.3.1(a) graphically shows the bit variability for the first 256 bits of each IE type. We can clearly see that the most variable bits come from the SSID IE and the `Vendor Specific` IE. For the SSID IE, also note that each first bit per byte of the SSID field has less entropy than the other bits. This is because all SSID strings that we observed were ASCII encoded. Other useful sources of uniquely identifying bits are the capability IEs: the capabilities of the device will be different depending on the used Wi-Fi chipset.

Figure 6.3.1(b) shows the bit *instability*[9] for the same number of bits. As ex-

[9]We chose to visualize the instability, since the majority of bits are stable, and plotting all

Table 6.3.1: An overview of the total bit entropy sum for each observed IE type. Note that the sum of bit entropies does not take correlations between bits into account. It should not be confused with the overall information entropy of the IE values.

Information Element	Σv_i	$\Sigma 1 - s_i$
AP Channel Report	0.000	0.000
DS Parameter set	0.625	0.411
Extended Capabilities	32.790	0.061
Extended Supported Rates	28.373	1.716
HT Capabilities	13.299	0.176
Information element order	40.327	2.529
Interworking	32.491	0.000
RSN Information	0.000	0.000
SSID parameter set	87.570	30.590
Supported Rates	22.529	1.317
VHT Capabilities	10.424	0.000
Vendor Specific	285.449	135.288

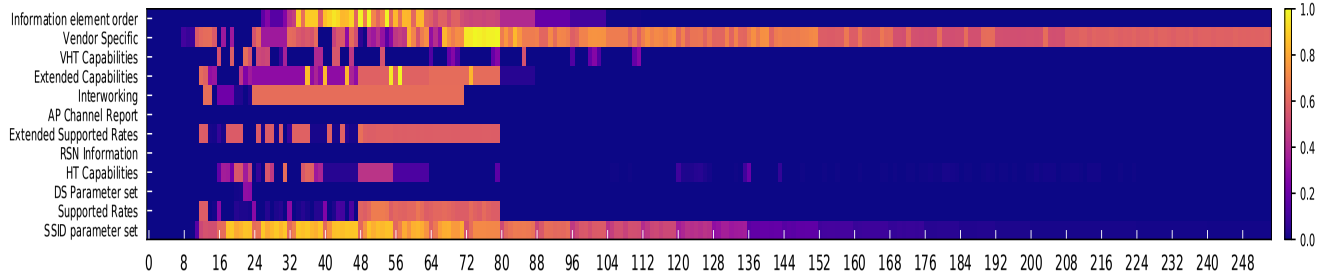
pected, the SSID IE is unstable because the SSID field in a `Probe Request` is often different for each individual probe. Likewise, a large portion of the `Vendor Specific` IE is very unstable, which means that these bits are unlikely to be useful in long term fingerprints. At the same time, some bits from the `Vendor Specific` IE are stable and highly variable, making them interesting uniquely identifying bits.

Note that some of the capability IEs have a very small amount of instability. This is an interesting result, because we would not expect capabilities of the Wi-Fi hardware to change. We *believe* that some vendors modify the announced capabilities of the device in order to enforce some kind of behavior from the AP (e.g. announcing support for low data rates only, in order to improve reliability or range at the expense of data rate).

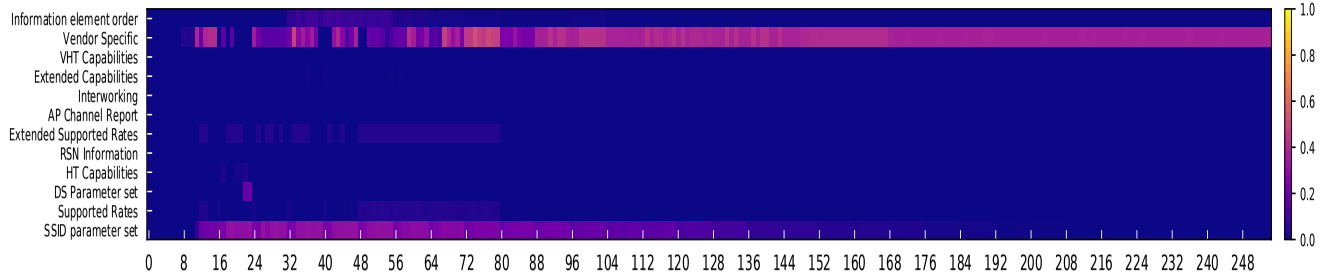
Discussion

The user defined threshold λ can be tweaked according to the needs of the location tracking system. If fingerprints that remain stable for an extended duration are

stable bits would hence clutter the heatmap.

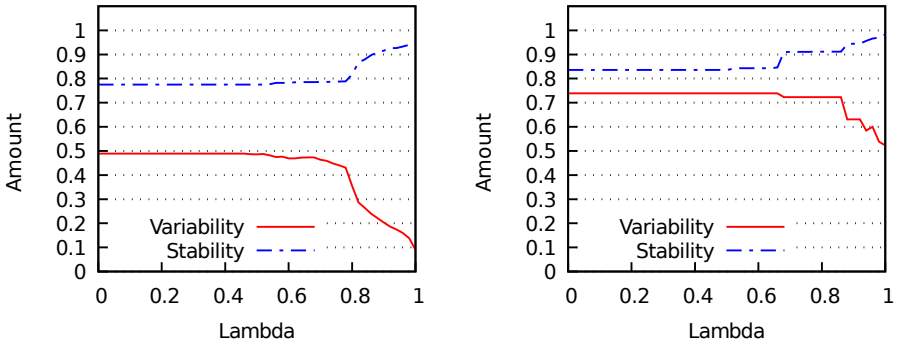


(a) Variability of IEs



(b) Instability of IEs

Figure 6.3.1: Graphical representation of the per-bit (x-axis) variability (a) and instability (b) for each observed `Information Element` (y-axis). Brighter colors indicate a higher entropy and thus a higher variability and instability.



(a) λ between $[0, 1]$ for a large test set of 10,000 devices

(b) λ between $[0, 1]$ for a small test set of 100 devices

Figure 6.3.2: Graphical representation of the tradeoff between variability and stability, which can be controlled via λ. Incorporating more stable bits in the fingerprint increases stability of the fingerprint at the expense of fingerprint variability.

required, λ can be increased. This will cause bits with a stability above the threshold to be exclusively incorporated into the fingerprint, trading stability for variability in the process. The tradeoff between variability and stability is depicted graphically in Figure 6.3.2.

Figure 6.3.2 also shows that even with $\lambda = 1$, the fingerprint stability is not perfect. This is because some devices transmit an additional IE (for example a `Vendor Specific` IE) on occasion, resulting in two different fingerprints for a single device. However, even in this case random MACs used by this device can still be mapped to the original MAC address if both fingerprints are associated to the original MAC address at some point in time. An algorithm for associating a random MAC to the original MAC will be discussed in Section 6.5.

One might argue that instead of an exact match of the fingerprint bits, a different metric such as the (weighted) Hamming distance or Jaccard similarity coefficient of the bits could be used. However, observe that these metrics are unsuitable for *stable* bits: a stable bit will never change for a specific device. Hence, a change in this bit indicates that a different device is observed, and therefore a different fingerprint should be generated.

6.4 Transmission frequency

For non-cooperative location tracking, it is desirable that the tracked device transmits radio signals as often as possible. After all, a MS can only fingerprint an observed device when this device is both in range and transmits information that can be used for identification. A device that is moving and infrequently transmits might be "missed" by a MS positioned at a certain location. Moreover, devices might be unassociated and in sleep mode, rarely transmitting frames.

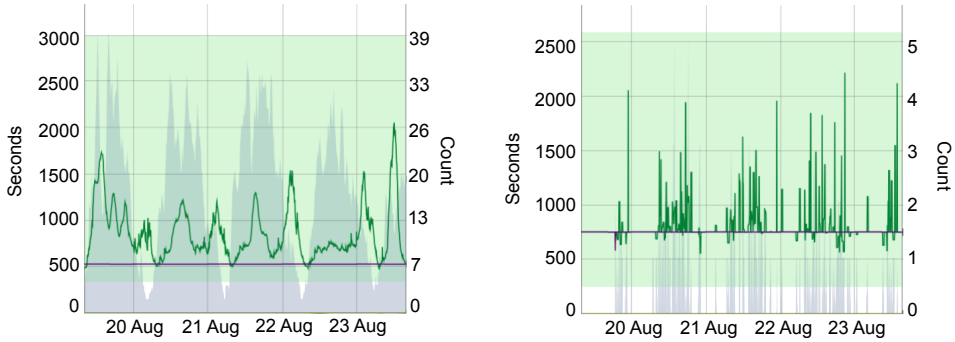
To further clarify the issue, consider the following example use case. In August 2015, we deployed a tracking system on the roads near Pukkelpop 2015, a popular music festival located in Kiewit, Belgium, in order to measure traffic congestion. Figure 6.4.1(a) shows the estimated travel time between two points on a road segment near the festival site. The travel time was determined by measuring the time difference between observations of identical mobile devices at the start and end points. Since vehicles were required to drive slowly on this segment, we were able to capture many mobile devices that were observed at both points, resulting in an accurate estimation of the true travel time. Fluctuations between day and night, and the increase in congestion at the start and end of the festival can be observed.

Figure 6.4.1(b) however, shows the same setup for a segment with a similar distance between the two points, but where the MSs are located near a highway entry and exit ramp. Here, cars were allowed to move faster. Since the transmission frequency of mobile devices located inside the cars remains identical, the probability of matching a device between the start and end points decreases, and so does the accuracy of the travel time estimation.

A possible solution for tracking such rapidly moving devices could be to increase the number of MSs on a road segment [176]. However, the cost of the location tracking system would increase, and the benefit would still be marginal if the device in question transmits its identifier infrequently.

6.4.1 Instigating transmissions

To solve the problem of infrequent transmissions by nearby devices, a tracking system can actively try to instigate transmissions from devices in the vicinity. Here, the MS can craft frames which exploit a certain protocol mechanism [176] or vendor-specific vulnerability as we will see in Section 6.5. As an additional



(a) Festival site road segment

(b) Highway ramp road segment

Figure 6.4.1: Travel time in seconds for a road segment (green line). The purple line shows the travel time in optimal conditions as calculated by Google Maps. The blue graph in the background shows the number of device matches that were involved in the travel time calculation.

benefit, the instigated response might further reveal details about the targeted device [36], which could serve as an implicit identifier. Thus, using this technique, the MS can increase the number of observed devices, increase the number of transmissions per device and generate more bits as input to a fingerprinting algorithm at the cost of actively transmitting frames and becoming detectable. Let us now determine which frames are allowed to be transmitted and received by a device while in an unassociated state, by investigating the 802.11 standard.

The 802.11 standard defines 4 different states that can exist between a pair of STAs. Here, each state defines a class of MAC layer frames that may be exchanged by the transmitting and receiving STAs, as shown in Figure 6.4.2. From the MS's point of view, the most interesting frames to consider are *Class 1* frames: these frames do not require authentication or association to an AP in order to be exchanged between peer STAs, and are almost always unencrypted[10]. A MS can utilize this class of frames to either sniff traffic between unassociated STAs or to inject arbitrary frames into the network.

It should be noted however, that for a device to be able to receive Class 1 frames,

[10]With the exception of `Self-protected Action` frames (used in mesh networks) and some Management frames that are transmitted when the robust management frame service (802.11w) is enabled.

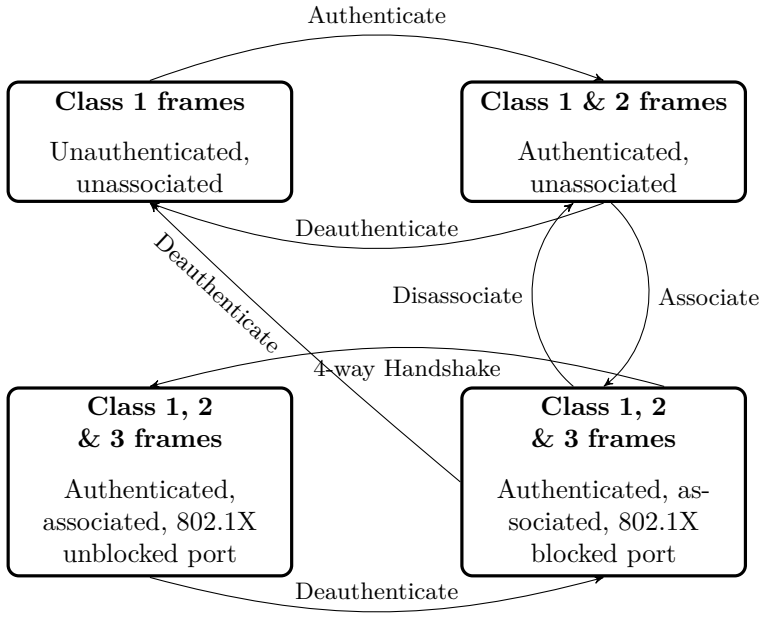


Figure 6.4.2: 802.11 non-mesh STA state transition diagram. Each state corresponds to one or more classes of frames that may be exchanged between two STAs [127, p. 1012].

it must be tuned to the same channel as the transmitter[11]. A station will listen on a channel for at least `MinChannelTime`. If the channel is idle, it will switch to the next channel. Otherwise it will wait until `MaxChannelTime` [127, p. 107] before switching to the next channel [60, 61, 87]. Thus, the MS could wait for the tracked station to switch to its own channel or alternatively, transmit the Class 1 frame on multiple channels at the same time. In the latter case, monitoring these channels is required as well in order to receive any responses to the stimulus frame, e.g. by using multiple interfaces in monitor mode.

6.4.2 Beacon frames

Among Class 1 Management Frames [127, p. 1013], a first type that we can utilize to increase the transmission probability of nearby devices is the `Beacon` frame. Many popular retail stores, hotels, bars, network operators, universities,

[11] Since Wi-Fi channels overlap, it is sufficient to be tuned to an overlapping channel.

etc. offer internet access to their visitors in the form of an (often open) wireless network. Such networks can be identified with for example the SSIDs "attwifi", "tmobile", and "eduroam". On the other hand, the popularity of SSIDs such as "linksys" and "dlink" can be attributed to the default vendor configuration of certain home APs.

In a tracking system, we can configure the MSs to spoof said SSIDs by transmitting crafted `Beacon` frames matching the SSID *and* Robust Secure Network (RSN) configuration, i.e. the security parameters, of the target network. Surrounding devices that automatically connect to known Wi-Fi networks and have one or more of these spoofed SSIDs in their PNL, will be more inclined to transmit frames because of automatic connection attempts [34, 62, 88, 176]. For example, `Probe Request` frames will be transmitted by the device before connecting in order to obtain the capabilities of the (fake) AP.

To increase the maximum number of SSIDs that can be spoofed and to conserve computing power on the MS, one can choose to only spoof `Beacon` frames and ignore `Association Requests` from surrounding devices. Further, the `Beacon Interval` field value can be increased so that `Beacons` do not need to be transmitted as frequently.

Similar to spoofing popular SSIDs, we can spoof Personally Identifying Wireless Networks (PIWNs) as well [34, 62]. Instead of composing a list of popular SSIDs, we can capture SSIDs from `Probe Requests` transmitted by a specific device and propagate this information to the other MSs. Hence, the goal here is not to trigger transmissions from as many surrounding devices as possible, but to instigate transmissions from one particular device that is associated with this rare SSID. An example of this approach is graphically shown in Figure 6.4.3.

Lastly, Vanhoef et al. have recently demonstrated that by including Hotspot 2.0 IEs in `Beacon` frames, Windows 10 and Linux clients will transmit Access Network Query Protocol (ANQP) Requests using their original MAC address to request more information about the AP [268]. We will discuss Hotspot 2.0 and ANQP in detail in Section 6.4.4.

6.4.3 Control frames

The Request To Send (RTS), Clear To Send (CTS), and Acknowledgement (ACK) Control Frames [127, p. 1012] form another type of Class 1 frames we can use for instigating transmissions. The first two of these frames, RTS and CTS, are

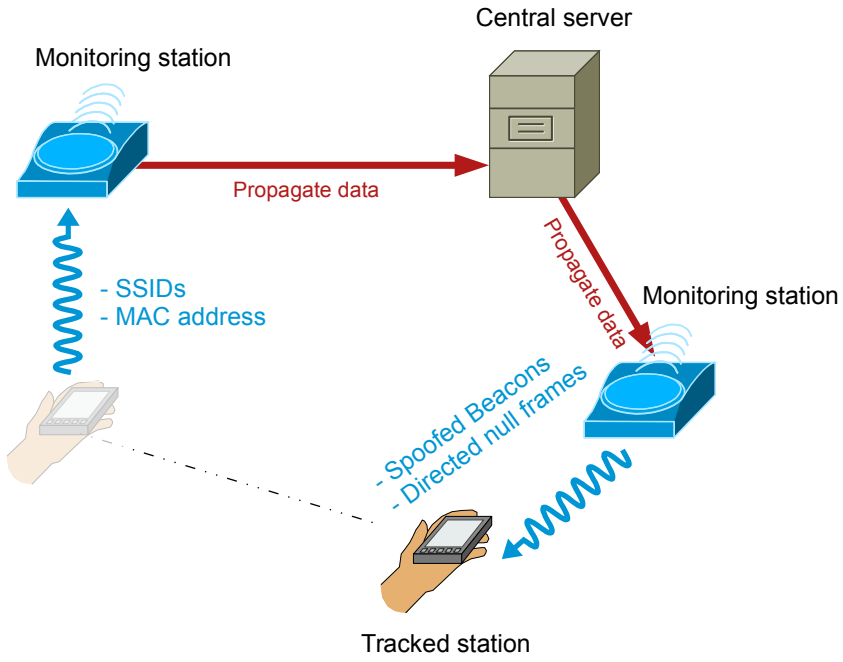


Figure 6.4.3: An example topology where data transmitted from the tracked station, such as its MAC address and SSIDs are captured and propagated to other MSs. This information can subsequently be used to spoof PIWNs and to transmit directed Null Data frames, instigating more transmissions from the tracked station.

exchanged prior to data frames in order to distribute medium reservation information. A STA receiving either one of these frames can extract the period of time that the medium is to be reserved from the `Duration` field, and consequently wait before transmitting in order to avoid collisions [127, p. 824]. As demonstrated by Musa et al. [176], this mechanism can be exploited by transmitting fake RTS frames containing *the receiver's* MAC address as the TA field of the RTS frame. The receiver will then respond with a CTS frame containing its own MAC address, revealing its presence[12].

ACK frames can be exploited in a comparable manner. According to the 802.11 standard, *each* data frame must be positively acknowledged[13] with an ACK

[12] After publication of this work, a variant of this attack was discussed by Martin et al. in [164]. They demonstrate that some devices will keep responding to RTS frames addressed to its non-random MAC address even if MAC address randomization is enabled.

[13] Although technically, exceptions such as `Block Ack` frames, which can acknowledge multiple

frame. Therefore, most Wi-Fi chipsets have implemented the ACK mechanism in hardware as to ensure a timely acknowledgement of received frames. Now, consider what happens if we transmit a data frame with both the RA and TA fields set to the tracked device's MAC address. In this case, the receiver's hardware will simply copy the TA field from the data frame (its own MAC) into the RA field of the ACK frame, again revealing its presence.

A disadvantage of using these Control Frames for instigating transmissions is that one is required to know the receiver's MAC address. This might be problematic if it is frequently randomized, since a STA should not respond to its original MAC address after randomization. On the other hand, if the MAC address is randomized infrequently or if the device still responds to its original MAC address, this approach can be used to reliably instigate a response for every transmission.

6.4.4 Action frames

Action frames are a type of frame intended for extended management functionalities [127, p. 449]. At the time we conducted this experiment, there were 20 non-reserved "categories" of Action frames, each offering a different service. For example, Public (Category 4) Action frames are Class 1 frames intended for inter-BSS, intra-BSS, AP to unassociated STA, and GAS communications [127, p. 743].

As we saw earlier, a STA is allowed to transmit and receive Class 1 frames even when it is not a member of any BSS, which makes these frames interesting candidates for location trackers. The 802.11 standard states that in this case, the "wildcard BSSID" should be used, which is equal to the broadcast MAC address "`ff:ff:ff:ff:ff:ff`". In the following sections, we will discuss Block Acks, GAS, Spectrum and Radio Measurement frames, Tunneled Direct-Link Setup (TDLS), and Wireless Network Management (WNM).

Block Acks

In Section 6.4.3, we have seen that every 802.11 frame must be positively acknowledged on receipt. However, STAs with QoS support can choose to acknowledge multiple frames at the same time via `Block Acks`, reducing overhead. When two communicating STAs both support the `Block Ack` mechanism, the originator

frames at the same time or a "No Ack" policy can be enforced by the transmitting STA.

STA can set up a `Block Ack` exchange by sending an `Add Block Ack (ADDBA) Request` frame. The recipient STA must then accept or reject the request by responding with an `ADDBA Response` frame. Then, the originator can transmit multiple data frames followed by a `Block Ack Request`, which is acknowledged according to the `Block Ack` policy [127, p. 904].

Unlike normal ACKs, `Block Acks` are not Control frames, but a category of Action frames that was introduced in the 802.11e amendment [126]. An important consequence is that both the transmitter and receiver MAC address fields are present in the frame.

Generic Advertisement Service

In 2011, the IEEE introduced a set of Class 1 Public Action frames under the 802.11u "Interworking with External Networks" amendment. The Interworking amendment, as the name suggests, primarily focuses on enabling information transfer between 802.11 devices and Subscription Service Provider Networks (SSPNs). These are subscription based networks offered by a certain provider, such as cellular networks. The amendment additionally defines a number of new procedures for network discovery and selection, interaction with emergency services, and a QoS mapping from the SSPN's QoS settings to the 802.11 QoS mechanism [127, p. 78]. In 802.11u, an interesting candidate for instigating transmissions is GAS.

GAS allows a STA to discover network properties or services provided by a SSPN, such as whether the network provides an internet uplink. Here, the SSPN is the entity that validates the user's credentials and offers its services to the user through the AP. GAS can also be used to discover the services offered by a peer STA, e.g. Wi-Fi Direct or P2P Groups [42].

A STA can discover available services by embedding a GAS `Initial Request` inside an 802.11 `Public Action` frame and transmitting it to the peer STA or AP. Since GAS queries may be transmitted *before* association, they allow the mobile device to select the most suitable P2P Group or AP before connecting. A unique byte value called the "Dialog Token" is used for matching requests with their corresponding responses in case several GAS requests are transmitted concurrently.

For querying information from the peer STA (e.g. in case of an AP: whether the network provides internet access), different advertisement protocols can be used. The default and mandatory supported advertisement protocol is ANQP. Upon

receiving a GAS `Initial Request`, the peer STA replies with a GAS `Initial Response`. If the response is too large to fit in one frame, the remainder of the response is queried and delivered with respectively GAS `Comeback Requests` and GAS `Comeback Responses`.

With the objective of instigating transmissions in mind, GAS has a few interesting properties. First, a device that has Interworking enabled must support GAS, and GAS queries can be performed peer-to-peer in the unassociated state. Second, ANQP must be supported as the default advertisement protocol per standard definition [127, p. 1145]. ANQP queries can however contain different elements, each with a different purpose. As an example, the "Capability List" element contains the capabilities supported by a STA.

With this knowledge, MSs can create their own ANQP requests, embed them in a GAS `Public Action` frame, and broadcast these frames in order to obtain a GAS `Response` for each device in the vicinity that supports GAS. Since the frame can be broadcast and transmitted in the unassociated state, this technique has the potential to instigate transmissions on demand if the targeted device supports GAS.

Spectrum and Radio Measurement

`Spectrum` and `Radio Measurement` frames, defined in the 802.11k amendment, can be exchanged between a pair of STAs to determine the channel load, the received signal power, noise histograms, etc. The type of measurement to perform is determined by the `Measurement Type` field of the `Measurement Request` frame.

According to the 802.11 standard, support for the `Basic Request` is mandatory and a STA in a BSS should only generate a `Basic Report` in response to a `Basic Request` if the request is received from the AP with which it is associated [127, p. 1048].

Tunneled Direct-Link Setup

A TDLS link can be set up between two peer STAs when they wish to use a feature that is not supported by the BSS itself, e.g. a certain high throughput data rate. Frames transmitted in this fashion are said to be transmitted over the TDLS direct link.

Support for the TDLS protocol can be determined by inquiring the STA with a `TDLS Discovery Request`. If the targeted peer STA supports TDLS and if the BSSID is correctly set, it must respond with a `TDLS Discovery Response`.

Wireless Network Management

Another service introduced in 802.11e is WNM. This service is used for assorted management tasks such as requesting diagnostics from a STA, announcing that a STA will enter sleep mode, requesting channel usage information, etc. Two interesting candidates for instigating transmissions are the `Event Request` and `Timing Measurement Request` frames.

The former type can be used to request another STA to report one or more events, such as the `Peer-to-Peer Link` event [127, p. 777]. The latter type is used to synchronize a local clock time between two STAs [127, p. 1131], which could potentially serve as a useful implicit identifier in the context of tracking.

Wi-Fi Direct

A specification for facilitating peer-to-peer communication between two non-AP STAs, named "Wi-Fi Direct", was released to the public by the Wi-Fi Alliance in 2010 [279]. Instead of connecting to a real AP, a STA can take on the role of AP and form a Peer-To-Peer (P2P) Group. Other STAs can discover these P2P Groups using passive (`Beacons`) or active (`Probe Requests`) scanning mechanisms [42]. Since the user of a device explicitly needs to cooperate in order to configure the device to scan for P2P groups[14], we will not consider Wi-Fi Direct further.

6.4.5 Stimulus frame candidates

Of all stimulus frame types discussed above, GAS Requests seem the most promising for instigating transmissions, since unassociated peer STA to peer STA communication is explicitly allowed by the standard for this frame type. This is not the case for all frames we discussed, though it would be interesting to see

[14] In Android 6.0.1 for example, the user must go to the "Wi-Fi Direct" menu under the "Advanced Wi-Fi" settings to scan for P2P Groups.

how a device or chipset reacts when it receives these frames from an unassociated STA such as a MS. For example, `Measurement` frames, TDLS frames, and WNM frames can normally only be transmitted peer-to-peer when *both* STAs are associated to the same BSS. In Section 6.5.3, we will determine whether these constraints are respected by the device vendor's implementation, and under which conditions the discussed frames are accepted by the receiving STA. Furthermore, we will compare these frames in terms of their ability to increase the transmission frequency of nearby devices.

6.5 Evaluation

In order to quantify the effectiveness of our IE based fingerprinting technique and of our techniques for instigating extra transmissions from nearby devices, we have performed several experiments. We shall henceforth refer to these experiments as respectively the *fingerprinting* and *transmission rate* experiments. The used code and anonymized data sets have been made available publicly at `https://github.com/rpp0/wifi-mac-tracking`, `CRAWDAD.org` [142], and `Wicability.net` [217].

6.5.1 Attacker model

We first define the following goals that an attacker attempts to accomplish. In the fingerprinting experiment, the attacker's goal is to uniquely identify as many devices as possible without relying on explicit identifiers. In the transmit rate experiment, the attacker attempts to instigate as many transmissions as possible from nearby devices. We make the following assumptions about the attacker and observed devices:

- At least one MS with an interface configured in monitor mode is used to track devices. To determine a trajectory, at least two MSs are required.
- Observed devices may or may not be associated to an AP. The attacker cannot determine whether this is the case.
- The attacker has a minimum amount of information at their disposal: we assume that in the best case (for the attacker), only a single `Probe Request` frame is observed by the MS per device. Additionally, explicit identifiers such as the MAC address may be randomized for every transmitted frame.

- Devices may appear or disappear from the tracking system's set of currently observed devices at arbitrary times. Consequently, techniques such as correlating the frame sequence numbers (Section 6.3.2) or scrambler seed (Section 6.3.1) cannot be used, since these values may have diverged or reset by the next time a device is observed.

Note that the above constraints are typical in non-cooperative tracking scenarios, such as tracking the visitors at an event or tracking (smartphones located in) vehicles on the road.

6.5.2 Fingerprinting experiments

For evaluating our *fingerprinting technique* (Section 6.3.3), we have set up a low-cost tracking system consisting of 8 commodity hardware MSs in a remote positioning topology (see Figure 6.2.1 on page 89). For the MS, we have used MikroTik 5RB912UAG-2HPnD devices equipped with an AR9342 chipset (see Figure 6.5.5 on page 130), though any device that supports monitor mode could be used for the same purpose. The stock firmware of the devices was replaced with OpenWRT Chaos Calmer[15]. Frames were captured using a custom application with `libpcap`, and one interface in monitor mode.

The system was deployed at the Glimps 2015 music festival in Ghent, Belgium, which took place from 10 to 12 December 2015. The MSs were placed at the locations shown in Figure 6.5.1 on page 127. Here, each MS was configured to forward one `Probe Request` frame per unique device to a central server over a secure link. Recall that we designed our fingerprinting technique with the constraint of having only one `Probe Request` at our disposal in mind. In total, 51,975 `Probe Requests` were analyzed. In compliance with ethical research guidelines, no data frames were captured at the festival in order to ensure the privacy of the visitors.

Fingerprint uniqueness

The effectiveness of our fingerprinting approach from Section 6.3.3 was evaluated by measuring the ratio of the number of unique fingerprints over the number of unique devices (MAC addresses). Ideally, there should be one fingerprint

[15] OpenWRT is a free Linux distribution for embedded devices, and can be downloaded at `https://openwrt.org/`.

per device. Each fingerprint is solely based on the bits from a single `Probe Request`. Only non-random MACs (28,048 out of 83,055 MACs) were considered, so that the MAC address can be used as a baseline to compare the performance of the fingerprinting algorithm. Furthermore, incorporating random MACs in the analysis would overestimate the observed number of unique devices.

Figure 6.5.2(a) on page 128 plots the overall fingerprint uniqueness for several test set sizes of non-random MACs for $\lambda = 0$, so that the variability is maximized and the stability is minimized. Figure 6.5.2(b) shows the same experiment, but with $\lambda = 1$ so that the variability is minimized and the stability is maximized. Recall from our discussion in Section 6.3.3 that the variability represents the uniqueness of the fingerprint, and that the stability gives an indication of how likely a fingerprint will remain identical for a given device. It should be mentioned that these results still underestimate the fingerprint uniqueness for two reasons. First, a non-standard compliant implementation may use MAC address randomization, but not set the locally administered bit [268]. As a result, the same device will incorrectly be interpreted as a set of different devices with the same fingerprint, hence underestimating the uniqueness of this fingerprint. Second, although unlikely in practice, `Probe Requests` could be spoofed by an adversary in order to disrupt a tracking system.

From the results shown in Figure 6.5.2, we conclude that for a MS that has observed a small dataset of 50 to 100 devices, the uniqueness of our fingerprint ranges from at least 80.0 to 67.6 percent. If the test set size is increased further, more devices with similar IEs will be encountered eventually, and the uniqueness of the fingerprint will therefore decrease. For large datasets of 1,000 to 10,000 devices, the uniqueness of the fingerprint drops between at least 33.0 to 15.1 percent. These results are encouraging, because a single MS will typically only observe a small set of devices, and the uniqueness for such small sets is high.

Deanonymization

Since the goal of the tracking system is to deanonymize devices, i.e. to link random MACs to a single fingerprint, having a large number of overlapping fingerprints is unfavorable.

To overcome this issue, note that we can utilize the fingerprinting algorithm in conjunction with temporal information. After all, the number of devices observed by a MS over a certain period of time is likely to be much smaller than the complete set of observed devices. Furthermore, when a device exposes its non-

random MAC address at some point in time, random MAC addresses with the same fingerprint near that time are more likely to be associated to that device. The deanonymization of random MAC addresses can thus be performed on a *subset* of devices instead of the complete set. As shown in Figure 6.5.2, decreasing the test set size (or time interval) indeed increases the relative uniqueness of fingerprints, since the probability of observing devices with similar IEs decreases.

Multiple approaches can be considered for determining the subset size. In a naive approach, one could bin devices according to a specified time interval, but then the question remains of how to choose the bin size and how to handle devices that are observed at the boundary of a bin. Alternatively, the fingerprint of the random MAC can be mapped to the MAC that was *closest* in time and has an identical fingerprint for the highest probability of a correct match. In Figure 6.5.3 (page 128) for example, the random MACs $r_1 - r_5$ with fingerprint f_1 are mapped to their corresponding non-random MACs $m_1 - m_3$. The algorithm pseudocode is shown in Figure 6.5.4.

When using the above algorithm to map a random MAC to its closest non-random MAC, we discovered that a matching fingerprint can be found with 99% probability. However, it should be noted that these mappings cannot be guaranteed to be correct as opposed to our experiments where only non-random MACs were considered. That is, the information required to validate the accuracy of the mapping (i.e. the true MAC address of the device) is not available in an uncontrolled environment, which is an inherent problem in *non-cooperative* deanonymization. One solution to overcome this problem could be to install an application on each mobile device partaking in the experiment. This application can provide the true MAC address to the fingerprinter for each random MAC, so that the accuracy of the deanonymization can be determined. A large dataset containing this information would be an interesting contribution for future work.

6.5.3 Transmission rate experiments

To correctly evaluate the *transmission rate increasing techniques* (Section 6.4.1), a number of complications had to be addressed. A first complication is that in order to measure the transmission frequency of a device, the device must remain in range of a MS an equal (preferably large) amount of time for each tested technique. This is rarely the case in a field setup (e.g. a music festival or shopping center), since here, devices are able to roam freely and are typically observed only a few times by a single MS. Secondly, the set of tested devices must

Table 6.5.1: Number of devices per vendor OUI as observed during our evaluation experiments. A combined total of 138 unique devices were observed during the two experiments.

Vendor	# of devices
Intel	40
Apple	21
Samsung	13
Lenovo / Motorola	11
OnePlus	11
LG	7
Hon Hai	6
Nokia / Microsoft	5
Murata	4
Compal	2
HTC	2
Cisco	2
TP-Link	2
Axis Communications	2
Other	10

be diverse and sufficiently large in order to be representative.

For these reasons, we have chosen to perform the evaluation of the transmission rate increasing techniques at our research lab. Here, devices are more likely to remain in range of the MS for extended durations compared to a field setup. At the same time, there is a healthy model and vendor diversity between devices owned by the researchers (see Table 6.5.1).

We have performed two experiments. The goal of the first experiment is to determine under which conditions a device will respond to a stimulus frame. Our second experiment shows how these methodologies compare against each other and against traditional approaches such as `Beacon` spoofing in terms of transmission frequency. Only non-random MACs were considered in order to prevent multiple observations of the same device.

Response conditions

For each of the frames discussed in Section 6.4.4, we have determined under which conditions and for which STAs they can be used to instigate transmissions. We define four test cases in decreasing order of knowledge required by the MS:

1. **Known BSSID:** The targeted STA responds to a broadcast stimulus frame only if the `BSSID` (`addr3`) and `Transmitter Address` (`addr2`) fields are correctly set to the associated AP. In other words, no frames from unassociated STAs are accepted. This requires the most knowledge by the MS, since the MS must encounter the target while it is associated to an AP, and this AP's BSSID must be spoofed.
2. **Unicast:** The targeted STA responds to the stimulus frame only if addressed directly. Here, the MS would only need to have knowledge of the target's current MAC address. The BSSID is set to "`ff:ff:ff:ff:ff:ff`" (the wildcard BSSID).
3. **Broadcast BSSID:** The targeted STA responds to the broadcast stimulus frame when the BSSID field is set to "`ff:ff:ff:ff:ff:ff`". This allows the MS to probe all devices in range.
4. **Zero BSSID:** The targeted STA responds to the broadcast stimulus frame when the BSSID field is set to "`00:00:00:00:00:00`". This non-standard behavior might indicate a misinterpretation of the standard or an implementation bug where the BSSID field is not properly validated.

This experiment was performed as follows: first, we used a TP-Link TL-WN722N dongle (Atheros AR9271 chipset) in monitor mode (see Figure 6.5.5) to scan for in-range BSSIDs and STAs. For this purpose we have created a Python script that uses the "Scapy" packet manipulation library. The code of this script is published on Github[16].

After the initial scan, the script continuously transmits each type of stimulus packet for 60 seconds for each of the four test cases. Between each test, the experiment was paused for 10 seconds to prevent slow processing of packets from influencing the next test. The experiment was run for a total period of 17,038 seconds, observing 136 unique STAs and 27 BSSIDs. The packet trace of this experiment containing each of the stimulus frames and their responses can be found at Online Resource 1 [214]. There were no other devices transmitting these

[16] `https://github.com/rpp0/wifi-mac-tracking`

Table 6.5.2: Overview of the number of unique STAs out of 136 that responded to a stimulus frame type (rows), for the 4 test cases that we defined in Section 6.5.3 (columns).

	Known BSSID	Unicast	Broadcast BSSID	Zero BSSID
ADDBA Request	27	7	4	3
GAS Request	4	6	7	6
Basic Measurement Request	9	1	0	0
CCA Request	11	1	0	0
Channel Load Request	7	2	2	2
STA Statistics Request	8	2	2	2
Frame Request	8	2	2	2
Link Measurement	8	2	2	2
WNM Event Request	2	2	2	2
WNM Timing Measurement Req.	2	2	2	2
TDLS Discovery	0	0	0	0
TDLS Setup	0	0	0	0

requests at the time of the tests. The number of devices that sent a response for the stimulus frame per experiment is shown in Table 6.5.2. These results will be discussed in the following sections.

ADDBA Request

For the `ADDBA Request` frame (Section 6.4.4) test, we have determined that 27 out of 136 devices responded during the known BSSID test, 7 devices replied to unicast frames, 4 devices responded to a broadcast BSSID `ADDBA Request`, and 3 devices replied in the zero BSSID test.

Among the devices that responded to broadcast and zero BSSID frames were 3 Intel chipsets returning an `ADDBA Response` with error code 37 (request declined), and 2 Axis Communications chipsets responding with a `Block Ack Error` frame

(category 131). These devices thus leak their current MAC address to the MS in response to a broadcast ADDBA frame.

GAS Request

In the `GAS Request` frame (Section 6.4.4) experiment we continuously transmitted ANQP queries containing the "Vendor Specific" element. We have chosen this element specifically because it is the only element supported in peer-to-peer mode by the most popular supplicant for Android and Linux devices, `wpa_supplicant` [14].

We have observed that 4 devices responded in the known BSSID case, 6 devices replied to unicast frames, 7 devices responded in the broadcast BSSID case and 6 devices in the zero BSSID case. Unlike what we have seen for ADDBA Requests, there is no significant difference in the number of observed devices between the known BSSID and broadcast BSSID tests. This means that if GAS is supported by a device, the device is likely to respond to broadcast peer-to-peer frames.

Support for GAS by a device could be determined in two ways. A first is to look at the Interworking IE transmitted by a device in `Probe Requests`. However, we observed that not all devices that transmit the Interworking IE respond to GAS Requests, and not all devices that respond to GAS Requests transmit the Interworking IE. Therefore, the presence of the Interworking IE in `Probe Requests` is not a useful metric to determine how many devices support peer-to-peer GAS Requests.

A better approach is to look at "Passpoint" certification instead. Devices that support 802.11u are often marketed using the terms "Passpoint" and "Hotspot 2.0". Here, Passpoint is the label that a device obtains when it passes the certification program by the Wi-Fi Alliance, and Hotspot 2.0 is the name of the technical specification that was developed by the Wi-Fi Alliance based on the 802.11u amendment [275].

A full list of 938 Passpoint certified devices can be found via the Wi-Fi Alliance Product Finder tool [276]. It should be noted that this list is merely a lower bound for vulnerable devices. For example, among the devices that responded, we observed two Axis Communications devices that are not Passpoint certified.

Spectrum Management Request

In context of Spectrum Management (Section 6.4.4), we have tested `Basic Requests` and `Clear Channel Assessment (CCA) Requests`.

For `Basic Requests`, no devices responded during the broadcast BSSID and zero BSSID test cases. During the known BSSID testcase however, a response was received for 9 devices from various vendors. Only a single Intel chipset responded with a `Spectrum Management Error` frame during the unicast test. The results for `CCA Requests` were identical, except 11 devices were observed during the known BSSID test.

Given these results, we conclude that although Measurements Reports can be instigated if supported by the device, the BSSID must be known to the MS, and the STA must be associated.

Radio Measurement Request

For Radio Measurement (Section 6.4.4), we have tested `Channel Load`, `STA Statistics`, `Frame` and `Link Measurement Requests`. For each of these frames, two Axis Communications chipsets responded with error frames during the broadcast BSSID, zero BSSID, and unicast test cases. During the BSSID test cases, we observed responses from 7 devices for the `Channel Load` experiment, and 8 devices for the other experiments. Interestingly, the responses all originated from Motorola and OnePlus smartphones, which suggests that Radio Measurement is only supported by these devices.

WNM and TDLS

For WNM related frames, we were only able to instigate error responses from the two Axis Communications chipsets. Other devices did not respond in any of the test cases. No devices responded to TDLS frames. A possible explanation is that these protocols are rarely supported[17]. We decided to mention these results nevertheless, since they could be of value to future work.

[17] In Wireshark, the de facto standard tool for packet inspection, WNM `Event Request` IE parsing is not fully supported, as can be observed from the provided traces.

Table 6.5.3: An overview of the advantages of each technique discussed in Section 6.4.

	Beacon spoofing	RTS / Null Data	GAS Request	ADDBA Request
Response contains current MAC	●	●	●	●
Response contains implicit identifiers	●			
Immediate reply from device	◗*	●	●	●
Can be broadcast	●		●	◗†
Requires no knowledge about device			●	●
Commonly supported	●	●		●

* With an implementation dependent delay (see [88]) and only if at least one SSID in the device's PNL is guessed.
† Implementation dependent.

Discussion and comparison

Based on our experimental results, we hypothesize that `GAS Request` frames will be the most effective to instigate transmissions in practice. These frames can be broadcast per standard definition and will trigger responses regardless of whether the targeted STAs are associated to an AP. However, the GAS protocol must be supported by the receiving device. A second interesting type are the ADDBA frames, since some Intel chipsets similarly respond to broadcast frames. This behavior can additionally be used as an implicit identifier. In Table 6.5.3, we compare the techniques using these frames to previous approaches.

For the comparison experiment, our setup consists of a single MS that forwards all captured frames to a central server. The MS hardware is identical to the hardware used in Section 6.5.2, i.e. a MikroTik RB912UAG-2HPnD equipped with an AR9342 chipset (see Figure 6.5.5 on page 130).

To compare the effectiveness of the techniques from Section 6.4 in a realistic tracking scenario, we have measured the number of Class 1 frames that the MS received over a total period of 8 hours. Only devices with non-random MACs

that sent more than 100 frames (32 devices) were considered. The tests for each technique were interleaved in order to mitigate the effects of changing channel conditions. As such, the tests were performed intermittently for a period of 5 minutes each.

Besides novel techniques, we have tested the techniques used in previous works, i.e. `Beacon` spoofing (Section 6.4.2) and directed `Null Data` frames (Section 6.4.3). Recall that we assume the MS has no knowledge about nearby devices, analogous to the "broadcast BSSID" test case from Section 6.5.3. For the `Null Data` technique however, we had to relax this assumption and allow unicast, since responses cannot be observed otherwise. For the `Beacon` technique, our MS spoofed several local and popular networks, such as "linksys", "TELENETHOMESPOT", and "Proximus_FON".

The effectiveness of each technique was determined by comparing the number of received Class 1 frames from the device against the control test, where no special techniques were used. Table 6.5.4 shows the average number of frames received by the MS during the control test and technique tests. For each test, the standard deviation is high because the STA transmit behavior highly differs between implementations. We can also see that on average, only the Beacon spoofing technique performs worse than the control test. This can be attributed to a number of reasons: the tracked device must have one of the spoofed SSIDs in its PNL[18], the resulting `Probe Request`'s transmission is not immediate, the `Beacons` themselves cause more channel contention, and some devices stop probing after association [88].

Figure 6.5.6 graphically shows for each technique, the percentage of the 32 observed devices in function of the percentage of improvement compared to the control test. A log scale was chosen since some devices responded to *each* stimulus frame, vastly increasing their transmission frequency. In general, GAS frames appear to be the most favorable: unlike `Null Data` frames, GAS frames can be broadcast, appear to be supported by the most devices and also instigated the most transmissions. ADDBA frames were slightly less effective, since not all implementations respond to this type of frame. Null Data frames performed similarly, but require knowledge of the tracked device's MAC address. Finally, `Beacons` are the least favorable technique, which can be attributed to the fact that only 3 devices in our lab responded to the spoofed SSIDs. However, a `Probe Request` contains more implicitly identifying information (e.g. IEs) than other responses.

[18]Therefore, the choice of which SSIDs to spoof in a tracking system impacts performance

Table 6.5.4: Comparison between the average number of frames received by the MS during the control test and the technique tests for 32 unique devices. The average number of frames received highly varies between devices, but is greater than the control test for each technique except the Beacon technique.

	Average	Standard deviation
Control test	**155.8**	**224.7**
Common Beacons	148.2	219.4
Null Data	956.3	4529.8
ADDBA	168.9	233.5
GAS	13219.2	48693.7

6.5.4 Practical location tracking

Now that we have detailed and evaluated several techniques for fingerprinting a device and instigating transmissions from it, the question remains of how to apply these techniques in practice. The fingerprinting approach from Section 6.3.3 can be used to create a unique identifier for devices observed by the MSs, based on a single `Probe Request`. Temporal information can then be used to link random MACs with their corresponding non-random MAC (see Section 6.5.2).

If the transmission frequency of tracked devices is low or if the tracked devices are moving fast[19], the techniques that we discussed in Section 6.4.1 can be used. ADDBA frames and GAS Requests are the most effective for instigating transmissions, but their responses do not contain as much uniquely identifying information as `Probe Requests`. Therefore, these techniques will work best for non-random MACs. On the other hand, spoofed Beacon frames can instigate `Probe Requests` and are therefore useful for both non-random and random MACs, but this approach does not increase the transmission frequency significantly.

significantly.

[19]Note that when a device is moving fast, we can only use broadcast (or unicast if the target MAC is non-random and known), since roaming devices are rarely associated to one particular APs for an extended duration. Id est, we cannot exploit the "Known BSSID" assumption in this case.

6.6 Countermeasures

To prevent MAC-layer based tracking systems from tracking a user's device without their knowledge through the techniques discussed in this work, several countermeasures can be put in place by vendors of mobile devices. We will now briefly discuss these countermeasures. Ideally, they should be combined instead of considered separately.

- **Enable MAC address randomization:** Since the MAC address is a globally unique identifier, it must always be completely randomized for every transmitted frame when actively scanning for APs. In order to not break roaming functionality, the real MAC address can still be used when associating to an AP, on the condition that the device's PNL does not contain SSIDs that can be guessed by an attacker (see Section 6.4.2). Devices that use `wpa_supplicant` can enable this countermeasure through the options `mac_addr`, `rand_addr_lifetime`, and `preassoc_mac_addr`.

- **Reduce `Probe Request` frequency:** Since `Probe Request` timing information can be used as an implicit identifier [63], these frames should be transmitted infrequently and at random intervals.

- **Avoid directed `Probe Requests`:** `Probe Request` frames ideally must only contain an `SSID Parameter Set` IE with the broadcast or empty (null) SSID in order to prevent leakage of the device's PNL or connection history. Alternatively, the device can choose to only scan for networks passively [34, 63, 87].

- **Defer transmission of IEs:** Instead of transmitting all IEs for every `Probe Request`, the device should only share this information with an AP in the association stage, since `Probe Requests` contain identifying information [268]. This limits the tracker's opportunities to instances where the user manually connects or where an SSID from the PNL is known.

- **Ignore broadcast Class 1 frames:** Peer-to-peer Class 1 frames transmitted to `ff:ff:ff:ff:ff:ff` should be ignored. A receiver should only respond to such a request if the frame is directly addressed to its *current* MAC address.

- **Randomize `Sequence Numbers`:** The `Sequence Number` field in the MAC header should be randomized while the STA is unassociated in order to prevent deanonymization [112, 268].

- **Validate state machines:** A STA should only interpret a frame when it is received in the correct state. For example, an `ADDBA Request` frame should not be accepted from unassociated STAs, as this would imply a transfer of data frames.

- **Trusted locations:** The user should be allowed to select trusted geographical locations where the Wi-Fi chip is exclusively enabled, in order minimize the risk of being tracked [34].

6.7 Related work

In previous work, various non-cooperative 802.11 MAC layer based tracking algorithms and topologies have been proposed. Abott et al. have implemented a non-cooperative tracking system to estimate travel durations for vehicles on the road by solely monitoring MAC addresses [5]. Bonné et al. have created a similar tracking system for tracking movement patterns of event visitors [33]. Musa et al. have implemented a tracking system for vehicles using a probabilistic approach [176].

For *fingerprinting* devices on the MAC layer, an approach where a combination of transmitted network data, SSIDs, specific MAC header fields and transmission rates is used was discussed by Pang et al. However, in this work MAC header fields on their own were not yet considered practical to distinguish users uniquely [191]. Neumann et al. have used several parameters from the Radiotap header to learn a fingerprint for a device [178]. Cunche et al. utilized the rarity and frequency of SSIDs to fingerprint and link devices [63]. The works by Bonné and Chernyshev et al. [34, 52] use SSIDs to map observed devices to a set of visited geolocations using databases such as WiGLE.net [280]. Vanhoef et. al apply clustering techniques to a selection of IEs and to sequence numbers in order to identify a device. They also study the variability and stability of IEs using information entropy, but only consider the entropy of IE fields in their entirety and do not identify precisely which bits are responsible for leaking information as opposed to this study. Furthermore, in their work it is assumed that the adversary can acquire multiple transmissions from tracked devices in order to identify them [268]. Other works made use of timing differences in `Probe Requests` [62, 76, 87, 168].

One of the first works that aims to improve MAC layer fingerprinting by *eliciting more transmissions* from devices was published by Bratus et al. [36]. Here, the reaction of a STA to stimulus `Deauthentication` frames, `Beacons`, `Probe`

`Responses`, and failed authentications was observed and used to fingerprint the device. However, it is assumed that the STA is associated to an AP or joining a BSS. The work presented by Musa et al. [176] deserves a special mention because it demonstrates a number of preliminary techniques to instigate transmissions in context of non-cooperative tracking on the MAC layer. They have spoofed `Beacon` frame SSIDs to increase the frequency of `Probe Requests` from unassociated devices, and additionally used injected RTS frames for eliciting CTS frame responses. Similarly, introducing a known or common SSID to increase the transmission frequency of nearby unassociated devices is mentioned in the studies performed by Cunche et al. [62] and Bonné et al. [34]. An overview of all relevant works and their features is given in Table 6.7.1.

6.8 Chapter conclusions

Despite efforts by vendors for implementing MAC address randomization, we have shown how a device can nevertheless be fingerprinted and deanonymized, even if the device is not cooperating or not associated to an AP. In our approach we have discussed how a tracking system can combine implicitly identifying IE bits from a single `Probe Request` frame to form a fingerprint that is at least 80.0 to 67.6 percent unique for small sets of 50 to 100 devices, and at least 33.0 to 15.1 percent unique for large sets of 1,000 to 10,000 devices. We have evaluated these results using two datasets. The first dataset was recorded at Glimps 2015 and the second at our research lab, containing respectively 28,048 and 138 unique devices. Additionally, we have discussed and compared our work against previous works that aim to achieve similar goals. An overview of these works was given in Table 6.7.1.

Further, we have shown how these fingerprints can be combined with temporal information and how extra frames can be instigated by a tracking system when a device sends frames infrequently or not at all. We have demonstrated how a MS can exploit protocol design flaws and implementation vulnerabilities to achieve this goal, given that nearby devices support the respective protocols. More specifically, we have studied a wide array of Class 1 frames, such as Beacon, RTS/CTS, Null Data, GAS Request, ADDBA Request and other Action frames. Such frames can be actively injected by a MS to expose the presence of nearby devices more frequently and reveal more implicitly identifying information on both the PHY and MAC layers to the tracking system. We have experimentally determined that GAS frames are particularly interesting in this regard, as these frames can be broadcast and used to instigate transmissions on demand from Hotspot 2.0

and Passpoint compatible devices while unassociated. Compared to the control test, we measured a transmission rate improvement of 50 to 10,000 percent for 3 to 9 out of 32 devices (9.37 to 28.1 percent). ADDBA frames can be exploited in a similar fashion in some implementations.

As the diversity between devices increases in terms of capabilities and supported protocols, measures must be taken by vendors in order to prevent this kind of unsolicited location tracking by third parties. Fortunately, some of the defences suggested in this work and in related works have now indeed been incorporated into the Wi-Fi standard: in 2018, the IEEE released the 802.11aq amendment, which mentions randomization of the MAC address during scanning, resetting the sequence number and reseeding the OFDM scrambler seed after a MAC change, and refraining from sending `Probe Requests` that contain the SSID [128, p. 57–58]. Vendors of mobile devices have also started implementing these privacy-focused features in practice. For example, since Android 10, MAC randomization is enabled by default, even when connecting to an AP [107].

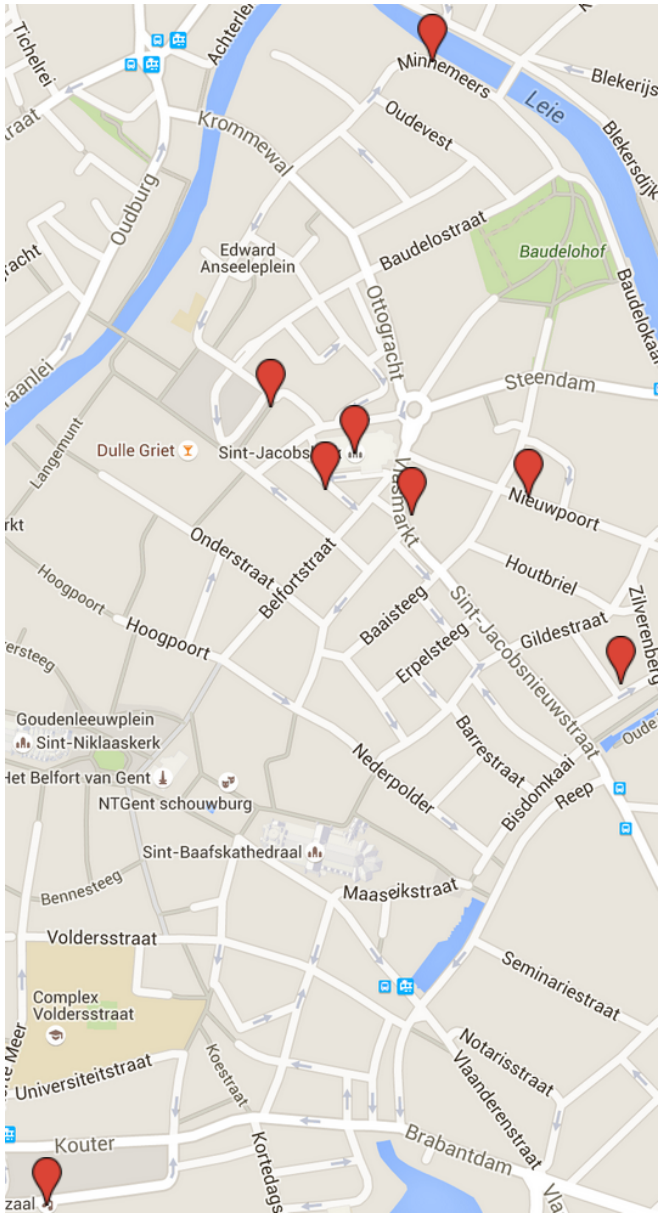
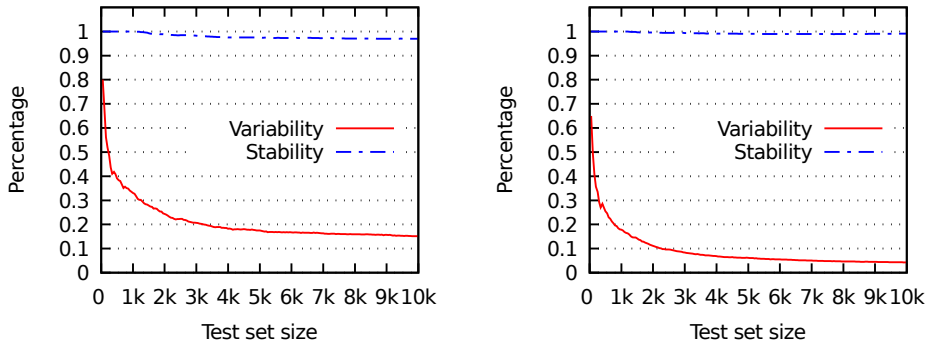


Figure 6.5.1: Distribution of MSs for the performed fingerprinting experiments.



(a) $\lambda = 0$ (b) $\lambda = 1$

Figure 6.5.2: The variability (solid line) of fingerprints for non-random MACs decreases due to collisions as the test set size increases. The fingerprint stability (dashed line) also slightly decreases due to some bits not being considered as unstable during training (e.g. when the training set is small). Results shown for a training set of 1,000 MACs.

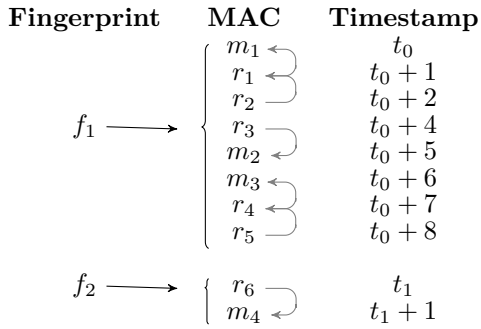


Figure 6.5.3: An example that shows how the fingerprints f_1 and f_2 can be used in conjunction with temporal information in order to link the random MAC addresses $r_1 - r_6$ to the non-random MAC addresses $m_1 - m_3$.

```
1: procedure DEANONYMIZE(frame)
2:     t ← frame.timestamp
3:     m ← frame.addr2                                  ▷ Transmitter MAC
4:     f ← fp(frame)                                    ▷ Get fingerprint bitstream(s)
5:     if is_random_mac(m) then
6:         for i in range (0, len(all_frames)) do
7:             t_i ← frame_i.timestamp
8:             m_i ← frame_i.addr2
9:             f_i ← fp(frame_i)
10:            if f_i = f then
11:                d_i ← abs(t − t_i)
12:            end if
13:        end for
14:        return m_i where d_i is minimal
15:    end if
16:    return m
17: end procedure
```

Figure 6.5.4: Procedure for deanonymizing random MAC addresses

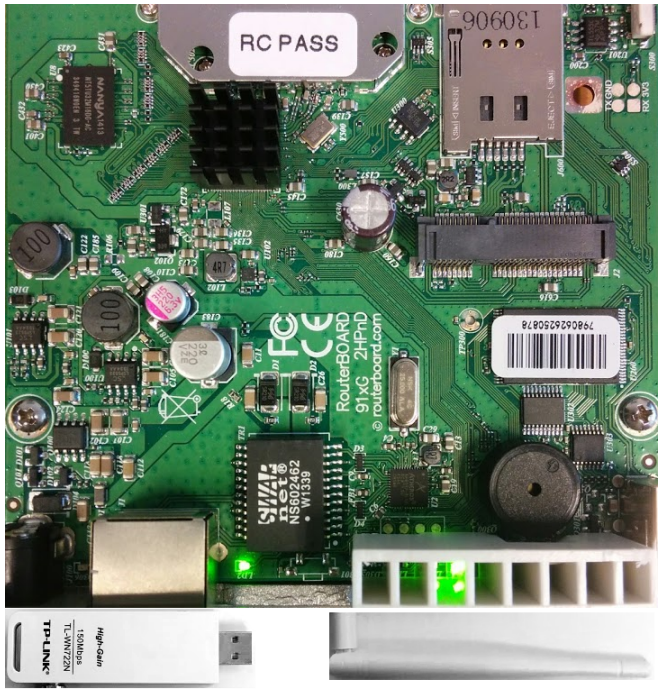


Figure 6.5.5: Top view of the hardware that was used for the MSs in our experiments.

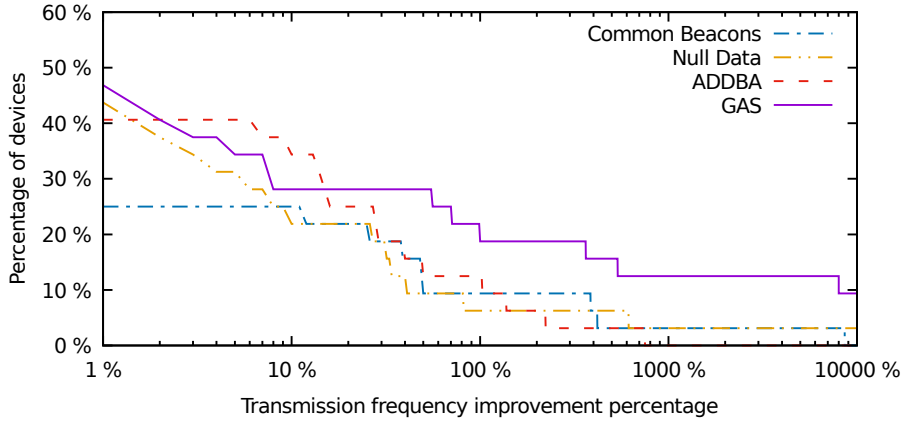


Figure 6.5.6: The percentage of observed devices in function of the percentage of improvement compared to the control test in log scale. The GAS technique is the most likely to trigger a response, and instigates the most responses compared to the control test: for 3 to 9 out of 32 devices (9.37 to 28.1 percent) there is an improvement of 50 to 10,000 percent.

Table 6.7.1: An overview of 802.11 PHY and MAC layer non-cooperative fingerprinting techniques.

	PHY				MAC									
	Clock skew	Scrambler seed	Time domain	Frequency domain	MAC address	Sequence number	Duration field	Timing	MAC header bits	SSIDs	Information Elements	Payload data	Stimulus frames
Kohno et al. [139]	●										●		
Arackaparambil, Jana et al. [10, 130]	●												
Hall, Ureten et al. [115, 263]			●										
Brik, Corbett et al. [38, 61]				●									
Bloessl et al. [29]		●											
Desmond, Franklin et al. [76, 87]							●						
Neumann et al. [178]							●		●				
Chernyshev, Cunche et al. [52, 62, 63]					●					●			
Pang et al. [191]					●				●	●	◗	●	
Guo et al. [112]						●							
Cache et al. [40]							●						
Abott, Bonné et al. [5, 33]					●								
Bonné, Musa et al. [34, 176]					●								●
Bratus et al. [36]									●				●
Vanhoef et al. [268]		●				●					●		
This work					●			●			●		●

Part III

Implicit Information Leakage in Wireless Communication

Why do you have to translate and decode things? Just let the image be. It will have a special kind of reality that it won't once it's decoded.

Laurie Anderson

7

A Multi-Channel Software Decoder for the LoRa Modulation Scheme

7.1 Introduction

In the introduction of this thesis, we briefly mentioned the increased interest of the industry in the automation or optimization of business processes through the use of IoT devices in an LPWAN. Such devices can for instance be used in context of smart metering, location tracking, WSNs, smart transportation systems and health monitoring [124]. In LPWAN networks, several low-power embedded devices are typically deployed in areas of interest, and perform M2M communication or interact with services on the internet in order to complete a certain computing task. For example, an internet-connected embedded device may be distributed to individuals suffering from a cardiovascular disease, so that their condition can be monitored in real time by doctors.

The heightened attention for these use cases sparked the creation of several PHY-layer modulation protocols that are optimized for LPWANs, i.e. for low power consumption and long range communications. Examples of these protocols are LoRa [156], Sigfox [241], Wi-Fi HaLow [278], LTE-M [2], and Weightless [274]. Out of those, LoRa is a proprietary, low-power, and long-range modulation scheme developed by Cycleo and acquired by Semtech in 2012 [234]. It is currently among the more popular protocols, with numerous gateways already deployed on a global scale [259].

Due to its proprietary nature, specialized hardware is required in order to transmit or receive LoRa messages. Examples of such hardware are the SX1272 transceiver developed by Semtech [235] and the RN2483 transceiver developed by Microchip [172]. Both transceivers expose a serial interface to the user. The serial interface can only be used to make high level configuration changes to the LoRa modem, and to transmit or receive payloads using LoRa modulation. Hence, the entire PHY layer of these transceivers is abstracted in hardware, and therefore cannot be accessed or modified.

Having access to the PHY layer of a wireless protocol is a desirable feature with many interesting use cases for research and development. For example, recent works have demonstrated the possibility to fingerprint individual transceivers using only PHY-layer properties of the signal [71, 219, 271]. This could be useful for tracking devices or intrusion detection. Another use case could be to perform software simulations of the PHY layer, e.g. to determine the effect of various channel conditions without requiring multiple physical deployments of hardware transceivers [24, 171]. As a final example, enabling modifications to the PHY layer allows for rapid prototyping of improvements in terms of security, performance

or reliability [27, 247].

In this chapter, we provide several contributions that aim to bring the advantages of PHY-layer access to the LoRa modulation scheme. First, we provide a detailed description of the LoRa PHY layer. This description includes newly added and undocumented features of the LoRa specification that were reverse engineered from hardware LoRa transceivers. To the best of our knowledge, we are the first to provide a complete and validated overview (**C6**). Second, we present our algorithms for the detection, synchronization, and decoding of raw PHY-layer LoRa frames using SDRs. These algorithms include a novel decoding approach and a novel clock drift correction approach for LoRa, both implemented in a complete and open-source software LoRa decoder using the GNU Radio framework (**C7**). Our decoder is capable of decoding multiple channels simultaneously in real time regardless of the frame's network identifier, similar to the capabilities of "monitoring mode" devices in context of 802.11 (Wi-Fi). Finally, we evaluate our decoder in a lab setup, and show that it can interoperate with hardware LoRa transceivers, using only inexpensive Commercial Off-The-Shelf (COTS) SDRs such as the RTL-SDR. In Chapter 8, we will subsequently use the decoder presented in this chapter to build a Proof of Concept (PoC) PHY-layer fingerprinting system for LoRa devices.

The structure of this chapter is as follows. In Section 7.2, we will present an overview of the LoRa PHY layer in consideration of our first contribution. Section 7.3 then shows how this knowledge can be applied to build a complete software LoRa decoder. In addition, we detail our novel demodulation and clock drift correction approaches. Our decoder will be compared in terms of compatibility with existing LoRa hardware and accuracy in Section 7.4, followed by a discussion of these results. Works related to this research will be discussed in Section 7.5. Finally, in Section 7.6, we will state the conclusions of this research.

7.2 LoRa PHY layer

In order to correctly decode LoRa-modulated data, a receiver must sequentially perform seven operations on the PHY layer, namely detection, synchronization, demodulation, deinterleaving, dewhitening, decoding, and packet construction. A partial description of these operations can be found in several technical reports released by Semtech [233, 235, 236, 251] and in the paper presented by Goursaud et al. [108]. However, the information contained within these works is insufficient to build a decoder that can interoperate with real hardware LoRa transceivers. To this end, we have reverse engineered a RN2483 LoRa transceiver, and provide

the first complete overview of the LoRa PHY layer in this section.

7.2.1 Modulation

The LoRa modulation scheme is based on Chirp Spread Spectrum (CSS) modulation [108], and defines a "chirp" as a single symbol [236]. A standard, unmodulated linear chirp is called a "base chirp", and can be mathematically described in function of the time t as follows [161]:

$$x(t) = e^{i(\varphi_0 + 2\pi(\frac{k}{2}t^2 + f_0 t))}, t \in [0, T] \tag{7.1}$$

where φ_0 is the initial phase, k is the rate of frequency change, and f_0 is the initial frequency. Given the channel bandwidth BW, the parameters f_0 and k are set so that the frequency increases from $f_0 - \frac{\mathrm{BW}}{2}$ to $f_0 + \frac{\mathrm{BW}}{2}$ over the duration T of the chirp. Hence, $f_0 = -\frac{\mathrm{BW}}{2}$ and $k = \frac{\mathrm{BW}}{T}$. Here, the chirp duration T depends on the bandwidth of the signal and on a parameter called the Spreading Factor (SF) according to the relation $T = \frac{2^{\mathrm{SF}}}{\mathrm{BW}}$ [233].

Given that $x(t + nT) = x(t)$ with $n \in \mathbb{N}$, an integer value $i \in \{0, 1\}^{\mathrm{SF}}$ can be modulated onto the base chirp by introducing a time shift of $\hat{t} = \mathrm{Gray}^{-1}(i)\frac{T}{2^{\mathrm{SF}}}$ to the signal in Equation 7.1, where Gray^{-1} stands for a Gray decoding operation [109]. This way, a symbol is essentially quantized into 2^{SF} time shift bins divided over the bandwidth, called "chips", that determine i. Upon reception of a modulated chirp with an unknown time shift $x(t + \hat{t})$, the chip value i can be recovered by sampling the signal at the chip rate and calculating:

$$i = \mathrm{Gray}(\arg\max(\ |\mathrm{FFT}(\ x(t + \hat{t}) \odot \overline{x(t)}\ )|\ )\ ) \tag{7.2}$$

where $\overline{x(t)}$ denotes the conjugate of a base chirp, the $\odot$ operator indicates element-wise multiplication, $|\mathrm{FFT}(x)|$ signifies the magnitude of the Fast Fourier Transform (FFT) of x, and Gray stands for Gray encoding. Figure 7.2.1 shows an example where a value of $i = 20$ is modulated onto the base chirp, shifting it by 192 samples.

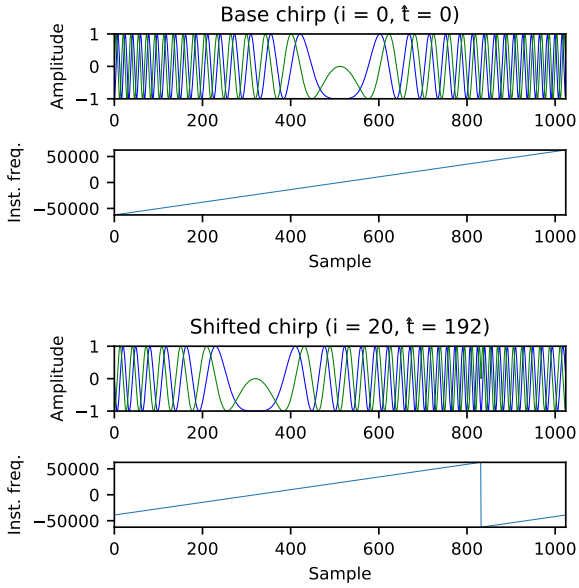


Figure 7.2.1: Example of an unmodulated LoRa base chirp and a chirp modulated with $i = 20$. The top rows of the figures show the time domain complex signal, whereas the bottom rows of the figures show the instantaneous frequencies of the signals in Hz.

7.2.2 Interleaving

When using the modulation approach described above, errors can be introduced due to noise, interference, and time or frequency offsets. These errors cause the receiver to derive an incorrect chip value from the modulated symbol. For example, a burst of noise could cause the peak of the FFT spectrum to appear at a different chip, therefore corrupting the entire chip value.

In order to limit the impact of bursty noise to a single bit error per symbol, multiple chip values are stacked together such that a bit matrix $\{0,1\}^{\mathrm{SF}\times(4+\mathrm{CR})}$ is obtained. Here, the Coding Rate (CR) or equivalently, the number of parity bits, can range from 1 to 4. For example, when using $\mathrm{SF} = 7$ and $\mathrm{CR} = 4$, we obtain a matrix $\{0,1\}^{7\times 8}$ as shown in Figure 7.2.2. A codeword of $4 + \mathrm{CR}$ bits is then obtained by diagonally deinterleaving the matrix. As such, the first chip value provides all first Least Significant Bits (LSBs) of the codewords, the second chip value provides all second bits of the codewords, etc. The direction of the

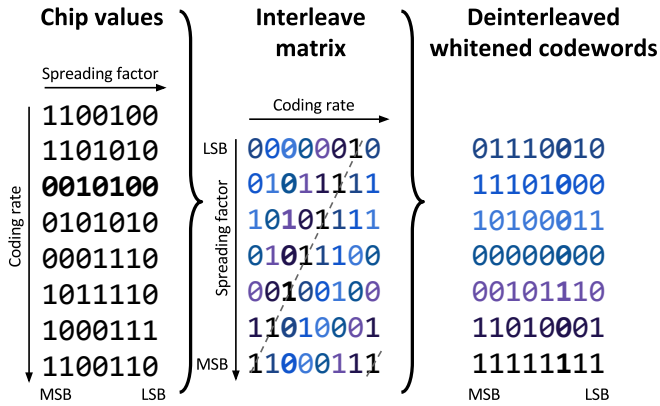


Figure 7.2.2: Diagonal deinterleaving with SF $= 7$ and CR $= 4$. The bits of the third chip value are shown in bold.

interleaving diagonal appears to be upwards in practice, in contrast to the LoRa patent where the interleaving diagonal direction is downwards[1] [233]. Observe that as a result of the interleaving operation, an entirely corrupted chip value now only affects one bit per codeword.

The LoRa specification also defines a "reduced rate" mode, in which the top two rows of the interleaving matrix are discarded. Consequently, the dimensions of the matrix become $\{0,1\}^{(\mathrm{SF}-2)\times(4+\mathrm{CR})}$, yielding two codewords less after deinterleaving. The discarded rows correspond to the LSBs of the chip values, which are more prone to errors because they correspond to narrower frequency bins in the FFT spectrum. Therefore, in reduced rate mode, a decreased data rate is traded for an increased robustness to noise. The PHY layer header of LoRa frames is always transmitted in reduced rate mode, whereas the payload bytes are only transmitted in reduced rate mode when SF 11 or SF 12 is used [235, p. 28, 112].

7.2.3 Coding

After deinterleaving, a number of codewords of size $4 + \mathrm{CR}$ are obtained by the receiver. The codewords of the frame payload are "whitened" in order to keep the data Direct Current (DC)-free [235, p. 75]. Here, whitening is defined as an

[1] Note that this has no impact on the decoding performance.

operation where the data is XOR-ed with a 9-bit LFSR after synchronization [235, p. 72]. The used coding algorithm is not explicitly mentioned in the patent or chipset datasheet [233, 235], leaving this an open choice to the vendor.

For the transceivers we considered during our tests (see Section 7.3), we have reverse engineered the coding scheme and discovered that a modified version of $4/(4 + \text{CR})$ Hamming coding is utilized in practice. Hence, each codeword results in 4 data bits when decoded, which are subsequently parsed according to the LoRa frame structure. In Section 7.3.3, we will discuss our approach for decoding the data further.

7.2.4 Frame structure

On the PHY layer, LoRa defines a frame structure with the following sequentially transmitted fields [235, p. 27–29]:

- **Preamble**: A variable-sized sequence of base chirps that is used for time and frequency synchronization.
- **Frame synchronization symbols**: Two modulated chirps whose value can be used as a network identifier. A hardware LoRa transceiver will drop frames containing synchronization symbols that do not match a preconfigured value.
- **Frequency synchronization symbols**: Two conjugate base chirps followed by a conjugate chirp with duration $\frac{T}{4}$, which can both be used for fine frequency synchronization.
- **Header (optional)**: Field containing the payload length, used data rate, a bit indicating the presence of a payload CRC, and 1-byte header checksum. A CR of 4 is always used in combination with reduced rate mode for the header[2] [233]. The header can be explicitly transmitted (*explicit* mode) or left out of the frame (*implicit* mode). In the latter case, the transmitter and receiver must configure the coding rate and CRC presence bit beforehand.
- **Payload**: Variable-length field containing the transmitted MAC layer data and a 2-byte CRC of this data.

Figure 7.2.3 shows an example LoRa signal and its frame structure.

[2]Since the header contains the payload length and coding rate, it is essential that these fields

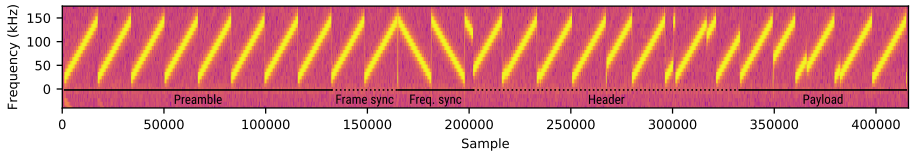


Figure 7.2.3: Annotated spectogram of an example LoRa signal transmitted with the RN2483 LoRa transceiver using SF 11 and CR 5, and received with a USRP B210 SDR.

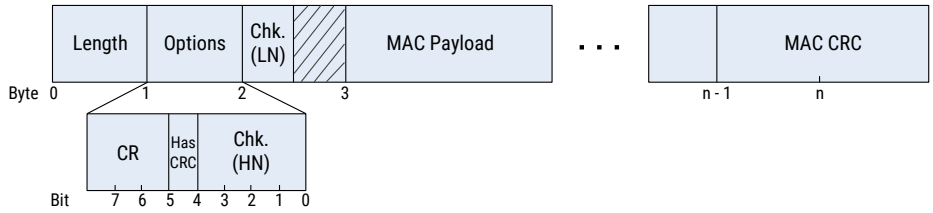


Figure 7.2.4: The fields of the LoRa PHY layer.

Header structure

The exact length and bit order of the fields in the PHY layer header are not explicitly stated in the specifications. However, since the transmission of an explicit header requires a SF of at least 7, and the header is always transmitted with CR = 4 at a reduced rate [233], the header must fit in an interleaving matrix of $\{0,1\}^{(7-2)\times 8}$. Therefore, the header length must be equal to 5 codewords of 8 bits, i.e. 40 bits in total. Any remaining bits in the interleaving matrix are used for the payload.

After decoding, the header data thus has a length of $40 \cdot \frac{4}{8} = 20$ bits or 2.5 bytes. We have experimentally determined that when transmitting data using a Microchip RN2483 LoRa transceiver, the left-to-right order of the PHY header is as follows: a single payload `Length` byte, followed by a nibble for the CR and MAC CRC presence, the High Nibble (HN) of the header checksum, and finally the Low Nibble (LN) of the header checksum[3]. An overview of these fields is given in Figure 7.2.4.

---

are decoded correctly. Hence the increased robustness measures.

[3]Only the 5 LSBs of the checksum byte appear to be used in practice.

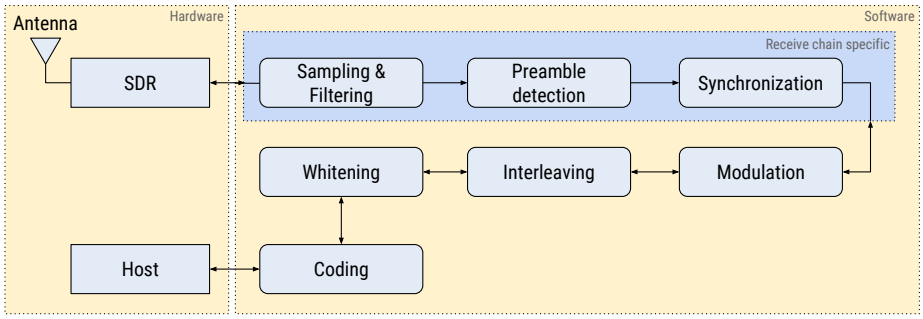


Figure 7.3.1: An overview of the LoRa decoder components, showing the various logical processing blocks required for the transmission and reception of baseband I/Q samples with an SDR.

7.3 Software demodulator

We have implemented the complete PHY layer of LoRa in software, using the open source GNU Radio signal processing framework. The source code of the decoder is publicly available on Github[4]. An overview of the decoder components is given in Figure 7.3.1.

7.3.1 Detection and synchronization

As a first step in the demodulation process, the receiver must detect the LoRa preamble. To this end, we exploit the repeating property of the preamble by using the Schmidl-Cox algorithm, which defines two quantities $P(d)$ and $R(d)$ as follows [230]:

$$P(d) = \sum_{m=0}^{L-1} (x_{t+m}^{*}\ x_{t+m+L}) \tag{7.3}$$

$$R(d) = \sum_{m=0}^{L-1} |x_{t+m+L}|^2 \tag{7.4}$$

[4] `https://github.com/rpp0/gr-lora`

where L is the length of a symbol, t is the sample index of the complex signal x, and x^* denotes the complex conjugate of x. Next, $P(d)$ and $R(d)$ are used to calculate a timing metric $M(d)$:

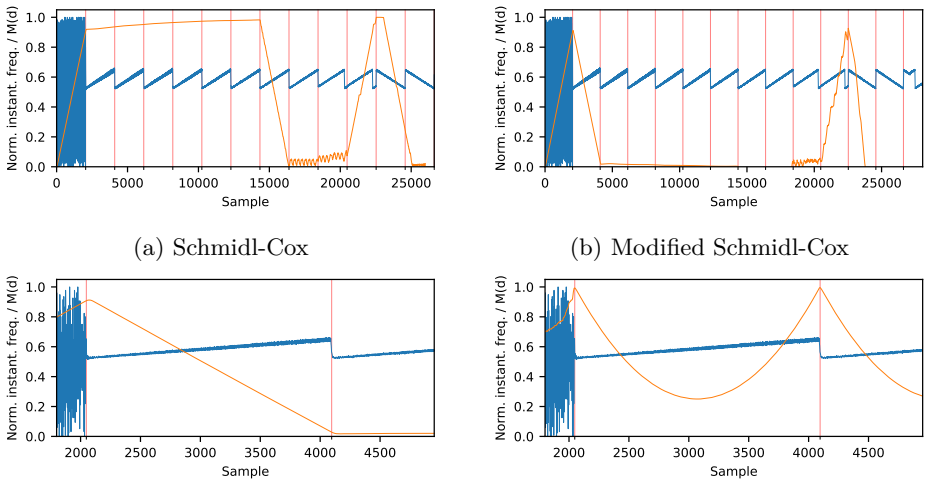
$$M(d) = \frac{|P(d)|^2}{R(d)^2} \tag{7.5}$$

The timing metric $M(d)$ essentially calculates a normalized autocorrelation of length L over two symbols, which will be maximal when two consecutive symbols are encountered in the signal. An added advantage of this approach is that any errors introduced by the channel or SDR (e.g. interference, Carrier Frequency Offset (CFO) and Sampling Frequency Offset (SFO) errors), consistently influence both symbols and therefore minimally affect the result of the correlation. To efficiently compute Equation 7.3 – 7.5 in software, we use Single Instruction, Multiple Data (SIMD) instructions provided by Vector Optimized Library of Kernels (VOLK). Figure 7.3.2 (a) shows the resulting value of the timing metric $M(d)$ when evaluated for a complex LoRa signal. Observe that the function reaches a plateau around sample 2,500, which confirms the presence of a preamble.

Although the Schmidl-Cox algorithm can determine the presence of a preamble effectively, the plateau of the timing metric leads to uncertainty as to the start of the symbol [230]. Wang et al. proposed a variation on the Schmidl-Cox algorithm where $M(d)$ is subtracted with a time-delayed version $M_2(d)$ of itself [273]. Consequently, the plateau becomes a peak as shown in Figure 7.3.2 (b), making it possible to take the arg max of the timing metric in order to estimate the start of the preamble. However, for LoRa signals, this estimate is not sufficiently accurate, as shown in Figure 7.3.2 (c).

To solve this problem, we only use the standard Schmidl-Cox metric with a threshold to assert that the second symbol window is located anywhere inside the preamble. Next, we generate an ideal, locally synthesized base chirp and calculate its instantaneous frequency $\omega_l(t)$, as well as the instantaneous frequency of the received LoRa signal $\omega(t)$. Finally, a sliding window normalized cross-correlation is performed, and the index of the maximum value of the cross-correlation is chosen as the starting point of the symbol:

$$\text{symbol start} = \underset{i \in \{0,1,\ldots L\}}{\arg\max} \; (\omega_l \star \omega)(i) \tag{7.6}$$



(a) Schmidl-Cox

(b) Modified Schmidl-Cox

(c) Modified Schmidl-Cox (zoomed)

(d) Our sync. algorithm (zoomed)

Figure 7.3.2: Normalized instantaneous frequency of a LoRa signal (blue) and the normalized Schmidl-Cox timing metric $M(d)$ (orange). The vertical red lines indicate the symbol window length L. In (a), the timing metric reaches a plateau upon encountering two consecutive and identical symbols. A modified version (b) results in a single peak at the first symbol, but is insufficiently accurate to determine the start of the preamble (c). Our synchronization using the normalized cross-correlation of the instantaneous frequency (d) shows a sharp peak at the start of each preamble symbol.

The result of this operation is visualized in Figure 7.3.2 (d). Note that by performing the cross-correlation on the instantaneous frequency of the signals instead of their complex values, any CFO errors imparted by the channel are automatically mitigated similarly to the Schmidl-Cox algorithm. Thus, the accuracy of the synchronization is not affected by the CFO. On the contrary, if one would consider the complex-valued signals, a correction of frequency errors introduced by the channel and SDR would have to be performed before the signal can be cross-correlated with the locally synthesized chirp.

At last, in order to verify the correctness of the time synchronization, we threshold against the maximum correlation coefficient between the instantaneous frequency of the locally generated chirp and received chirp. The LoRa frame is rejected if the correlation coefficient falls below a certain tolerance value, since this indicates a failed synchronization or false positive during the detection stage (see Section 7.3.1).

7.3.2 Demodulation

Following time synchronization, the receiver can demodulate the chip values as discussed in Section 7.2. However, the FFT-based demodulation approach specified in [233] is sensitive to frequency offset errors, which cause the magnitudes of the FFT (and thus the chip values) to shift. Therefore, besides time synchronization, an accurate frequency synchronization is also required. Furthermore, this synchronization must be applied for each LoRa channel separately in order to fulfill our requirement of multi-channel decoding motivated in Section 7.1.

Since channelization and separate processing of each channel is an expensive operation to perform in software, and since we would like to retain any frequency offset errors caused by the transmitters so that we can fingerprint them as discussed in Chapter 8, we have developed a novel demodulation technique that is independent of the frequency. Our technique removes the need for frequency corrections and allows to decode LoRa frames in real time on all channels and without additional processing overhead, but at the cost of a reduced robustness compared to the FFT approach discussed in Section 7.2.1. This tradeoff will be discussed in Section 7.4.

In our methodology, we first calculate the instantaneous angular frequency $\omega[t] = \frac{d\varphi[t]}{dt}$. We then smoothen and decimate $\omega[t]$ with a constant decimation factor of $\frac{s_f T}{2^{\mathrm{SF}}}$, where s_f is the sampling frequency. This ensures that the number of samples in $\omega[t]$ is equal to 2^{SF}. Subsequently, the digital gradient of f is calculated:

$$D_t[\omega[t]] = \omega[t+1] - \omega[t] \tag{7.7}$$

This operation can be intuitively seen as a high pass filter on the instantaneous frequency, or as the second order derivative of the phase. Since the frequency of a base chirp linearly increases with k, i.e. $\omega(t) = kt + f_0$, its derivative $\omega'(t)$ is equal to k. For a modulated chirp however, D_t will exhibit a sharp peak at the transition from high to low frequency. If present, the position of the peak indicates the time shift $\hat{t}$. Otherwise, the time shift is equal to 0 (base chirp).

7.3.3 Decoding

In the decoding stage, the chip values are deinterleaved to form codewords of $4 + \mathrm{CR}$ bits. The first 8 codewords of a frame can be directly decoded to form

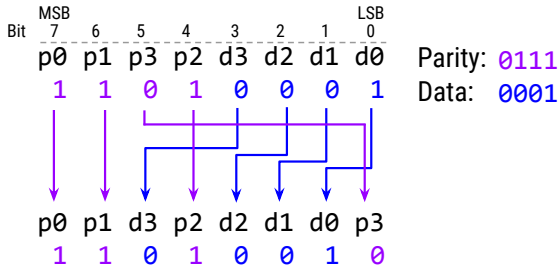


Figure 7.3.3: The bit mapping of an example Hamming codeword used in LoRa (top) to a standard Hamming code (bottom), for a CR of 4 and data `0x1`.

the PHY-layer header. On the other hand, we found that the payload symbols must be dewhitened first. Although the datasheet released by Semtech specifies the usage of a 9-bit LFSR, a different and unknown whitening LFSR appears to be implemented in practice [31, 136]. We have reverse engineered the whitening sequence using the following approach.

If we represent the whitening process as $c_w^{(j)} = c^{(j)} \oplus w^{(j)}$ with c_w the whitened codeword, c the unwhitened codeword, w the output of the LFSR and j the byte index, we can determine $w^{(j)}$ by transmitting a known codeword $c^{(j)}$ and calculating $w^{(j)} = c_w^{(j)} \oplus c^{(j)}$. For example, a payload with all codewords set to zero will result in $w^{(j)} = c_w^{(j)} \oplus 0$, or the whitening sequence itself. Hence, after transmitting a payload consisting of all zeros, the resulting whitening sequence can be retrieved and stored in a lookup table.

With knowledge of the whitening sequence w, the last step after dewhitening is to perform Hamming decoding on the codewords. We found that in LoRa transceivers, the data bits are positioned at bit indices 0, 1, 2, and 3 of a byte. This is in contrast to standard Hamming which uses indices 1, 2, 3, and 5 as data bits. This mapping is graphically depicted in Figure 7.3.3. After extraction of the data bits and error correction or detection based on the present parity bits, the demodulator outputs the data to a UDP socket for further processing by higher-layer applications.

7.3.4 Clock drift correction

The crystal oscillators in the SDR and LoRa transceiver inherently have a relative and unknown clock drift, which causes a loss of synchronization over time. This is especially problematic for long payloads in combination with higher SFs due to the symbol lengths in these configurations. To correct for the clock drift, we propose a blind estimation technique[5] that exploits oversampling of the transmitted signal at a rate of N.

As a first step, our technique requires an accurate initial acquisition of the timing offset using the algorithm discussed in Section 7.3.1. Assuming that the timing error *per symbol* Δt, measured in number of samples, satisfies the inequality $|\Delta t| < \frac{N}{2}$, we can determine Δt as follows:

1. The symbol is demodulated normally as described in Section 7.3.2, in order to obtain the chip value i and the time shift $\hat{t}$.
2. At the receiver, a locally generated ideal upchirp is now modulated using i, which introduces a time shift of $\hat{t}_l$ on the local signal (see Section 7.2).
3. Since the locally generated chirp is not subject to relative clock drift, the timing error is: $\Delta t = \hat{t}_l - \hat{t}$. The receiver can now correct $\hat{t}$ by adding a time offset of Δt to the received signal.

Note that if the timing error for a single symbol $|\Delta t| \geq \frac{N}{2}$, the decoder will incorrectly determine the chip value i and therefore propagate the error to subsequent symbols. For this reason, we interpolate from $\hat{t}$ to $\hat{t} + \Delta t$ rather than setting the value directly. This mitigates the effect of single-symbol demodulation errors on the clock drift correction algorithm. A higher oversampling rate N allows for more fine-grained timing error corrections at the cost of increased processing overhead.

7.4 Evaluation

Our demodulator was evaluated in a lab setup using different SDR models and LoRa hardware. More specifically, we have performed tests with the RN2483,

[5] Although the LoRa patent specifies the usage of "pilot" symbols for tracking timing [233], these symbols appear to be absent in practice. We therefore need to make use of blind estimation techniques.

SX1272 and RFM96 configured as transmitters and the Ettus B210 USRP, HackRF, and RTL-SDR configured as receivers. Each test was performed using a carrier frequency of 868.1 MHz, a sample rate of 1 Msps, and a distance between the transmitter and receiver of about 1 meter. The source code required for reproducing the test results and datasets from the accuracy experiments are publicly available on Github[6].

7.4.1 Compatibility

In a first experiment, we evaluated the compatibility of our decoder with hardware LoRa transceivers. Here, we used the USRP as the receiver and RN2483, SX1272 and RFM96 as transmitters. The compatibility was evaluated by transmitting the payload "`0123456789abcdef`" 100 times for all possible combinations of CR and SF, and checking the number of correctly decoded frames. A configuration is considered compatible when transmitted LoRa frames are consistently correctly decoded under ideal channel conditions and a high SNR.

The results of this experiment are shown in Table 7.4.1. Observe that the only incompatible configurations are SF 11 and SF 12 for the RFM96 transceiver. After a manual inspection, we found that the cause of this incompatibility is due to the fact that the RFM96 transceiver does not enable the reduced data rate mode as mandated in the LoRa specification [235, p. 28]. In fact, even when transmitting a message from the RFM96 to either the hardware SX1272 or RN2483 hardware transceivers, the resulting frame is always corrupted for SF 11 and SF 12. When we manually disabled reduced data rate mode in our decoder, we achieved full compatibility with all devices. This confirms that the issue lies within the hardware implementation of the RFM96, and shows that our decoder can be used to troubleshoot incompatibilities between LoRa devices from different vendors.

7.4.2 Accuracy

In the accuracy experiments, we measured the Packet Error Rate (PER), i.e. the ratio of erroneous packets received over the total number of packets transmitted, for a fully compatible transmitter configured with SF 7 and CR 4. We also artificially introduce two types of channel distortions to the I/Q signal at the receiver, namely Gaussian white noise and CFOs.

[6] `https://github.com/rpp0/lora-decoder-paper`

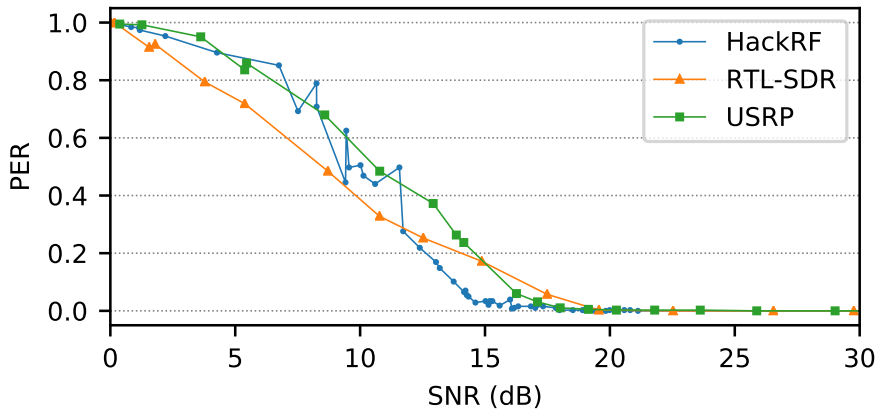
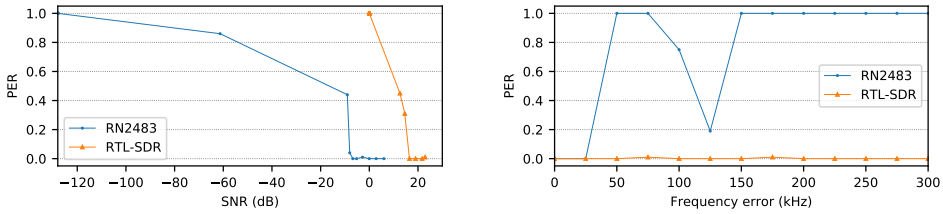


Figure 7.4.1: The effect of artificial Gaussian white noise on the accuracy of the decoder. The PER of our decoder for all evaluated SDRs approaches 0 around a SNR of 20 dB.

Figure 7.4.1 shows the effect of artificial Gaussian white noise on the decoding accuracy when using a SF of 8 and CR of 4, the RN2483 as a transmitter, and the HackRF, RTL-SDR and USRP SDRs as receivers. The receivers were each configured with a receive gain of 10 dB, and the transmit power of the RN2483 was configured to 1 dB. Note that even so, the effective receive gain between the SDRs differs due to their different hardware and antenna designs. From the figure we can derive that a SNR around 20 dB is at least required in order to obtain a PER of 0 for all SDRs.

In Figure 7.4.2 (a), we used the USRP to add Gaussian noise at the *transmitter* instead of the receiver in order to compare our decoder (RTL-SDR) with a hardware LoRa transceiver (RN2483). For this experiment, both receivers were placed at an equal distance from the USRP, and 100 frames with a payload of "`0123456789abcdef`" were transmitted to calculate the PER. Observe that the RN2483 is still capable of receiving frames well below the noise floor, whereas our decoder stops receiving any frames at 0 dB SNR. At this point, the transient of the gradient cannot be detected by our gradient-based decoding algorithm due to the presence of excess noise.

On the other hand, when we introduce a CFO instead of Gaussian noise to the transmitted signal, our decoder outperforms the hardware for a CFO larger than



(a) (b)

Figure 7.4.2: Comparison between the RTL-SDR using our software decoder versus a hardware RN2486 LoRa transceiver in terms of resistance to Gaussian noise (a) and resistance to frequency offset errors (b). Due to the properties of our gradient-based decoding algorithm, our decoder is shown to be unaffected by frequency errors at the cost of a significantly higher sensitivity to Gaussian noise compared to the RN2483.

50 kHz or when transmitting at a different channel as shown in Figure 7.4.2 (b)[7]. The hardware device must be retuned in order to capture frames from a different channel, whereas our decoder is capable of capturing frames from multiple channels simultaneously.

Based on these observations, we conclude that our decoding technique essentially trades flexibility (i.e., being able to decode multiple channels simultaneously at no additional cost) for an increased sensitivity to Gaussian noise. Although the increased sensitivity to noise reduces the functioning range of the receiver, the flexibility of our approach allows for building a fully compatible and multi-channel LoRa monitoring device at a very low cost. In Chapter 8, we will use the discussed techniques to extract PHY-layer features from LoRa signals and fingerprint individual transceivers.

7.5 Related work

A high-level system architecture overview of LoRa is given by Centenaro et al. [45]. Sikken has provided a general overview of the LoRa frame structure and decoding stages, but does not discuss the low-level details of the modulation [242]. Goursaud et al. have provided a detailed and formal analysis of LoRa modulation,

[7]Note that at 125 kHz, the PER of the RN2483 decreases momentarily back to 0.2. This could possibly be explained by the CFO causing the FFT peak to wrap back to an approximately correct position.

but do not discuss the interleaving, whitening and coding [108].

A first complete analysis of the LoRa PHY was presented by Knight [136]. However, in their work, it is assumed that the dewhitening operation is performed before deinterleaving, that the header is whitened, and that a different interleaving pattern is used than the diagonal interleaving specified in [233]. Based on our own experiments, we concluded that these claims are inaccurate. A decoder named `gr-lora` was developed based on their analysis, but appears to only be able to decode short frames transmitted without a header. We believe this limitation is the result of errors made during their reverse engineering process.

Other software LoRa decoders for SDRs have been developed by Blum et al. [31] and by Project Sdrangelove [223]. However, we were unable to decode any frames transmitted by hardware LoRa transceivers using these decoders. Finally, at the time of writing, none of the decoders in other works have developed a clock drift correction algorithm for decoding long LoRa frames.

Besides work on LoRa, GNU Radio based decoders have previously been developed to decode protocols such as Global System for Mobile Communications (GSM) [9, 144], Long Term Evolution (LTE) [75], and 802.11 [28, 271] using SDRs.

7.6 Chapter conclusions

In this chapter, we have provided an in-depth examination of the LoRa PHY layer (**C6**), and demonstrated our open source LoRa software decoder, which is implemented using the GNU Radio framework (**C7**). Our decoder can be used to receive LoRa frames in real time with inexpensive SDRs such as the RTL-SDR, and is able to interoperate with existing LoRa transceivers.

A set of frequency-invariant techniques for the detection, demodulation, and clock drift correction of LoRa frames were furthermore introduced, which allow to decode multiple channels simultaneously in real time, at the cost of an increased sensitivity to noise compared to COTS LoRa radios. More specifically, our evaluation shows that a SNR of at least 20 dB is required for the PER to approach 0. However, an additional benefit of our methodology is that any frequency offset errors imparted by the device are preserved, which allows us to fingerprint LoRa devices. This aspect will be discussed in the next chapter of this thesis.

Table 7.4.1: Results of the compatibility experiment, showing the LoRa transceivers and configurations for which our decoder was able to successfully decode frames.

Transceiver	SF 7, CR 1	SF 7, CR 2	SF 7, CR 3	SF 7, CR 4	SF 8, CR 1	SF 8, CR 2	SF 8, CR 3	SF 8, CR 4	SF 9, CR 1	SF 9, CR 2	SF 9, CR 3	SF 9, CR 4	SF 10, CR 1	SF 10, CR 2	SF 10, CR 3	SF 10, CR 4	SF 11, CR 1	SF 11, CR 2	SF 11, CR 3	SF 11, CR 4	SF 12, CR 1	SF 12, CR 2	SF 12, CR 3	SF 12, CR 4	Implicit
Custom module (RN2483)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	N/A
Pycom LoPy (SX1272)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	N/A
Dragino Pi HAT (RFM96)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓*

*Only device in our possession with an API for sending frames in implicit mode.

You have to fight for your privacy or you lose it.

Eric Schmidt

8

Physical-Layer Fingerprinting of LoRa devices using Supervised and Zero-Shot Learning

8.1 Introduction

PHY-layer device identification is a technique through which it is possible to uniquely identify devices by looking at small differences in their analog Radio Frequency (RF) signals. These differences are caused by imperfections of their analog hardware components, which are often unique and allow to create a fingerprint of the device [38, 116, 210]. PHY-layer device identification has been proposed for various purposes such as access control and the detection of cloning and wormholes [207]. Several works have demonstrated the feasibility of PHY-layer device fingerprinting for a variety of wireless technologies such as Radio Frequency Identification (RFID) or High Frequency (HF) and Very High Frequency (VHF) transmitters (e.g. [68, 70]). However, limited research has been performed in context of fingerprinting devices that implement newer LPWAN-oriented protocols.

As mentioned earlier, LoRa has become one of the most promising wireless technologies for IoT appliances in LPWANs, suitable for devices that need to infrequently send small amounts of data over long distances. In terms of security and privacy, the current LoRaWAN MAC specification provides built-in data confidentiality, integrity, and device authentication. However, it does not offer privacy. More specifically, all LoRaWAN messages contain a unique MAC address that can identify the sender device [251]. Nonetheless, one could envision a future, privacy-preserving version of the LoRaWAN protocol to support applications where a certain degree of privacy is needed. For example, the MAC address could be periodically randomized similarly to current Wi-Fi implementations. Alternatively, the LoRa PHY layer itself can be used in conjunction with other MAC-layer protocols. However, even if a privacy-preserving MAC-layer protocol is used, it remains unclear whether adversaries could still identify LoRa devices based on their analog RF signals alone.

In this chapter we tackle exactly the problem outlined above and investigate whether an adversary can identify, locate or track LoRa devices regardless of any privacy-preserving mechanisms used in the higher layers. To this end, we propose a fingerprinting methodology that applies supervised machine learning techniques to radio signals in order to distinguish between multiple known transmitters (**C8**). To demonstrate its feasibility, we apply this technique to devices that use the LoRa modulation scheme. Our methodology is inspired by recent advances in image and speech recognition, where state-of-the-art performance was achieved using *raw* data [183, 245, 249]. Our classifier achieves 59%–99% accuracy for 22 LoRa devices, even when devices with identical chipsets are far

away from the fingerprinter. Additionally, our methodology can be applied to any part of the frame in contrast to previous works, where only part of the frame (e.g. the preamble [116, 205, 206]) is used for fingerprinting. Our technique is fully automated, passive, does not rely on underlying properties of the modulation scheme, and can be performed with sample rates as low as 1 Msps. This is achieved by using the raw I/Q data as a high-dimensional feature vector, as opposed to the conventional low-dimensional vector of features based on signal error measurements [38] or statistical measurements (e.g. skewness, variance, kurtosis, etc.) [205, 206]. Moreover, we show how our classifier can be extended to recognize *previously unseen* transmitters. This is achieved by applying zero-shot learning techniques, where a classifier is evaluated without any available training data for certain unknown classes. We performed a number of experiments that reveal the effects of (i) the distance between the fingerprinter and the LoRa device, (ii) changing channel conditions, and (iii) the sample rate on the classifier's performance.

8.2 Fingerprinting LoRa on the PHY layer

PHY-layer fingerprinting leverages inherent small artifacts in the analog RF signals transmitted by wireless devices to uniquely identify them. These differences are caused by distinctive hardware or antenna designs, or imperfections introduced in the analog hardware components during the manufacturing process [38]. The PHY-layer fingerprint of any wireless device is also influenced by other factors such as the channel conditions.

8.2.1 Features

As a first step in the fingerprinting process, a combination of radiometrics that are sufficiently discriminative to uniquely identify the devices needs to be selected. We will henceforth refer to such radiometrics as "features". Similarly to MAC-layer fingerprinting (see Chapter 6), the features selected for PHY-layer fingerprinting should exhibit high intra-class stability and high inter-class variability. One the one hand, high intra-class stability entails that features extracted from the same LoRa device should be consistent for multiple transmissions, sufficiently resistant to changing channel conditions, and be independent of the physical location of the device. This ensures that all radio transmissions sent by

a single device are mapped to the same fingerprint. On the other hand, high inter-class variability entails that features contain sufficient entropy, such that transmissions by *distinct* radio chips can be easily distinguished.

8.2.2 Classification

In traditional approaches for classifying devices based on PHY-layer features, N acquisitions of several features are reduced in dimensionality and then averaged into one final, low-dimensional feature vector. This feature vector is subsequently learned by a classifier in order to construct a template for a specific device [67, 71, 116, 210]. Finally, the template is matched with the device by means of a similarity metric, such as Euclidian distance [210], entropic distance [116], or KL-divergence [271].

Although such approaches allow to distinguish between devices with high accuracy, we can identify several shortcomings. First, N is determined empirically, which makes these approaches hard to generalize for different channel conditions, hardware, or modulation schemes. Second, the selection of which (combination of) feature(s) to use depends on the expertise of the researcher. Such a manual selection of features, which is performed in an iterative way, can influence the result of the classifier on multiple levels. Some examples of these influential choices include the selection of using the transient as a feature itself, followed by sub-choices such as which sample rate to use, with which algorithm to extract the transient [141], the time-synchronization between the N acquisitions observed from the device to fingerprint, and which metric to use for defining the similarity between two devices. A possible justification for choosing low-dimensional features is that they require less computing power and training data, whereas high-dimensional features suffer from the "curse of dimensionality" [23]. By manually pre-selecting features, the classifier thus converges faster and requires less training data, but suffers from the shortcomings discussed above.

We propose a novel per-symbol classification methodology that aims to overcome some of these shortcomings by using high-dimensional features learned in an automated fashion. Our methodology is inspired by recent advances in Computer Vision (CV), more specifically in image and speech recognition. Here, state-of-the art classification results are achieved by learning on raw data, such as the image pixels or time-domain waveform samples rather than manually selected features [143, 245, 255]. To limit the dimensionality and to ensure payload independence of the classification, we apply our methodology to the information contained in each i-th LoRa symbol $s(t)^{(i)} \dots s(t)^{(n)}$ separately for a frame con-

sisting of n symbols. The extraction of features from these symbols will be discussed in Section 8.3.3.

Supervised classification

In our supervised classification approach, the fingerprinter is given a reference set of F-dimensional input features $x^{(1)} \dots x^{(n)} \in X^{n \times F}$ extracted from $s(t)^{(1)} \dots s(t)^{(n)}$, and corresponding C-dimensional class label tensor $y^{(1)} \dots y^{(n)} \in Y^{n \times C}$. Here, the "class" refers to a single radio chip of a device, where the corresponding "class label" can be "LoRa 1–22". Given a model parameterized by the learned variables θ, the following loss function $L(\theta)$ is minimized during an initial training phase:

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{f=1}^{F} y_f^{(i)} \log(h_\theta(x_f^{(i)})) \tag{8.1}$$

In this equation, the hypothesis function h_θ outputs the predicted class for each of the different models given the learning model parameters θ, and the loss function minimizes the cross entropy between the predicted and true class labels.

After the training phase, the classifier extracts features and evaluates the model for each symbol in a LoRa frame in order to predict the most likely class. The class of the transmitter can be determined by performing majority voting on the symbols of the frame.

A requirement for accurate results under this classification approach is that a sufficiently large reference set of training samples for each of the device classes must be available. Furthermore, the accuracy depends on the quality of the training samples. A device should ideally be fingerprinted under different channel conditions. For example, by acquiring samples over a long period of time in order to prevent overfitting of the model on specific channel conditions. In Section 8.3.4, we will evaluate the effect of different channel conditions on the accuracy of our classifier.

Zero-shot classification

When fingerprinting a random observed radio signal, one typically would not possess a reference database of training samples for the associated (unknown) transmitters. In this case, the difficulty of the classification task is increased. Techniques that can deal with the absence of training data for some set of unknown classes have been given several names over different domains: zero-shot classification [157, 249] (image recognition), semi-supervised anomaly detection [187] (anomaly detection), or open set recognition (biometrics authentication). We will use the term zero-shot classification henceforth in this work.

Despite not having training data for unseen classes, the fingerprinter can still learn the *attributes* for a given set of known classes [146]. Such attributes can be interpreted as high-level, semantically meaningful properties that are used to describe a new class [145]. For example, if the attribute "CFO" of an unknown instance differs by at least T Hz from a known class, it could be considered as a new, unseen class[1]. The difficulty here is the choice of the threshold T, which must be large enough to ignore noise and small enough to classify an unseen instance as a new class.

Socher et al. [249] proposed a methodology to perform zero-shot learning on an image recognition problem, where only a subset of all observed classes is known. Since our LoRa fingerprinting objective is similar, we will apply a methodology similar to theirs, with ideas incorporated from the work of Lu et al. [157].

Before applying zero-shot classification to PHY-layer fingerprinting, we first need to determine whether an observed symbol belongs to a known class or to a previously unknown class. This can be considered as an outlier detection problem, where the unknown classes are outliers. To accomplish this goal, we have modeled the output values of the supervised classifier under a mixture of K multivariate Gaussian distributions similarly to the work of Socher et al. [249]. Here, K is the number of known classes, which is also equal to the number of output neurons. The parameters μ_k and σ_k of each Gaussian $\mathcal{N}_k(\mu_k, \sigma_k)$ are determined by respectively taking the mean and standard deviation of the output values after feeding the input features of a known class k to the neural network. Then, we

[1] A similar analogy can be made in the field of biology, where an unknown population can be considered as a different species if its features deviate sufficiently from an already known species.

perform outlier detection by evaluating the indicator function:

$$\mathbb{1}\{\sum_{k=1}^{K} h_\theta(x)\mathcal{N}_k(\mu_k, \sigma_k) < T\} \tag{8.2}$$

where $h_\theta(x)$ is the output of the hypothesis function parameterized by θ given x, and T is an outlier tolerance threshold.

Next, the actual classification can be performed. If a symbol is not an outlier, it should belong to a known class. Therefore, it can be classified using the supervised classification approach from Section 8.2.2. In the other case, we used the unsupervised DBSCAN [83] algorithm to cluster symbols transmitted by the same unknown class together. The ϵ parameter of the DBSCAN algorithm, which indicates the maximum distance between two points for them to be considered as in the same neighborhood, was set to the mean of the minimum Euclidean distance between all combinations of centroid pairs in $\{\mu_k \dots \mu_K\}$. This ensures that symbols transmitted by different devices are appropriately mapped to different clusters, while symbols transmitted by the same device are mapped to the same cluster.

8.2.3 Learning models

To model the observed features for our supervised and zero-shot classifiers described in Section 8.2.2, several approaches could be considered. In this work, we examined MLPs and CNNs (see respectively Sections 3.3.1 and 3.3.2 in the Preliminaries part), due to their success in similar classification tasks for other domains such as facial and speech recognition. Additionally, we briefly examine SVM-based models due to their popularity in previous PHY-layer fingerprinting works.

Multilayer Perceptron

Our MLP model consists of one fully connected hidden layer with ReLU activation functions, and one fully connected output layer. Hence, the input features are mapped to the output device classes using the hypothesis function:

$$h_\theta(x) = \sigma(\text{ReLU}(xW_1 + b_1)W_2 + b_2) \tag{8.3}$$

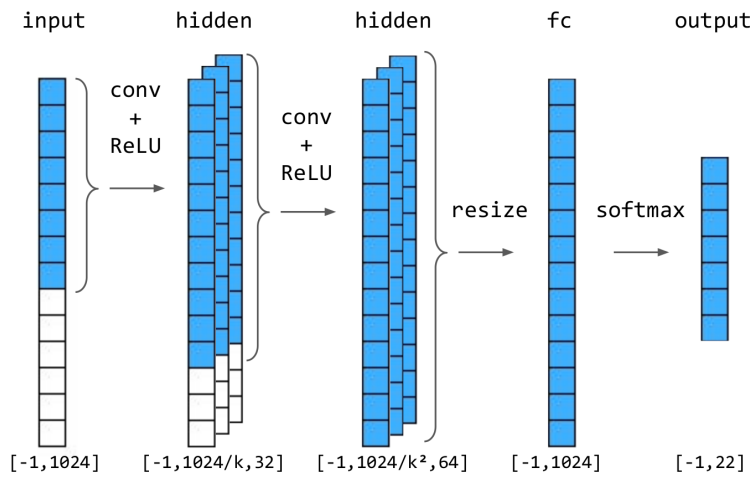


Figure 8.2.1: Architecture of our CNN model for 1024-dimensional inputs. The dimensions of each layer are indicated between square brackets.

Where σ denotes the softmax function, and W and b respectively denote the weights and biases of the neurons. The softmax function scales each output from the classification of $x^{(i)}$ to form a discrete probability distribution for each $y \in Y$. Thus, the model learns the estimated probability that the symbol was transmitted by the device with label y.

Convolutional Neural Network

Recall that CNNs learn parameters to cross-correlation filter layers, which allows them to identify both low level details at shallow layers and high-level features at deeper layers. Our CNN fingerprinting architecture consists of two hidden 1D convolution layers with kernel size k and ReLU activation functions, followed by a fully connected layer and softmax function for performing the classification, as shown in Fig. 8.2.1.

C-Support Vector Classification

SVMs are trained to find an optimal hyperplane, in which the margin of separation between two classes is maximized [118]. A different cost function from the previously discussed models is used, which takes on the following form (using the original notation) [48]:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \tag{8.4}$$

constrained by:

$$y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \text{ and } \xi_i \geq 0, i = 1, \ldots, l \tag{8.5}$$

In our experiments, we have used the SVM implementation of `sklearn` [193], which uses a one-vs-one scheme to perform multiclass classification.

8.3 Implementation and results

8.3.1 Laboratory setup

Our laboratory setup comprises inexpensive COTS hardware such as SDRs including an Ettus Research USRP B210[2] and a HackRF[3], antennas and a commercial standard laptop. All our experiments were performed with 5 different types of LoRa devices which all transmit at 868 MHz. Table 8.3.1 gives an overview of all LoRa classes including their chipset, identifiers and quantity. We designed a custom power supply board, which remained fixed in the same position, where we plugged in the LoRa transceivers before starting each of the fingerprinting experiments. This ensures that our results are minimally influenced by the distance between the SDR and LoRa devices.

[2] `https://www.ettus.com/`
[3] `https://greatscottgadgets.com/hackrf/`

Table 8.3.1: Overview of all LoRa devices involved in the experiments and their chipset, identifiers, and quantity.

Device	Chipset	Identifiers	Qty.
Custom board	RN2483	LoRa 1–3	3
Pycom LoPy	SX1272 [202]	LoRa 4	1
Dragino LoRa/GPS HAT	RF96 [81]	LoRa 5	1
Adafruit Feather 32u4	RF96 [7]	LoRa 6	1
RN2483 breakout board	RN2483 [15]	LoRa 7–22	16

8.3.2 Signal acquisition

To obtain samples for our symbol classifier, each of the 22 LoRa devices used in the experiments was configured to continuously transmit frames with a 4-byte payload, using coding rate $4/8$ and SF 7. This configuration yields 36 symbols per frame. The payload bytes were randomized to ensure that the resulting symbol values are random as well, thus removing any bias due to the payload data. Both the training and test samples from the LoRa transmissions were acquired with the B210 USRP tuned to a 868 MHz carrier, and a sampling rate that varies from 1 Msps to 10 Msps. To mitigate the effect of the USRP's DC bias filter, we configured our receiver program to capture signals at 868.1 MHz instead of 868 MHz.

To minimize the impact of translation variance of the features, each symbol needs to be time synchronized. For this purpose, we have used our custom LoRa decoder that was discussed in Chapter 7. Recall that our decoding algorithm preserves frequency offset errors imparted by the transmitters, so that we can use these as a feature in the classification while still being able to time-synchronize to the symbols. After extracting all synchronized symbols from the frame, the classifier needs to be trained on their features in order to distinguish between different LoRa transmitters. However, feeding the symbols directly to the model would increase the required training data by a factor of at least 2^{SF}, since each symbol can have 2^{SF} possible values. Furthermore, if by chance a certain symbol value occurs more frequently for a specific device in the training set, the model will overfit and incorrectly assume that this symbol value implicates a higher classification probability for that device.

The following approach was applied to mitigate this problem. We first calculate the ideal cyclic shift k of the symbol, i.e. its demodulated value. Then, we modulate the symbol with value $\overline{k} = -k$. Given that the number of chirps per

Table 8.3.2: Overview of the datasets used in this work.

ID	# Symbols	Sampling rate	Date
I	495,216	1 Msps	January 27, 2017
II	124,740	1 Msps	January 30, 2017
III	497,595	2 Msps	January 17, 2017
IV	127,476	2 Msps	January 27, 2017
V	221,622	5 Msps	February 2, 2017
VI	55,908	5 Msps	February 3, 2017
VII	219,718	10 Msps	January 31, 2017
VIII	56,528	10 Msps	February 3, 2017

symbol is given by c and that $s(t+nT) = s(t)$ (see Section 7.2.1), the modulated symbol $\hat{s}(t)$ becomes:

$$\hat{s}(t) = \begin{cases} s(t + \frac{c-(k+\overline{k})}{c}T), & \text{if } t < \frac{k+\overline{k}}{c}T \\ s(t - \frac{k+\overline{k}}{c}T), & \text{otherwise} \end{cases} = s(t) \tag{8.6}$$

Hence, as a result of this operation, each modulated symbol $\hat{s}(t)$ is transformed back to the base chirp $s(t)$, and the errors introduced by the hardware are preserved.

An overview of all collected datasets is given in Table 8.3.2. We will refer to these datasets in future sections of this work using their respective Roman numeral label.

8.3.3 Classifier training

For implementing and training the models described in Sect. 8.2.3, we use the `tensorflow` machine learning framework presented by Abadi et al. [4]. Before the training process, the entire dataset of collected features and labels is uniformly randomized. Then, a training set of 10,000 symbols and a cross validation set of 10,000 symbols are fetched from the dataset for evaluation during training. A test set of 1,500 symbols is used for evaluation after training. The randomization process ensures that the training set is not biased by any particular device.

The LoRa symbols from each dataset are converted to a feature tensor. As mo-

tivated in Sect 8.2.2, the I/Q signal could be used directly as a high-dimensional feature. Since `tensorflow` cannot operate on complex values however, the I/Q value can be decomposed into its amplitude and phase. In our approach, we instead use the magnitude of the Fourier transform of the signal, since we are interested in the CFO as a distinguishing features between the LoRa transmitters. Considering the LoRa configuration parameters and receiver sample rate, the matrix of feature tensors for n symbols is $X \in \mathbb{R}^{n \times m}$, with

$$m = 2^{SF} \frac{f_s}{BW} \tag{8.7}$$

where SF is the spreading factor, f_s is the sampling rate of the receiver, and BW is the bandwidth.

Next, each used feature tensor $x^{(i)}$ was z-score normalized to prevent extreme gradient values from occurring during the training phase:

$$\hat{x}^{(i)} = \frac{x^{(i)} - \mu_{x^{(i)}}}{\sigma_{x^{(i)}}} \tag{8.8}$$

This normalization also helps to reduce the effect of the absolute magnitude on classification, which is undesired since we do not distinguish devices based on their transmission power or physical location.

Finally, in each training step we feed a mini-batch of $b < n$ tensors to the classifier, and periodically log training set accuracy, test set accuracy, and cost function.

8.3.4 Fingerprinting experiments

Using the models defined in Section 8.2.3, we have trained and evaluated classifiers to distinguish between vendor models as well as individual LoRa devices. Furthermore, we have investigated the effect of the sample rate, distance, and channel conditions on the subset accuracy, macro-averaged precision, and macro-averaged recall of the classifier. Here, the subset accuracy is the proportion of labels with *entirely* correct predictions [99], i.e.:

$$\text{subset accuracy} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h_\theta^{(i)}(x^{(i)}) = y^{(i)}\} \tag{8.9}$$

where $\mathbb{1}$ is the indicator function that is equal to one when $argmax\ h_\theta^{(i)}(x_i) = y^{(i)}$. Macro-averaged precision and recall are defined as:

$$\text{macro-averaged precision} = \frac{1}{|Y|}\sum_{y=1}^{|Y|}\frac{tp_y}{tp_y + fp_y} \tag{8.10}$$

$$\text{macro-averaged recall} = \frac{1}{|Y|}\sum_{y=1}^{|Y|}\frac{tp_y}{tp_y + fn_y} \tag{8.11}$$

with tp_y the number of true positives for label y, fp_y the number of false positives for y, and fn_y the number of false negatives for y. In each of the following experiments, these metrics were calculated using `scikit-learn` [193].

Supervised classification experiment

In a first experiment, we have trained our models with labeled instances for the purpose of distinguishing between different device vendors and different devices of the same type. Intuitively, the former should be easier to distinguish as the analog hardware layout and design between various vendors should differ more compared to multiple devices of the same vendor model. Multiple devices of the same vendor model only differ as a result of manufacturing variations.

Based on our findings, the crystal oscillator of the radio chip is especially susceptible to fingerprinting, since small differences in the oscillation frequency of the crystal will introduce a measurable CFO error [271]. Indeed, the datasheet of the SX1272 LoRa radio chip states that the RF center frequency is derived from a reference crystal oscillator with a finite frequency precision [235]. Although such errors are benign in ordinary communications due to the tolerance levels of the LoRa modulation scheme, they can be exploited to fingerprint individual radio chips.

Contrary to previous works, where the CFO is explicitly measured as a scalar based on (averaged) samples of the signal [38, 69, 271], our approach uses the raw signal directly. As such, the CFO error manifests itself as a constant drift of the phase. This drift can be observed when examining the averaged unwrapped phase values of several symbols transmitted by the same radio chip (see Fig. 8.3.1 (a)). However, since the phase difference is small for each of the many sample

points, it is difficult for our classifier to learn this feature[4]. We mitigate this issue by transforming the signal to the frequency domain using the FFT and taking the magnitude. As a result of this transformation, the phase drift will shift the entire frequency spectrum, resulting in a large frequency difference for a few sample points of the FFT (see Fig. 8.3.1 (b)), which is easier for a classifier to learn. In both figures, a small CFO error can be observed between devices which have identical radio chips, whereas a large error is observed between different radio chip models, i.e. between the SX1272 (yellow), RF96 (cyan, magenta), and RN2483 (remaining colors). Finally, note that besides the CFO, other distinctive frequency components may be discovered by the classifier.

The datasets used for evaluating our classifier are shown in Table 8.3.2. For the test sets under identical channel conditions (**I, III, V, VII**), a disjoint and randomized subset from the same dataset was selected. The test sets under different channel conditions (**II, IV, VI, VIII**) were sampled from data sets which were recorded on a different day from the training set. Each model was trained for 10,000 epochs, which depending on the used model corresponds to several hours of training on a Dell Precision T3610 with an Intel® Xeon® E5-1620 v2 CPU (3.70GHz). Table 8.3.3 summarizes the results of feeding these datasets to the MLP, CNN, and SVM classifiers when fingerprinting *individual* chipsets. When fingerprinting the 3 chipset *vendors*, the accuracy was 99-100% for all datasets and classifiers (not shown in the table). Figure 8.3.2 shows a t-SNE visualization of the output weights learned by the MLP model after training on dataset **III**. We can observe several clusters corresponding to the different LoRa devices.

For the results of the remaining experiments that will be presented in the following sections, only dataset **III** and the MLP model were considered. We believe these results are the most interesting to mention, since 2 Msps is the maximum sample rate of low-cost SDR devices, such as the RTL-SDR, and since the MLP model is faster to train while achieving similar or better accuracy compared to the CNN and SVM models.

[4]A projection of the symbol phase to a 2D grayscale image would display the CFO error of a device as a slight difference in brightness.

Table 8.3.3: Accuracy, precision, and recall when fingerprinting individual chipsets using supervised learning with test sets of 1,500 symbols.

Dataset	Model	Accuracy	Precision	Recall	Ident. chan. cond.
I	SVM	68.80%	63.08%	69.37%	Yes
I	CNN	89.40%	90.32%	89.23%	Yes
I	MLP	93.33%	93.32%	93.04%	Yes
II	SVM	53.80%	48.17%	52.09%	No
II	CNN	58.60%	55.94%	58.15%	No
II	MLP	58.67%	52.87%	58.19%	No
III	SVM	76.27%	76.01%	76.05%	Yes
III	CNN	94.27%	95.16%	94.88%	Yes
III	MLP	95.40%	95.61%	95.34%	Yes
IV	SVM	59.53%	62.66%	59.99%	No
IV	CNN	67.60%	73.64%	68.17%	No
IV	MLP	71.47%	75.04%	72.36%	No
V	SVM	83.00%	83.07%	82.77%	Yes
V	CNN	96.53%	97.03%	96.78%	Yes
V	MLP	99.00%	99.03%	98.98%	Yes
VI	SVM	69.33%	67.07%	70.23%	No
VI	CNN	76.80%	82.56%	76.72%	No
VI	MLP	75.07%	74.89%	75.37%	No
VII	SVM	80.93%	80.61%	81.24%	Yes
VII	CNN	97.87%	98.03%	97.96%	Yes
VII	MLP	98.67%	98.77%	98.63%	Yes
VIII	SVM	56.53%	53.02%	57.94%	No
VIII	CNN	60.33%	62.00%	62.22%	No
VIII	MLP	60.80%	58.75%	63.07%	No

Zero-shot classification experiment

A second experiment evaluates our zero-shot classification approach from Section 8.2.2 after training on 10,000 random symbols from dataset **III**. Here, the randomization and training procedures were identical to the previous experiment, except that we excluded symbols belonging to certain classes from the training set. We subsequently observed whether the classifier was able to cluster these unknown classes together. The results of this experiment are shown in Table 8.3.4.

We observed that the accuracy of the zero-shot classification largely depends on which devices are excluded from the training set. For example, in experiment

Table 8.3.4: Accuracy, precision, and recall for the unknown "outlier" classes from the zero-shot classification experiments. The evaluation was performed on 1,500 symbols from dataset **III**.

Experiment	Excluded	Accuracy	Precision	Recall
ZS1	4,5,6	70.98%	75.36%	75.00%
ZS2	4	100.0%	100.0%	100.0%
ZS3	2,3,9,10,11	66.67%	41.45%	38.78%
ZS4	8,12,14,16,21	65.22%	48.55%	53.33%
ZS5	15,17,20	75.00%	63.44%	71.43%
ZS6	7,13,14	88.35%	67.82%	65.00%

ZS1, LoRa 4, 5 and 6 were excluded from the training set. Hence, the model was trained only on devices that have a RN2483 chipset. As a result, the classifier was not able to distinguish LoRa 5 and 6, i.e. both were grouped in the same cluster. This problem can be mitigated by including LoRa devices with similar fingerprints in the training set (see ZS6 in Table 8.3.4).

Effect of sample rate and channel

Ideally, our fingerprinter should be able to classify devices at low sample rates and various channel conditions. Therefore, we investigated the effect of these aspects on the classifier accuracy.

Ramsey et al. found that the fingerprinting accuracy increases with the sampling rate, but does not further improve above the Nyquist frequency [206]. On the contrary, in our experiments we observed that, under identical channel conditions, a sampling rate above the Nyquist frequency (250 KHz) increases the accuracy when devices have similar fingerprints. A higher sampling rate results in a higher granularity of frequency bins of the FFT spectrum, which allows the fingerprinter to detect more fine-grained frequency errors. Perhaps this effect was not noticed in [206] because of the lower number of devices involved in their experiments: the fingerprints could have been distinctive enough at a low sample rate already.

Table 8.3.3 shows the classifier accuracy, precision, and recall when using the learning models with datasets of different sample rates. The MLP classifier for example achieves 93% per-symbol accuracy for 22 LoRa devices under identical channel conditions, with sample rates as low as 1 Msps. However, when the

channel conditions change, the accuracy drops for each of the sample rates. This is to be expected, since the crystal oscillator may undergo temperature changes over time, which causes its frequency to change and subsequently overlap with training data from a different radio chip. When examining the confusion matrix, we observed that the misclassifications indeed mostly occur with neighboring clusters (see Fig. 8.3.2). This issue could be mitigated by providing more training data gathered over an extended period of time or by periodically updating the model (i.e. "adaptive learning") to reflect changes in the channel conditions. Such adaptive learning models could be considered in future work.

Effect of distance

We evaluated how increasing the distance between the LoRa devices and the fingerprinter affects the accuracy of our classifier. For this purpose, we performed a series of experiments within a building, where the fingerprinter was always kept in the same location, whereas the LoRa devices were placed in three different locations. In the first experiment, the LoRa devices were in an adjacent room which is approximately 20 meters away from the fingerprinter (D1). In the second and third experiment, the LoRa devices were placed in a room that is 50 meters (D2) and 100 meters away (D3) from the fingerprinter, respectively.[5]

For the signal test sets collected from D1, D2, and D3, our classifier respectively achieves an accuracy of 94.33% 98.40% and 96.40% after training on signals from the respective location. However, we found that the classifier achieves only 22.00% – 30.60% accuracy when a test set from one location is evaluated on a model that was previously trained on signals from a different location. From this observation we can conclude that the channel conditions significantly impact the accuracy of our classifier. In Section 8.4, we will briefly describe two possible ways to overcome this problem.

8.4 Discussion and implications

Training with artificial noise: Our experiments reveal that the accuracy of our classifier degrades under different channel conditions. In other words, our classifier can distinguish between devices more accurately when the LoRa signals in the training and testing phases are captured under similar channel conditions.

[5]Note that for the second and third experiments we used another antenna for LoRa 4, since the received signal was too weak.

Intuitively, one way to overcome this problem would be to use adaptive learning, which allows the classifier to continuously learn and dynamically adapt the models. However, this approach is susceptible to attacks where adversaries try to maliciously influence the way the classifier learns in their favor such that their signals are eventually considered as the signal of a fingerprinted device. Another possibility to mitigate this problem would be to add artificial noise to the training signals of the classifier, which could be used to simulate varying channel conditions in practice. There exist a wide range of techniques to model and estimate several types of communication channels. Similar techniques are applied in the domain of image recognition. Here, the training images are distorted in various ways to reduce overfitting of the classifier.

Fingerprinting for identification systems: In context of using radiometrics for the authentication of devices, we acknowledge that all (including our) existing PHY-layer identification systems are susceptible to impersonation attacks. Some previous works for instance rely only on extracting features from a small part of the signal, such as the preamble or the sync word for fingerprinting devices. In this case, adversaries could easily mount an attack by simply replaying the preamble or sync word of a signal and then appending their own payload signal. In our approach, the adversary would need to be able to mimic the PHY-layer features of the majority of all the symbols[6] within the entire message in order to identify as a legitimate device. This would be harder to achieve, but is still not secure because when using an SDR, the adversary has complete control over the PHY-layer, which allows to shape the signal to match the fingerprint of a different device.

Choice of learning models: Besides the MLP, SVM, and CNN learning models discussed in this chapter, many other models could have been considered for fingerprinting. Similarly, different hyperparameters, e.g. number of hidden layers, dropout probability, number of neurons, etc. could have been selected. We considered the examination of optimal architectural choices for the models out of the scope of this work. Nevertheless, we believe this would be an interesting and useful subject for future work, that can further increase the accuracy of PHY-layer fingerprinting systems.

[6] Assuming a majority vote is performed on all classified symbols in order to determine the transmitter of the entire message.

8.5 Related work

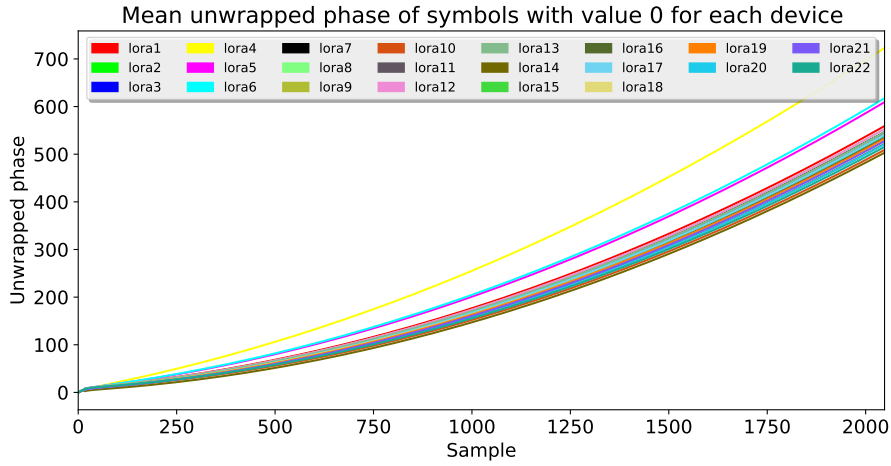
In the literature, there are mainly two types of identification techniques currently being used for RF fingerprinting: (i) those based on extracting features from the signal's transient and (ii) those based on extracting features from the modulated signal. In practice, most previous works consider features extracted using both techniques.

Ureten et al. proposed to extract the envelope of the instantaneous amplitude of 8 WiFi devices [262]. Subsequently, they were able to classify these signals with an error rate of 2% by using a Probabilistic Neural Network (PNN). However, they focused only on distinguishing classes of devices, and used sophisticated equipment with higher sampling rates compared to those used in our work. Rehman et al. developed a transient-based technique for fingerprinting Bluetooth devices both from the same and different vendors [210]. They performed their experiments with 7 Bluetooth transmitters and a sampling rate that varies from 32 Msps to 4 Gsps. Although they achieved slightly better accuracy compared to our work, they used fewer devices, a sampling rate 16 times higher and performed their experiments inside an anechoic chamber, which makes their study less realistic in practice. Remley et al. analysed the feasibility of fingerprinting 802.11 devices by extracting time- and frequency-domain features from 6 devices that belong to 3 different vendors [212]. However, similarly to the work by Rehman et al. [210], they conducted all these experiments in a controlled environment (anechoic chamber) to minimize interference and noise, and used a higher sampling rate. Brik et al. proposed PARADIS, a system to fingerprint 802.11 devices based on errors in the modulated frame. They evaluated their system by collecting frames from 130 devices from the same manufacturer and using them to train an SVM and a kNN classifier. As a result of these experiments, they show that it is possible to distinguish these devices with an accuracy of 99% [38]. The main limitations of this system are that it uses sophisticated equipment with a high sampling rate for capturing the signals, and that it uses modulation-specific features which means the fingerprinting process can not be applied directly in other modulation schemes. Han et al. proposed a technique called Geneprint, which identifies Ultra High Frequency (UHF) RFID devices with an accuracy of 99.68%. Geneprint makes use of features extracted from the signal's preamble using a USRP and a sampling rate of 10 Msps. Ramsey et al. introduced a technique to fingerprint IEEE 802.15.4 devices based on a combination of features extracted from their signal's preamble. This includes the variance, skewness, kurtosis of the instantaneous phase, frequency and amplitude [205, 206]. In [206], they also demonstrated how the fingerprinting can be done with a USRP and PXIe-1085 with a relatively low sampling rate that varies from 5 to 20 Msps.

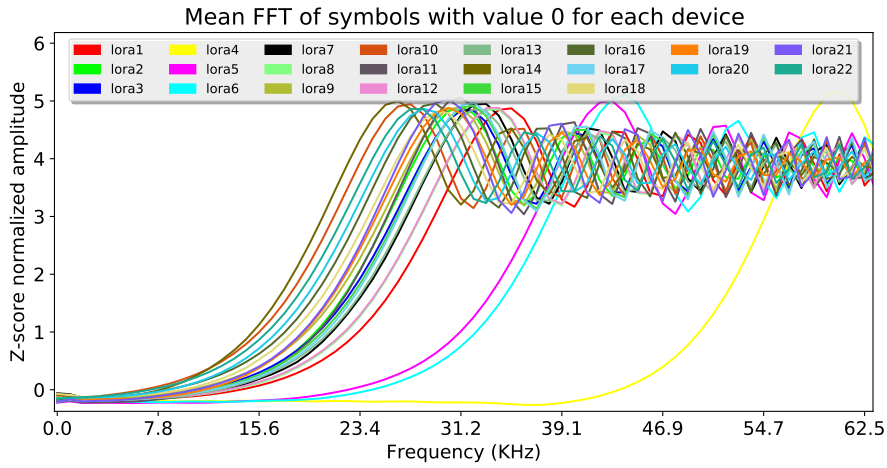
Their classifier implements a multiple discriminant analysis/maximum likelihood (MDA/ML) process in Matlab. The accuracy was reported as 100% in high SNR conditions. However, their experiments involved at most 6 devices. The previous two works are the closest to ours in terms of the selection of the sampling rate. However, they extract features only from the preamble, which may facilitate impersonation attacks as explained in Section 8.4. In this work we are the first to fingerprint LoRa devices on the PHY layer, by extracting features from a single symbol.

8.6 Chapter conclusions

In light of **RG2**, this chapter demonstrates an automated supervised classification approach for PHY-layer fingerprinting that can distinguish LoRa devices by analyzing their RF signals. Our classifier achieves 59%–99% accuracy when fingerprinting identical chipsets, and 99%-100% accuracy when fingerprinting different chipset models. We extended the classifier with zero-shot learning methods to recognize previously unseen classes and achieve 65%–88% accuracy for those classes under similar channel conditions. Our results show that an adversary can identify a transmitter independently of the used modulation scheme or cryptographic mechanisms being used in the higher layers. This can be exploited by an adversary to for example track devices.



(a)



(b)

Figure 8.3.1: Mean instantaneous unwrapped phase (a) and magnitude of the FFT transform (b) of LoRa symbols transmitted by each device. Minor CFO errors can be identified between devices with identical radio chips, whereas large differences are displayed between distinct radio chip models.

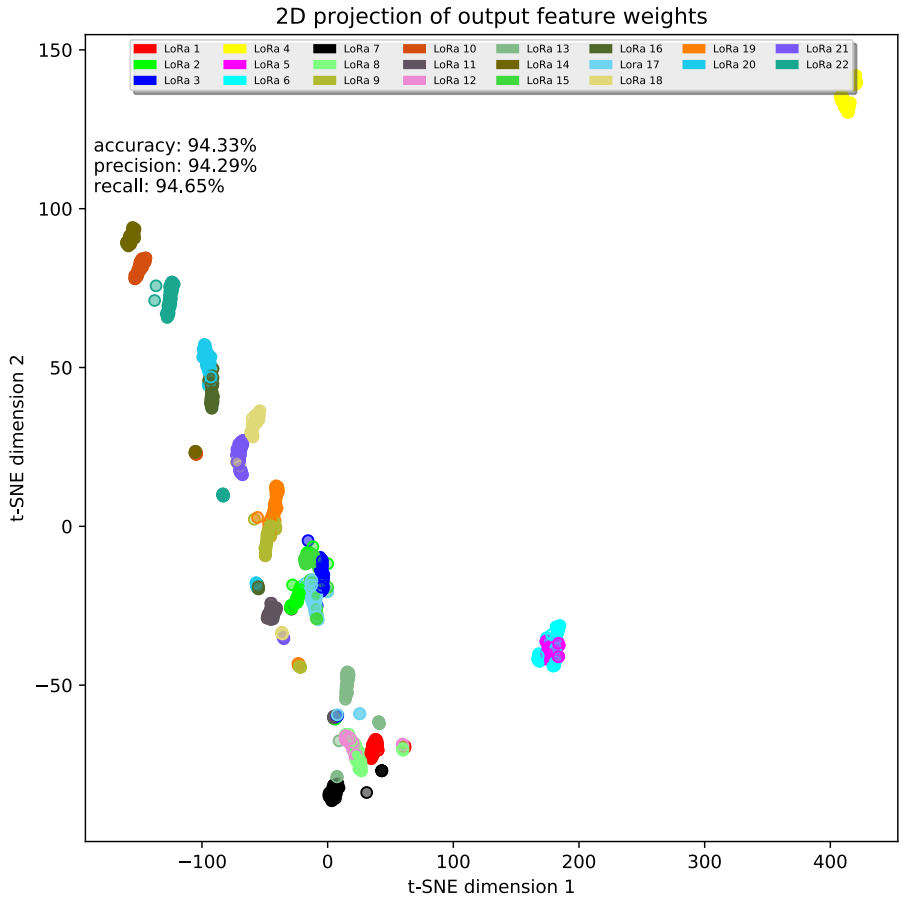


Figure 8.3.2: 2D t-SNE visualization of the output feature weights learned by the MLP model given dataset **III**. Each point represents a LoRa symbol, where the fill color indicates the true value and the outline color represents the predicted value.

Client-side security doesn't work.

Ryan Russell

9

Improving CEMA using Correlation Optimization

9.1 Introduction

Thus far in this thesis, we have only considered explicit information leakage in protocols and implicit information leakage from the PHY-layer radio transmissions of a wireless device. Another source of implicit information leakage originates from computation-dependent changes in the physical properties of a device's hardware that occur during the execution of a protocol implementation. Such leakage is called "side-channel leakage" (see Chapter 2). If present, measurements of physical properties can be statistically analyzed in order to infer information about the performed computation. This practice is referred to as SCA in the literature.

Previous SCA research has shown that secret information of a vulnerable cryptosystem can be exfiltrated through various types of side channels, including power consumption [137], temperature [39, 123], acoustic [16, 95, 238], and EM side channels [89, 93, 94, 97, 174, 175, 201, 204, 272]. Implicit information leakage through side channels is especially concerning in context of IoT devices and WSNs, since an adversary can more readily gain physical access to devices deployed in the field in order to perform side-channel measurements. The EM side channel is particularly interesting from the perspective of an adversary, for a number of reasons. First, EM waves can be captured without requiring physical contact with the hardware [174, 186]. This is in contrast to power consumption analysis, where the device under attack must typically be modified to obtain accurate measurements [261]. Second, EM leakage can originate from various components of a circuit due to coupling effects and circuit geometry [8, 170, 174] and is therefore hard to mitigate completely. Third, EM waves can potentially travel long distances, depending on the power and wavelength of the leakage signal. For example, Vuagnoux and Pasini demonstrated that EM emanations of PS/2 keyboards can be captured at a distance of up to 20 meters, even through walls [272]. Another example can be found in the work of Guri et al., where the invocation of specific memory-related instructions was used to transfer data via the resulting EM leakage, over a distance of up to 30 meters [114]. For these reasons, we will exclusively focus on the EM side channel in this chapter, although the concepts described could be applied to other side channels as well.

A known methodology to perform an EM-based side-channel attack on a device is Correlation Electromagnetic Analysis (CEMA) [73], which is based on the CPA attack for power side channels introduced by Brier et al. in [37]. In a CEMA attack, the adversary assumes that the power consumption of the hardware is related to the Hamming distance or Hamming weight leakage model.

Since consuming power produces electromagnetic radiation [137], the resulting EM emissions can be captured by an adversary and correlated with the leakage model in order to determine the secret information being processed.

Recall from Section 2.5.3 that in *template or profiling attacks* on the other hand, the adversary makes no preliminary assumption about the leakage model [186]. Instead, the adversary is assumed to be in possession of a "training" device identical to the device being attacked [49]. The attack is then performed in a two-phase process. In the training phase, the adversary estimates the probability distribution of the secret information in function of the measured EM leakage of the training device. Then, in the attack phase, the EM leakage of the device under attack is matched with the most probable template from the training phase [49, 201].

Recently, several works have indicated that profiling attacks can be interpreted as classification problems in the domains of ML and DL [41, 121, 152, 159, 195]. In this chapter, we introduce a novel profiling attack that, unlike these previous approaches, does not rely on classification for determining the secret key. More specifically, we do not directly *classify* individual EM traces into secret-key, intermediate, or Hamming values, but rather learn an *encoding*[1] of the EM traces that maximizes the Pearson correlation with the correct secret key. To this end, we introduce the "correlation loss function", which allows us to optimize the encodings for use in a CEMA attack using conventional ML optimization algorithms such as gradient descent. Furthermore, we show that our approach can be applied in the frequency domain, which removes the requirement of aligning the EM traces [93, 94, 174, 261]. We evaluate our methodology both on the ASCAD dataset [201], which features traces of masked AES operations, as well as a custom dataset, which is comprised of unprotected AES operations performed by an Arduino Duemilanove with an ATmega 328 CPU. We focus on information leakage during the execution of AES specifically, since AES is the default cipher used in WPA2, LoRa and many other wireless specifications. The EM traces of the custom dataset were recorded using a SDR and commodity EM probe, which shows that our approach can be used to perform a CEMA even with low-cost EM measurement hardware.

The structure of this chapter is as follows: in Section 9.2, we will first discuss a number of concepts that are necessary for understanding the remainder of the chapter. Section 9.3 then describes our "Correlation Optimization" technique, which is the main contribution of this work (**C9**). In this section, the technique

[1] Encodings are widely used for applications such as face verification and face recognition [232]. Some works in the machine learning literature refer to encodings as "embeddings".

will also be discussed and evaluated on both the ASCAD dataset, as well as the custom dataset. Related works are described in Section 9.4, followed by our conclusions in Section 9.5.

9.2 Background

9.2.1 Notation and terminology

In both the SCA and ML literature, the notational conventions and terminology vary depending on the authors of the work. Furthermore, some of the notations used in these domains overlap. As a concrete example, the output of the Hamming Weight (HW) power consumption model in the work of Brier et al. is denoted as W [37], which is also a common notation for the weight matrix in the ML domain. In the interest of avoiding ambiguities, we will now specify the notations and terminology used in this chapter.

We define a *dataset* as a collection of *traces*, where a trace is equivalent to a capture or measurement of the EM signal generated during one execution of an algorithm. Each trace consists of a number of *samples* of the instantaneous amplitude of the EM signal. A *training example* is a single trace that is used as the input to a ML model during its training phase. In this context, the samples of a trace (or a transformation thereof) are the *features* or *inputs* to the model. The *labels* or *classes* are then the outputs of the model. In this work, we will often refer to the outputs of the model for a given input trace as the *encodings* of that trace.

For ML-related concepts, we use the Stanford notation as detailed in Section 3.1. A recapitulation of all variables and notations used in this work can be found in Table 9.2.1.

9.2.2 Advanced Encryption Standard

The AES cipher is widely used for providing confidentiality and integrity in protocols such as Wi-Fi (802.11) [127], Bluetooth [30], TLS [80], and many others. The cipher has been extensively studied in previous works. For a detailed overview of the inner workings of AES, we refer the reader to the design document of AES [66]. This work focuses exclusively on side-channel attacks against

Table 9.2.1: Overview of the notations used in this chapter.

Symbol	Description	Sym.	Description
α	Scaling factor.	n_b	Number of bits.
$A^{[l]}$	Activations matrix of layer l.	n_l	Number of layers.
b	Bias tensor.	n_m	Number of training examples.
D	Current state.	n_x	Number of features.
d	Key byte guess score tensor.	ϕ	Encoding function.
ϵ	Value of 10^{-15}.	p	Plaintext tensor.
η	Noise tensor.	ρ	Pearson correlation coefficient.
g	Activation function.	R	Reference state.
H	Hypothesis matrix.	t	Time unit.
h	Tensor of model hypothesis values.	v	Intermediate value.
h_t	Power consumption model at time t.	$W^{[l]}$	Weight matrix of layer l.
$J(\hat{y}, y)$	Cost function.	X	Matrix of training examples.
k	True key tensor.	$x_j^{(i)}$	Feature j of training example i.
$\hat{k}$	Key guess tensor.	$\hat{Y}$	Matrix of model output values.
k_s	Byte s of k.	Y	Matrix of true values.
$\mathcal{L}(\hat{y}, y)$	Loss function.	$\hat{y}$	Model output values (encodings).
l	Layer index.	y	True model output.

the first round of AES. In this round, the `RoundKey` is initialized to the secret key $k = \{k_1, k_2, \ldots, k_{16}\}$ [66]. We will henceforth refer to the elements of k as "key bytes". Given a plaintext $p = \{p_1, p_2, \ldots p_{16}\}$, the intermediate value v after the `SubBytes` operation is equal to:

$$v = \text{SBox}(p \oplus k) \tag{9.1}$$

Because the secret key k is processed directly during the first round of the cipher, performing a side-channel attack is trivial. Several countermeasures have been developed to increase the difficulty of successfully performing an attack. A first countermeasure is called "masking", which attempts to change the power consumption characteristics of a device by randomizing the processed intermediate values [174]. An example of such an approach is the "table recomputation" method [200].

A second commonly implemented countermeasure is "hiding", which, as the name implies attempts to hide the hardware power consumption characteristics from an adversary. This can be done in a number of ways, for example by introducing jitter to the clock of the device, introducing dummy instructions, adding random interrupts, etc. [174].

9.2.3 Correlation Electromagnetic Analysis of AES

The classic CPA technique, as first described in the work of Brier et al. [37], utilizes the Hamming Distance (HD) to model the data leakage through a power side channel. More specifically, the number of bits switching from an n_b-bit reference state R to another state D at a given time t is assumed to be linearly related to the power consumption of the hardware [18, 174, 260, 261]. Formally, we define this power consumption model h as follows: [37]

$$h_t = \alpha HW(D_t \oplus R_t) + \eta_t \tag{9.2}$$

In the above equation, HW is the Hamming weight function, α is a scaling factor and η_t is a noise term at time t. Generally, α and η are unknown, and the adversary's goal is to recover D_t and R_t from h_t. If the underlying hardware implements pre-charged logic or if certain transitions are prohibited due to separated busses for data and addresses, the reference state R_t can be systematically equal to 0 [37, 181]. In this case, the HD model generalises to the HW model.

When applied to AES, a CEMA using the HW or HD model often targets the output of the `AddRoundKey` and `SubBytes` operations in the first round of the cipher, as the secret key is processed directly in this round [93, 159, 165, 174]. If we plug the intermediate value from Equation 9.1 into the HW leakage model, the EM leakage in function of the key and plaintext becomes:

$$h_t = \alpha HW(\mathrm{SBox}(p \oplus k)) + \eta_t \tag{9.3}$$

The adversary can now determine the value of h_t by supplying a known plaintext to the device under attack and analyzing traces of EM emanations captured during the first AES round. Recall that we define a trace as an AM-demodulated signal (i.e. a time series of signal magnitudes) captured using an oscilloscope or signal analyzer, that characterizes the power consumption of the device over time.

In order to determine the unknown k given h_t, the adversary can construct a hypothesis power consumption matrix $H_{ij} = \mathrm{SBox}(p_i \oplus j)$ for each key byte index $i \in \{1, 2, \ldots, 16\}$ and each possible key value $j \in \{0, 1, \ldots, 255\}$. This matrix essentially gives all power consumption values h_t for each possible key, assuming $\alpha = 1$ and $\eta = 0$. The adversary then calculates the Pearson correlation coefficients between the observed values for h_t from the captured EM traces and the hypothesized values from H for each key guess [37, 175]. This process is

repeated for each relevant time unit t, resulting in a three-dimensional correlation matrix:

$$\rho_{tij}(h_t, H_{ij}) = \frac{cov(h_t, H_{ij})}{\sigma_{h_t}\sigma_{H_{ij}}} \tag{9.4}$$

Finally, the best key guess $\hat{k}$ is determined by selecting the key value with the highest absolute value[2] of the correlation:

$$\hat{k}_i = \arg\max_{t,j}(|\rho_{tij}|) \tag{9.5}$$

9.2.4 Machine Learning and Deep Learning attacks on AES

In ML and DL side-channel attacks, the objective of finding the secret key k given a collection of traces is formulated as a *supervised classification problem*. Analogous to the classic TAs, solving this problem involves two phases: a training phase and testing phase.

In the training phase, a "training set" of EM traces is first collected from the reference device. Here, each trace or "training example" is labeled with a corresponding "class label". The class label is what the algorithm will be trained to predict and can be chosen in several ways. One possibility is to consider each possible key byte value as a separate class, which yields 256 class labels. Another possibility could be to consider the HW of each key byte value, resulting in only 9 class labels (all possible Hamming weight values of a single byte). The ML algorithm is subsequently trained, which means that a model is parameterised such that a certain predetermined loss function (e.g. the cross-entropy loss) is minimized.

After training is complete, a "test set" of EM traces is collected from the targeted device during the testing phase. Since the key bytes are unknown in this case, only the EM traces are available. The ML algorithm will output the probability of each EM trace belonging to a certain class, based on the parameters learned by the model during the training phase. The accuracy of this prediction largely depends on the quality of the training data, but also on the chosen type of ML

[2]The absolute value of the Pearson correlation is considered, since it does not matter whether the correlation between the leakage and a certain key byte is positive or negative.

model, hyperparameters, optimizer, and loss function. In this work, we will primarily focus on simple ML models such as MLPs. We will show that even such simple models outperform the state-of-the-art models from previous works when using our approach, which will be detailed in Section 9.3.

9.2.5 The ASCAD dataset

The ASCAD dataset was introduced by Prouff et al. in [201] for the purpose of providing a benchmark to evaluate ML and DL techniques in context of side-channel attacks. The dataset consists of three separate HDF5 files: `ASCAD.h5`, `ASCAD_desync50.h5`, and `ASCAD_desync100.h5`. Each file contains 60,000 EM traces (50,000 training / cross-validation traces and 10,000 test traces) captured with a sensor attached to an oscilloscope sampling at 2 GS/s. The traces contain 700 samples of the EM radiation emitted by an ATMega8515 device during the execution of the first round of a software AES implementation. The AES implementation is secured against first-order side-channel attacks with the masking technique (see Section 9.2.2). Traces in the `ASCAD.h5` file have been time-aligned in a preprocessing step, whereas the traces in `ASCAD_desync50.h5` and `ASCAD_desync100.h5` have been shifted with a maximum jitter window of respectively 50 and 100 samples [201]. Besides EM measurements, the ASCAD authors have provided the source code of the models used in their work. In Section 9.3.6, we will compare these models to our Correlation Optimization (CO) technique.

9.3 Correlation Optimization

When performing a CEMA attack as described in Section 9.2.3, recall that the adversary constructs the matrix ρ_{tij} which gives the correlation coefficients of all key byte hypotheses for each time unit in the trace. Then, the hypothesis with the highest correlation to the power consumption traces is selected as the most likely true key byte. Note that this correlation is determined by only *one time unit* for each trace, i.e. the samples at other time units are discarded. However, some of these discarded samples are correlated to the correct hypothesis, albeit to a lesser degree.

This observation raises the question of how information from multiple samples can be combined in order to improve the efficiency of CEMA. Since the leakage itself depends on the input plaintext, key, and complex electromagnetic interactions governed by Maxwell's equations in the underlying hardware, the samples that

contain useful information may not occur at the same time units over multiple traces. Measurements are also noisy (due to interference and fading effects) and highly dimensional (in terms of samples per trace), which complicates the issue further. Although classic TAs consider multiple time units as well [49], they are not suited for high-dimensional data [41, 152].

We now introduce our approach, called "Correlation Optimization", which aims to address the aforementioned issues by exploiting the information leakage from multiple samples. To this end, we consider the selection of "good" samples as a ML optimization problem. That is, given an input of trace samples $x^{(i)} = \{x_1^{(i)} \dots x_{n_x}^{(i)}\} \in X$ with $i \in \{1, 2, \dots n_m\}$, we would like to determine which *encoding* of samples $\hat{y}^{(i)} \in \hat{Y}$ can be obtained such that $\rho(\hat{Y}, Y)$ is maximal. Here, the term encoding refers to an arbitrary function ϕ of the input features, i.e.:

$$\hat{y}^{(i)} = \phi(x^{(i)}) \tag{9.6}$$

Observe that the above optimization problem aligns well with the problems from domains such as face recognition and face verification, where the ML model *matches encodings* of faces rather than predicting a class probability for each face[3] [232]. Since in these domains ML models achieve state-of-the-art performance, we will apply similar techniques in our approach.

9.3.1 The correlation loss function

Recall that we would like to find an encoding of input samples $\hat{y} = \{\hat{y}^{(1)}, \dots, \hat{y}^{(n_m)}\}$ such that $\rho(\hat{Y}, Y)$ is maximal. To this end, a loss function must be defined that can be evaluated by an optimizer algorithm in order to train the parameters of our ML model. Ideally, this loss function should be large when there is no linear correlation between $\hat{Y}$ and Y, and zero when the correlation is maximal. For a tensor of key bytes $y_k = \{y_k^{(1)}, \dots, y_k^{(n_m)}\}$ at index $k \in \{1, 2, \dots n_k\}$ and the corresponding tensor of encodings $\hat{y}_k = \{\hat{y}_k^{(1)}, \dots, \hat{y}_k^{(n_m)}\}$, we define the loss function as:

[3] To understand why, note that when given a database of millions of faces, it would be infeasible to train a model with a one-hot encoded class label assigned to each of the faces.

$$\mathcal{L}(\hat{y}_k, y_k) = 1 - \frac{cov(\hat{y}_k, y_k)}{\sigma_{\hat{y}_k}\sigma_{y_k} + \epsilon} \tag{9.7}$$

$$= \frac{\sum_{i=1}^{n_m}[(\hat{y}_k^{(i)} - \overline{\hat{y}}_k)(y_k^{(i)} - \overline{y}_k)]}{\sqrt{\sum_{i=1}^{n_m}(\hat{y}_k^{(i)} - \overline{\hat{y}}_k)^2}\sqrt{\sum_{i=1}^{n_m}(y_k^{(i)} - \overline{y}_k)^2} + \epsilon} \tag{9.8}$$

Here, ϵ is a small value (e.g. 10^{-15}) introduced to prevent division by zero. If we assume that $\hat{y}_k$ and y_k are mean-normalized, Equation 9.8 can be converted to a more convenient vector form:

$$\mathcal{L}(\hat{y}_k, y_k) = 1 - \frac{\hat{y}_k \cdot y_k}{\|\hat{y}_k\| \cdot \|y_k\| + \epsilon} \tag{9.9}$$

In case we want to simultaneously train the ML model for all values of k, a cost function[4] can be defined as follows:

$$J(\hat{y}, y) = \sum_{k=1}^{n_k} \mathcal{L}(\hat{y}_k, y_k) \tag{9.10}$$

Observe that the maximum value of the cost function J is 32 in the worst case (when all correlations are -1), and 0 in the best case (when all correlations are 1).

9.3.2 Evaluation methodology

For the evaluation of our CO technique, we will use the same methodology as Prouff et al., so that an objective comparison can be made. As such, our ML models will be trained to attack the third key byte of the masked AES implementation from the ASCAD database (see Section 9.2.5 for a description of this dataset). The implementation of these models was written in Python, using the library "Keras" [53] with a Tensorflow backend [3]. All source code, scripts, and data used to generate the figures is available on Github at `https:`

[4] In some ML papers, a loss function defines the cost for a single training example, whereas the cost function defines the total cost. In this work, we define the cost function as the total cost for all bytes of the AES key instead of all training examples.

//github.com/rpp0/correlation-optimization-paper.

t-fold cross-validation

Unless specified otherwise, each ML model was evaluated with a 10-fold cross validation, using a train and test split of respectively 45,000 and 5,000 traces. Hence, we first train a model from scratch with 45,000 random traces, and evaluate the performance of this newly trained model on 5,000 different (unseen) traces. This process is repeated 10 times, and finally the performance metrics are averaged to obtain a conclusion.

Performance metrics

In each of our experiments, we use two metrics to assess the performance of our models: "rank" and "confidence". Here, we define the "rank" as the index of the correct key in a tensor d that contains the scores assigned to a key byte k_s by the model, sorted such that $d_i \geq d_j \; \forall \; i, j \in \{1, 2, \ldots, 256\} \mid i < j$. For example, the score tensor can contain key byte probabilities or correlations. More formally, we define the rank as:

$$\text{rank}(d) = \{i - 1 | d_i = \text{score}(k_s)\} \tag{9.11}$$

Note that the lowest possible rank is 0. If we employ an optimal guessing strategy by iteratively guessing the key byte with the next highest score in d, the rank plus one corresponds to the number of guesses required to find the correct key. The *expected* number of key guesses is called the GE, and is commonly used to evaluate SCA methods [140, 181, 213, 253]:

$$GE = \sum_{i=1}^{K} iP(\text{rank}(d) = i - 1) \tag{9.12}$$

We obtain the expected value of the number of key guesses by averaging the rank over the 10 folds of the ML model, which we will denote as the "mean rank" in the coming figures. The "confidence" is then defined as the score difference between rank 0 and rank 1 or equivalently, as the distance between the maximum

score in d and the next highest score in d [175]:

$$\text{confidence}(d) = d_1 - d_2 \tag{9.13}$$

Analogous to the rank, we average the confidence over all 10 folds to obtain the "mean confidence". Although this metric is less frequently used in related works, we believe it is useful because it gives an insight into how well the model can distinguish a singular key byte guess.

Input and label preprocessing

The inputs and labels to the ML models are preprocessed during the training phase. First, all traces are split into mini-batches of 512 traces before being fed to the network. This removes the requirement of having to load the entire dataset in memory, at the cost of decreased performance since multiple iterations are now needed to process the entire training set. Note that if the mini-batch size is too small, its correlation loss might not be representative for the entire training set, and it may fail to converge as a result. We empirically determined 512 traces to be a good value for the mini-batch size for the ASCAD dataset.

Second, the true labels y are preprocessed such that $y_s^{(i)} = HW(SBox(p_s^{(i)} \oplus k_s^{(i)}))$, where s is the key byte index and i is the training example index. It is important to note that we *do not* supply the variable AES masking values during the training phase: an encoding that correlates the EM radiation directly with the corresponding key byte will automatically be learned by the model. Further, as we will discuss in Section 9.3.4, assuming the Hamming Weight power consumption model is not necessary, but will reduce the required training time and complexity of the model architecture. Finally, the training example input features $x^{(i)}$ are simply equivalent to the raw samples of trace i, unless specified otherwise.

9.3.3 Time-domain CO

Using the correlation loss function defined in Section 9.3.1, an encoding function ϕ that maximizes Equation 9.4 can be learned by the ML model. In this section, we will discuss the performance of the encoding function if we use time domain samples as its input features. To this end, we first train the ML model for 100 epochs on the three ASCAD datasets. Then, we generate the encodings for each

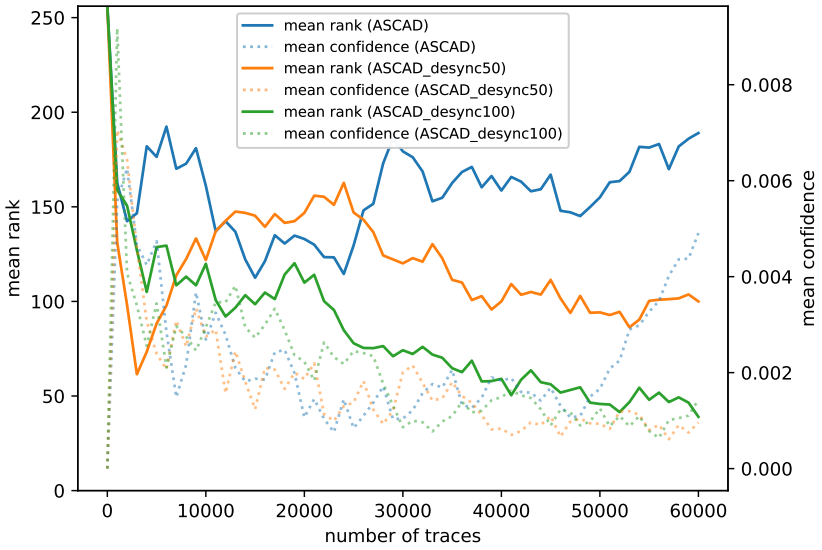


Figure 9.3.1: Mean rank based on a 10-fold cross-validation of the time-domain CO identity encoding function in relation to the number of validation traces used for a CEMA attack. The CEMA attack is unsuccessful even if all 60,000 traces are utilized, and the confidence in the guessed key bytes is low, ranging from 0.001 to 0.005.

of the traces using the trained model, and perform a CEMA attack on these encodings.

Identity function

In order to establish a performance base line, we first consider the case where ϕ is the identity function, i.e. $\hat{y} = \{x_1, x_2, \ldots, x_m\}$. This is equivalent to performing a regular CEMA attack on the EM traces as described in Section 9.2.3. Since there are no weights or other parameters to learn, we use all 60,000 available traces for the evaluation of the model. The results of this evaluation are shown in Figure 9.3.1.

As expected, the standard CEMA attack is not successful in guessing the true key due to the masking countermeasure that was implemented (see Section 9.2.2). Although the mean rank does seem to decrease slightly for `ASCAD_desync50` and

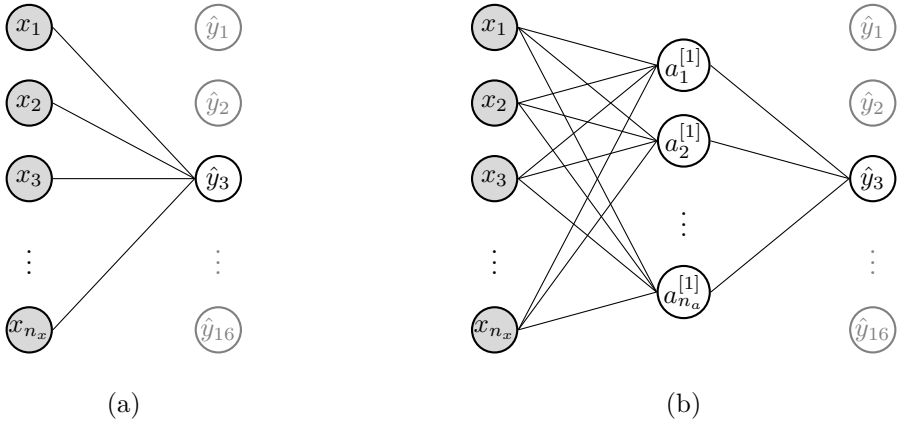


Figure 9.3.2: The single-layer (a) and two-layer (b) MLP architectures used in our work for CO. Only connections to the output for key byte 3 (y_3) are shown for clarity.

`ASCAD_desync100` when more traces are considered, note that the confidence is low for each dataset. This indicates there is no clear difference between the best key guess and the second best key guess, and that more traces would therefore be needed to obtain a reliable result.

Single-layer MLP

When using a single-layer MLP architecture for CO, we essentially let the learning algorithm determine a linear combination of time-domain samples that, when passed through a nonlinear activation function, results in a maximal correlation with the correct key for the entire training set. The output of the MLP is:

$$\hat{Y} = g(XW + b) \tag{9.14}$$

The architecture is visualized in Figure 9.3.2 (a). In practice, we add a batch normalization layer before the activation function to speed up the learning algorithm [129]. As the activation function, we chose the leaky ReLU activation in order to mitigate the occurrence of zero-gradients [158].

Note that the CEMA attack will be performed on a *single* output encoding sam-

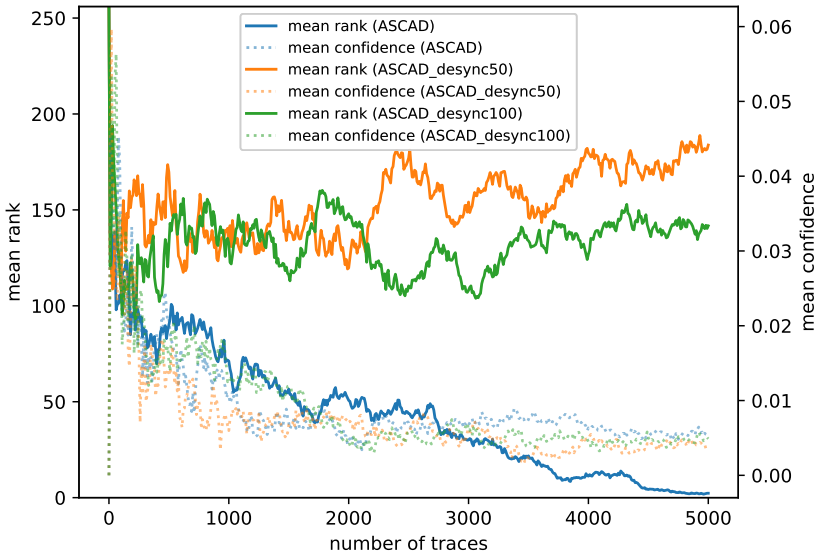


Figure 9.3.3: Mean rank based on a 10-fold cross-validation of the time-domain single-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. Compared to the identity function, an improvement can be observed for the `ASCAD` dataset: a minimum mean rank of 1 is obtained after 4,960 traces. However, the mean confidence in the guessed key bytes is still low, ranging from 0.003 to 0.005. For the desynchronized datasets, no improvement is observed.

ple, determined by the parameters learned by ML model. The mean rank and the confidence after training on 45,000 traces for 100 epochs and evaluating on a validation set of 5,000 different traces is shown in Figure 9.3.3.

Clearly, the model has learned an improvement over the identity function: a mean rank of 1 is achieved after 4,960 traces even though we used only 5,000 traces for the CEMA attack instead of 60,000. It should be noted that a mean rank of 0 *can be* achieved if more traces were to be added to the validation set, though we only show the first 5,000 traces for a fair comparison with the other experiments and related works.

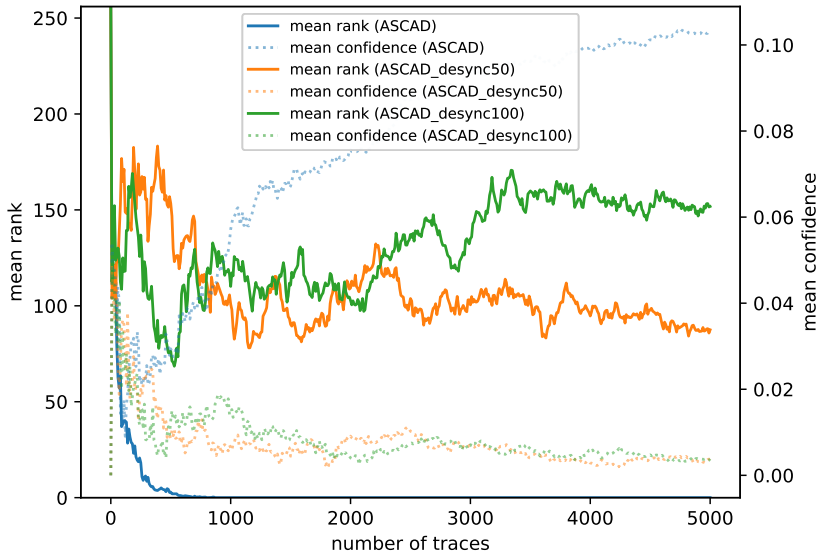


Figure 9.3.4: Mean rank based on a 10-fold cross-validation of the time-domain two-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. The CEMA attack successfully finds the correct key for the `ASCAD` dataset after 800 traces with a mean confidence of 0.038, which rises further to 0.10 as more samples are included in the validation set. However, the attack fails on the desynchronized datasets `ASCAD_desync50` and `ASCAD_desync100`.

Two-layer MLP

Although single-layer MLPs are intuitive in the sense that they essentially learn a single weight for each sample in a trace, they do not allow for learning complex encoding functions. Ideally, we would like to learn dependencies between samples as well. To achieve this, a more complex architecture can be used, such as an MLP with a hidden layer as shown in Figure 9.3.2 (b). The output of the model then becomes:

$$\hat{Y} = g(g(XW^{[0]} + b^{[0]})W^{[1]} + b^{[1]}) \tag{9.15}$$

Again, we add batch normalization layers to speed up training and perform a CEMA attack on the encodings of 5,000 traces after training on 45,000 traces.

The result is shown in Figure 9.3.4. With this model, we obtain a very encouraging result for the `ASCAD` dataset: the model has learned to defeat the masked implementation of AES and requires only around 1,000 traces to achieve a mean rank of 0. Furthermore, the confidence after 5,000 traces is an order of magnitude higher compared to the single-layer MLP.

Unfortunately, for the `ASCAD_desync50` and `ASCAD_desync100` dataset, the model is not able to learn a meaningful encoding. This is to be expected, as MLPs are very sensitive to translations of the input features [150]. Introducing deliberate clock jitter to the hardware of a device running AES would therefore be an effective countermeasure against time-domain CO.

9.3.4 Frequency-domain CO

As evidenced in Section 9.3.3, shifting traces in time to cause misalignment can be an effective mitigation against CEMA attacks. Cagli et al. discuss several methods to mitigate the issue of misalignment in their work: increasing the number of side-channel acquisitions, applying realignment techniques, and using CNNs [41]. However, another possibility is to consider traces in the frequency domain [93, 94, 174], as first proposed for Differential Electromagnetic Analysis (DEMA) attacks by Tiu in [261]. Inspired by this approach, we studied the effectiveness of CO in the frequency domain. To this end, we first preprocess each of the traces by applying a 700-point FFT and taking the magnitude of the result, discarding the phase information.

Identity function

Similarly to our approach for the time-domain CO, we first consider the case where ϕ is the identity function in order to establish a baseline for frequency-domain CO. Thus, we have $\hat{y} = abs(FFT(\{x_1, x_2, \ldots, x_m\}))$ as the output of the model for a single training example.

The result of performing a CEMA attack on all 60,000 traces is shown in Figure 9.3.5. Here, the mean rank is better compared to the time-domain CO identity function experiment: the mean rank for `ASCAD_desync100` reaches zero at 51,000 traces. This indicates that there is a single frequency component that, when analysed, allows us to defeat the masked AES implementation without CO. However, the confidence for each dataset is still low, and many traces are required for the

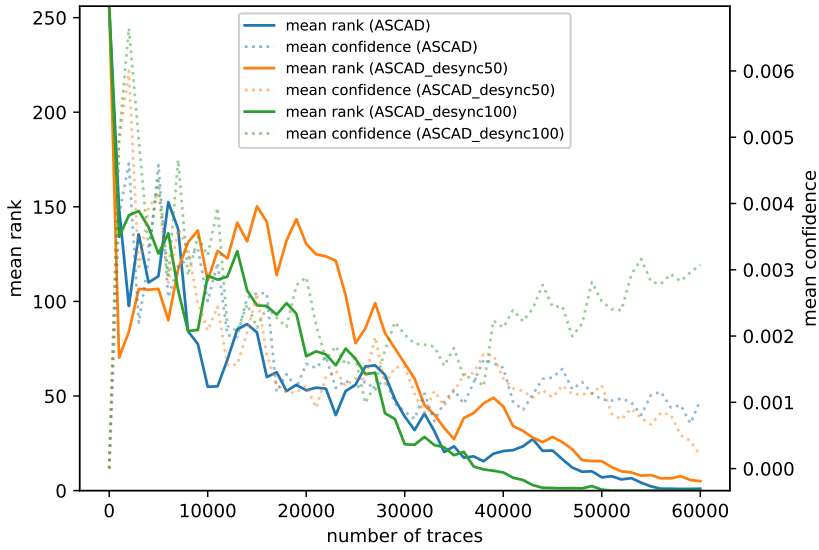


Figure 9.3.5: Mean rank based on a 10-fold cross-validation of the frequency-domain CO identity encoding function in relation to the number of validation traces used for a CEMA attack. The CEMA attack successfully finds the correct key for the `ASCAD_desync100` dataset after 51,000 traces, but the mean confidence is low (0.003). The attack is unsuccessful for the other datasets.

CEMA attack to succeed.

Single-layer MLP

Next, we consider linear combinations of the FFT frequency bins passed through an activation function. We use the same encoding function ϕ as for the time-domain single-layer MLP model from Section 9.3.3. Figure 9.3.6 shows the mean rank and the confidence of a CEMA attack on the encodings of 5,000 traces, after training on 45,000 traces for 100 epochs.

The improvement over the identity function is similar to what we observed for time-domain CO: only 4,840 traces are required to obtain a mean rank of 0, with a mean confidence of 0.008.

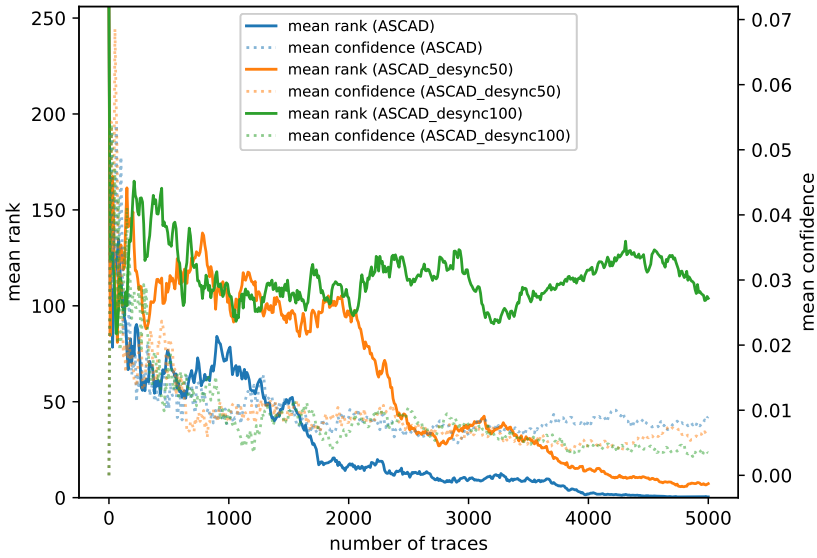


Figure 9.3.6: Mean rank based on a 10-fold cross-validation of the frequency-domain single-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. Again, an improvement compared to the identity function can be observed: a mean rank of 0 is obtained for the `ASCAD` dataset after only 4,840 traces, with a mean confidence of 0.008. The attack is unsuccessful on the other datasets.

Two-layer MLP

Using an MLP with a hidden layer allows the model to learn more complex relationships between frequency bins of the FFT. Again, we use the same encoding function ϕ as in Section 9.3.3. The result after evaluation of the model is shown in Figure 9.3.7. Observe that the model is now successfully able to learn a meaningful encoding function for *all of the ASCAD datasets*. After approximately 1,000 traces, a mean rank of 0 is achieved by each model. Furthermore, if we increase the number of traces, the mean confidence in the key guess increases as well. We believe this is a very encouraging result, showing that CO can be used to defeat both the masked AES and clock jitter countermeasures.

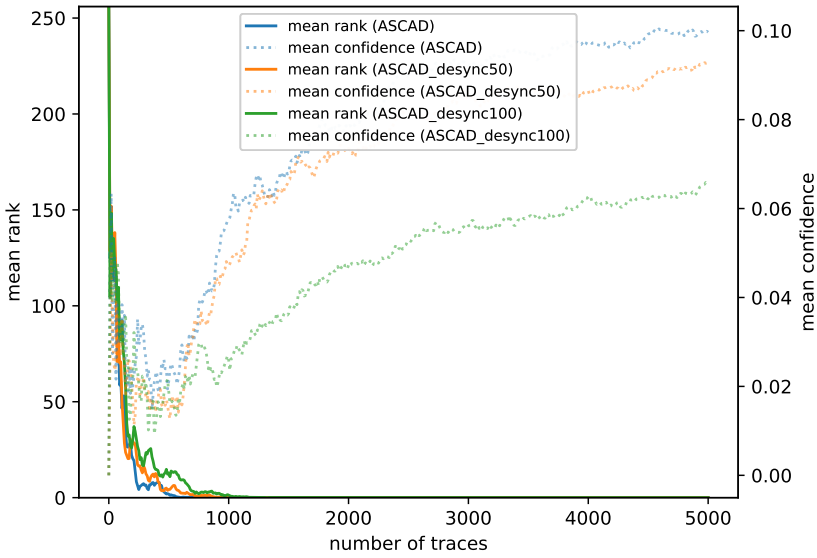


Figure 9.3.7: Mean rank based on a 10-fold cross-validation of the frequency-domain two-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. The correct key is guessed for each dataset after approximately 1,000 traces. Adding more traces to the validation set further increases the mean confidence to 0.099, 0.092, and 0.066 for respectively the `ASCAD`, `ASCAD_desync50` and `ASCAD_desync100` datasets.

No model assumption

As indicated in Section 9.3.2, the true labels $y^{(1)}, y^{(2)}, \ldots, y^{(n_x)}$ are preprocessed such that their key byte values correspond to $\text{HW}(\text{SBox}(p_s^{(i)} \oplus k_s^{(i)}))$. Hence, we assume that the cryptographic device leaks information based on the HW power consumption model. In the following experiment, we determine whether the power consumption model can be learned implicitly by the optimization algorithm. To this end, we label $y_s^{(i)} = \text{SBox}(p_s^{(i)} \oplus k_s^{(i)})$ so that the label values correspond to the intermediate value after the processing of the SBox. Then, we perform CO with the frequency-domain two-layer MLP model on the `ASCAD_desync100` dataset. The results are shown in Figure 9.3.8.

Compared to when a HW model was assumed (see Section 9.3.4), the correct key is guessed around 1,200 traces instead of 1,000 traces. Furthermore, the confidence is slightly lower, with 0.045 compared to 0.066. We conjecture that

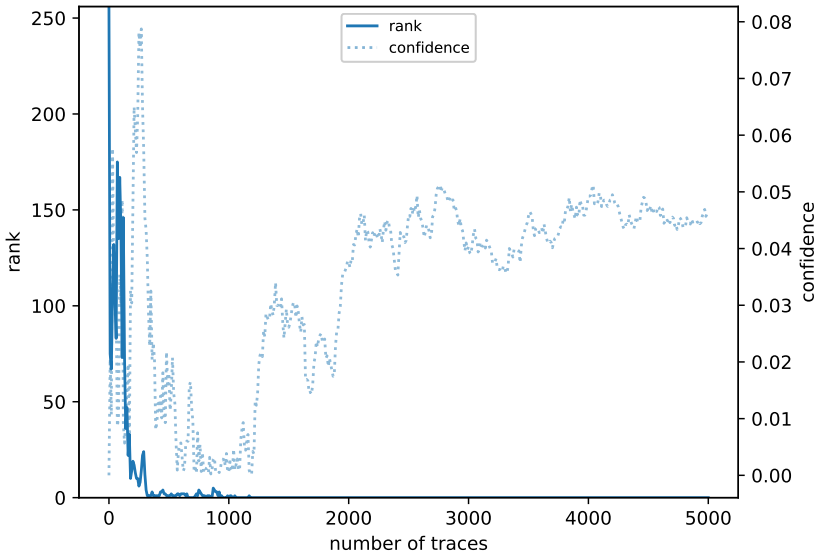


Figure 9.3.8: Rank and confidence of the frequency-domain two-layer MLP CO model in relation to the number of validation traces used for a CEMA attack. The power consumption model is not assumed; it is learned by the model. The correct key for the `ASCAD_desync100` dataset is guessed after approximately 1,200 traces with a confidence of 0.001, and increases further to 0.045 after 5,000 traces.

this slight decrease in performance is caused by the added complexity of learning the HW function.

9.3.5 Low-cost CEMA

The ASCAD dataset was recorded using an expensive oscilloscope and a sample rate of 2 GS/s. Since the frequency-domain CO from Section 9.3.4 showed promising results for noisy and unaligned data, we also investigated the effectiveness of the technique when using lower-cost hardware such as an SDR. To this end, we recorded a custom dataset of EM traces transmitted by an Arduino Duemilanove running a software AES implementation, using a Universal Software Radio Peripheral (USRP) B200 SDR sampling at 8 MS/s on the 64 MHz band with a TBPS01 EM probe and wideband amplifier. The experimental setup

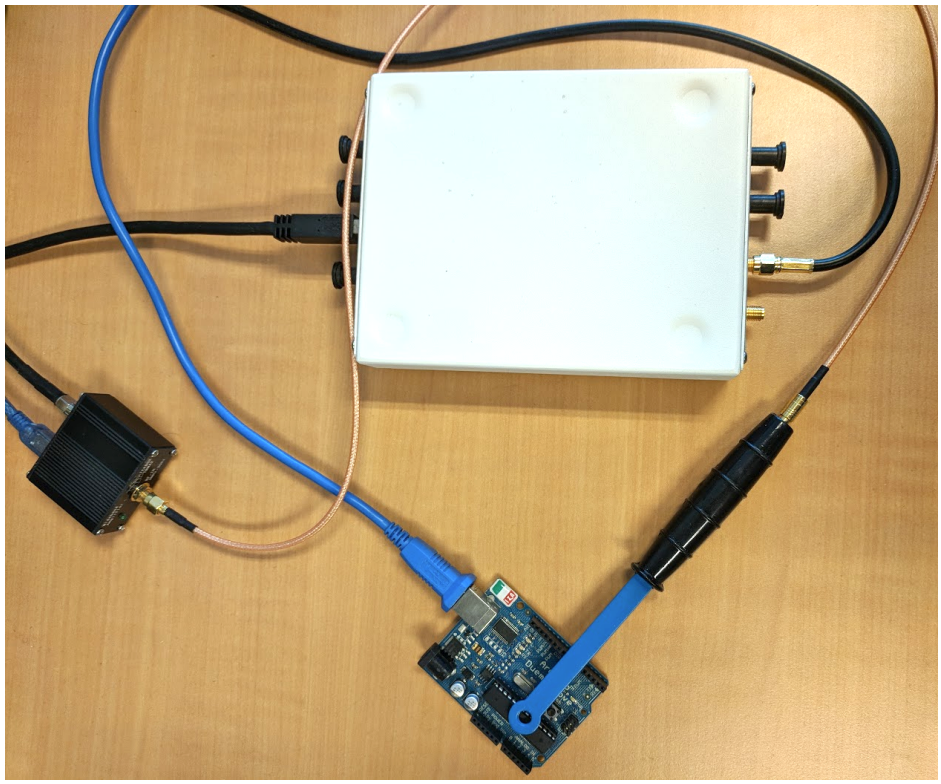


Figure 9.3.9: Experimental test setup for the low-cost CEMA attack. A Tekbox TBPS01 EM probe is positioned near the VCC pin of the ATmega 328, and is connected to the USRP B200 SDR through a 40 dB wideband amplifier.

is shown in Figure 9.3.9.

Dataset properties

The custom dataset consists of two sets of traces: a training set of 51,200 traces and validation set of 32,768 traces. Each trace contains the instantaneous amplitude of the I/Q signal provided by the SDR. No further preprocessing was performed. While recording the training set, the Arduino continuously performed AES encryption with a *random key*, such that one trace contains the EM leakage of one random encryption operation. The validation set contains EM traces of

AES encryptions performed with a *fixed key*. This ensures that the ML model will not overfit on one specific key during training, and that a sufficient number of traces is available to perform a CEMA attack during validation. We stress that the fixed key used during the validation step is never encountered during training.

CO results

At first, we directly used the MLP architecture from Section 9.3.4 to train the model on the 51,200 random-key training examples. This approach turned out to be unsuccessful: the model heavily overfits on the noise that is present in the training examples after each subsequent epoch, and does not learn a generalizable relation between the key byte value and EM leakage. As a result, the training set loss is very low, whereas the validation set loss is high.

In order to resolve this issue, we artificially generated more training data by applying the *data augmentation* technique. More specifically, for each of the training examples, we set the starting offset of the sample window for which the FFT is calculated to a random offset between 0 and 500 time units. Indeed, with this approach, a training example will be time-shifted slightly differently for each epoch, which reduces the overfitting of the MLP model. The result after training for 100 epochs on the augmented training set is shown in Figure 9.3.10.

Observe that after 22,000 traces, the CEMA attack successfully determines the correct key, without performing any alignment or filtering of the EM traces and with a SDR sample rate of only 8 MS/s.

9.3.6 Discussion

Comparison to previous approaches

Previous works in context of applying ML to SCA have in common that they all use a model optimized by minimizing the mean cross-entropy loss over the training set. Thus, for each training example, the probability distribution $P(\hat{y}^{(i)} = v \mid x^{(i)})$ is calculated for each intermediate value $v \in \{0, 1, \ldots 255\}$ and its cross-entropy with the one-hot encoding of the true intermediate value $y^{(i)}$ is determined. Intuitively, this can be regarded as a SEMA attack on a single trace, since no information from other traces is used.

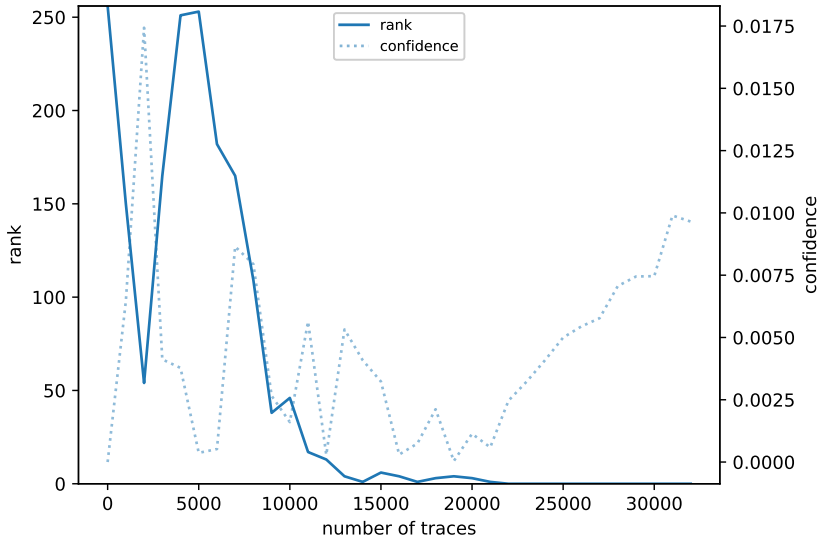


Figure 9.3.10: Rank and confidence for a single model trained on the custom dataset, after performing a CEMA attack on its encodings. The correct key is found after 22,000 traces.

Although the above approach has been demonstrated to achieve reasonable results in context of SCA, we believe it is more suited to image classification applications, where only one image is often available that needs to be classified. In SCA, there is the need for extracting information from *multiple* examples, due to the noisy nature of EM traces. CO achieves this by optimizing the correlation coefficient of a mini-batch with the true key byte value. This may explain why even a simple two-layer MLP architecture performs better than the `best_cnn` model from the ASCAD paper; their CNN is unable to determine the correct key for the `ASCAD_desync100` dataset (see [201, p. 39]). We confirmed this result by retraining their `best_cnn` model for 100 epochs and performing a 10-fold cross-validation with the ASCAD datasets, analogous to the experiments previously discussed in Sections 9.3.3 and 9.3.4. The results of this experiment are shown in Figure 9.3.11.

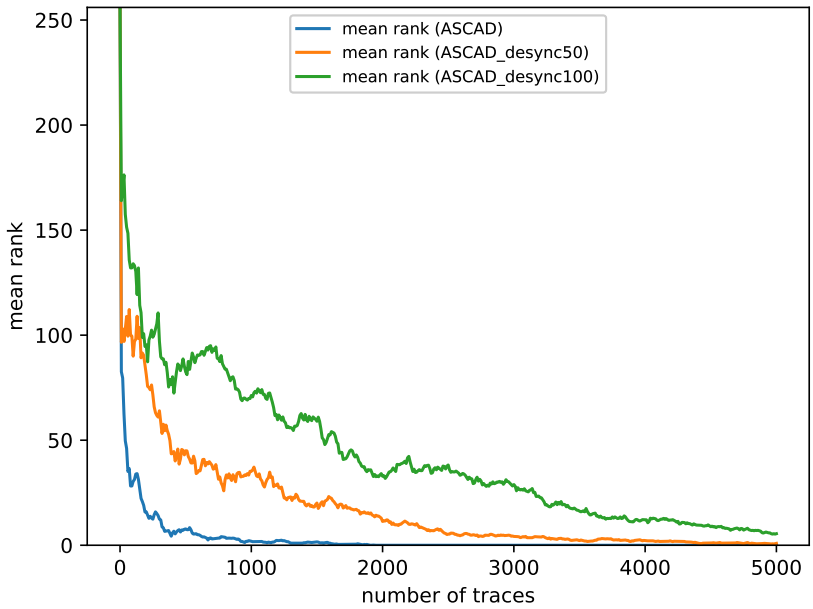


Figure 9.3.11: Mean rank based on a 10-fold cross-validation of the `best_cnn` model from the work of Prouff et al. [201], in relation to the number of validation traces used for a classification attack. Their model is able to determine the correct key for the `ASCAD` dataset after 1,910 traces, but performs poorly on the `ASCAD_desync50` and `ASCAD_desync100` datasets compared to our best CO models.

Complexity

The most complex model we considered is the two-layer MLP model from Section 9.3.4. This model contains 180,741 parameters and took 271 seconds to train for 100 epochs on a Dell Latitude laptop with quad-core Intel Core i5-7300U CPU at 2.60 GHz (no GPU). By comparison, the ASCAD `best_cnn` model contains 66,652,544 parameters. In addition, the presence of convolutional layers further increases the complexity of this model. As a result, training the ASCAD CNN models for 100 epochs with this architecture takes 5.21 days per model on the same machine.

Hybrid CO and round detection

In our custom dataset containing the Arduino EM traces, each trace contains a variable number of samples, and the only assertion is that precisely one AES encryption is performed within the trace. In Section 9.3.5, we demonstrated that it is not required to align the traces in a preprocessing step due to the frequency-domain approach. The only requirement is that the first round of the AES encryption is contained within the FFT window. However, the larger the FFT window, the more irrelevant information will be included in the power spectrum and hence, the more traces are required to successfully perform a CEMA attack. This raises the question of how the FFT window can be optimally selected.

We did not consider the optimal selection of FFT windows in this work, but leave a number of interesting pointers for future work. A first possible approach is to let the neural network determine a boundary of where the first AES round starts. This could be achieved with an algorithm such as YOLO [209], which automatically determines bounding boxes of certain objects (in this case an AES round). A second approach could be to perform a "hybrid" CO that takes place both in the time domain as well as the frequency domain. For example, by calculating a spectogram with overlapping FFT windows of the entire trace and finding the spectogram section with the highest correlation or by using wavelet transforms.

9.4 Related work

The first study of using ML in SCA was conducted by Hospodar et al. in 2011, where a power analysis of a software AES implementation without countermeasures was performed using a SVM [121]. In 2015, Lerman et al. compared ML techniques with the classic TA and show that ML are especially interesting when the number of useless samples in a leakage trace increases and/or when the training set size is small. Maghrebi et al. apply DL techniques in side-channel context, and show that DL-based attacks are more efficient than ML-based and template attacks [159]. This observation was recently nuanced by Picek et al. who suggest that ML techniques can perform on a similar level or even outperform DL techniques depending on the level of noise, number of measurements and number of features [195]. Cagli et al. studied the robustness to misalignment of CNNs for an unmasked AES implementation [41]. A comprehensive study regarding the application of DL algorithms in context of EM side-channel attacks was conducted by Prouff et al. in [201]. They also introduced the ASCAD benchmark database,

which was used in extensively Section 9.3 to evaluate our approach.

The use of frequency-domain features in SCA was pioneered by Tiu et al. in 2005 [261]. A similar technique was used in context of CPA in [229] to mitigate misalignment problems. Barenghi et al. performed CEMA attacks on intervals of the FFT bins of the trace in order to determine leaking frequencies [18]. This technique was further extended and explored in context of performing SCA using SDRs by Montminy et al. [175]. Other techniques to find the most leaking frequencies are investigated in [170, 260].

9.5 Chapter conclusions

We introduced a novel approach to improve CEMA attacks, called Correlation Optimization (CO). In this approach, a ML model is trained to learn "encodings" of a set of EM traces, which are subsequently used in a CEMA attack. These encodings are optimized such that their Pearson correlation with the secret key is maximal, by minimizing the correlation loss function defined in this chapter. This is in contrast to previous works, where models are trained to classify individual EM traces into secret-key byte, intermediate, or Hamming values by optimizing the mean cross-entropy of the class probabilities.

The identity function (regular CEMA attack), one-layer and two-layer MLP models that we considered in this work were evaluated on the ASCAD benchmark datasets for SCA [201] in both the time and frequency domain. Our best model, the frequency-domain two-layer MLP, is on average able to find the correct key byte after considering 1,000 traces from the `ASCAD`, `ASCAD_desync50`, and `ASCAD_desync100` datasets. We believe this is a significant improvement over the work of Prouff et al., where TA attacks, their "MLP best" model and "CNN best" model all failed to find the correct key byte for the `ASCAD_desync100` dataset after considering 5,000 traces. As such, this work contributes to **RG2** of this thesis.

In addition to evaluating our models on the ASCAD datasets, we examined their performance on a custom dataset as well. The custom dataset contains unprotected AES EM leakage traces of an Arduino Duemilanove recorded with a USRP B200 SDR sampling at 8 MS/s on the 64 MHz band. Even though the traces are highly dimensional and noisy, our frequency-domain two-layer MLP model is able to find the correct key after 22,000 traces, without requiring prior trace alignment. Finally, in consideration of **RG3** and in order to allow for reproducing the results that were presented in this chapter, all used

datasets and code have been published to Github at the following location: `https://github.com/rpp0/correlation-optimization-paper`.

Part IV

Conclusions and Future Work

Absolutely secure systems do not exist.

Adi Shamir

10 Conclusions

In this thesis, we defined and examined two types of information leakage in context of wireless communication: explicit and implicit information leakage. The former type pertains to information leakage that occurs as a result of design or implementation flaws in a protocol, whereas the latter is related to information leakage that stems from measurable side effects occurring during the execution of a specific protocol implementation.

As a first example of explicit information leakage, we revealed a vulnerability in the LEAP and PEAP protocols used in "WPA2-Enterprise" Wi-Fi networks, covering **RG1**. This vulnerability allows a proximal adversary to authenticate as a legitimate user without knowing the credentials of this user, by establishing a MITM position and performing a LEAP challenge to the mobile device while simultaneously initiating a PEAP protocol with the AS. We showed that the resulting LEAP MSCHAPv1 challenge response can be reused as a MSCHAPv2 challenge response in the PEAP protocol. Moreover, we showed that the adversary can even participate in regular communication with the AP, since the session keys are derived from the TLS master secret. In light of **RG2**, we developed a PoC tool and showed that all Apple devices prior to OS X Yosemite, Apple TV 7 and iOS 8 were vulnerable to the attack[1]. To achieve **RG3**, the vulnerability

[1]See the security advisories at respectively `https://support.apple.com/en-us/HT203112`,

was reported to Apple through a responsible disclosure procedure, allowing them to fix the issue in subsequent releases of OS X and iOS (see CVE-2014-4364). In addition, we proposed 5 mitigation strategies that can be implemented by network administrators and users themselves. Apple mitigated the issue by disabling LEAP by default in future releases.

After a further examination of the 802.11n and later Wi-Fi standards, we also discovered a vulnerability in the way frame aggregation is handled, allowing an adversary to leak payload data to the headers and lower layers of the OSI model (**RG1**). The vulnerability is caused by the A-MPDU parsing algorithm detailed in [127] and can be exploited similarly to the PHY-layer PIP attack introduced by Goodspeed et al. in 2011 (see [105]): the adversary crafts a payload containing valid MAC-layer Wi-Fi frames, including 4-byte A-MPDU delimiter headers, and embeds this payload into any application-layer data. When the A-MPDU header of frames containing such a payload has been damaged while in transit to the receiver (for example due to frame collisions), the A-MPDU parsing algorithm will overflow into the payload data of the application layer and parse the Wi-Fi frame crafted by the adversary instead of the original frame created by the AP. We created an open source PoC to test for this attack (**RG2**). Next, we investigated the impact of frame size on the aggregation probability and proposed a simple linear model to predict the number of successful frame injections per sent A-MPDU, based on the aggregation probability and frame corruption probability. Lastly, we proposed 6 mitigations to this attack for network administrators and vendors, and compared them in terms of requirements and costs (**RG3**).

Given the rising use of smartphones, Wi-Fi and wireless communication in general as highlighted in the introduction of this thesis, it is furthermore important to consider the *privacy* of the user as well. We therefore explored whether the MAC address randomization algorithms as implemented by smartphone vendors are sufficient to prevent involuntary location tracking by third parties (**RG2**). To do so, we first introduced two metrics based on information entropy: the variability and stability of a fingerprint. These metrics allow to identify, for each bit in a 802.11 frame, the suitability of this bit for use in a per-device fingerprint (**RG1**). The uniqueness of resulting fingerprints was evaluated in a practical experiment where 8 MSs were deployed at the music festival "Glimps" in Ghent in 2015 (**RG2**). We found that for small sets of 50 to 100 devices, the fingerprint is 80.0 to 67.6 percent unique, whereas for large sets of 1,000 to 10,000 devices the fingerprint is 33.0 to 15.1 percent unique.

`https://support.apple.com/en-us/HT203058`, and `https://support.apple.com/en-us/HT201395`.

Since the capability of an adversary to track a certain target depends on the transmission frequency of the target's device, we also investigated whether this frequency can be artificially increased by transmitting "stimulus frames" to the target (**RG1**). As the result of examining 12 frame types, an experiment in a controlled environment (**RG2**) demonstrated that `GAS Request` frames and `Block Acknowledgement` frames are particularly suited for this purpose. However, it should be mentioned that at the time of performing the experiment, GAS was still a very new service (introduced in 802.11u), meaning its lack of support back then has likely increased over the years.

Using the findings from the fingerprinting and stimulus frame experiments, we composed a methodology for effectively tracking mobile devices even when MAC address randomization is enabled. This was achieved by linking `Probe Response` frames with random MACs together using their fingerprint until the true MAC address is revealed (for example, when connecting to an AP). Next, we discussed several countermeasures that can be applied to reduce the effectiveness of such tracking techniques (**RG3**).

Besides the explicit information leakage in `Probe Requests` of Wi-Fi frames that reveals which device performed a transmission, we identified a type of implicit information leakage in the LoRa PHY-layer modulation scheme that can be used for similar purposes. We showed that this information leakage occurs due to small chirp frequency offset errors between different devices that originate from manufacturing defects (**RG1**). An adversary with access to the PHY-layer LoRa signal, for example by using an SDR, can thus analyze the frequency error of chirps to fingerprint individual devices. Since for this reason we do not want to correct for frequency errors upon receiving LoRa messages as opposed to conventional LoRa hardware, we moreover developed a custom demodulation technique and open source GNU Radio-based decoder implementation that allows to synchronize to multiple LoRa devices regardless of any frequency errors imposed on the signal (**RG2**). However, we also showed that this technique is less resistant to Gaussian noise and therefore requires a high SNR. We demonstrated the feasibility of fingerprinting PHY-layer LoRa signals by training SVM, MLP and CNN models on 8 different datasets of LoRa chirps, where the best classifier is capable of classifying the 22 individual devices with 93.33% – 99.0% accuracy on test sets that were recorded on the same day and location as the training sets. If the test sets are recorded at a later time, the accuracy drops to 58.67% – 76.80% due to changes in the channel conditions, meaning the channel conditions are implicitly fingerprinted alongside the device itself. Furthermore, we showed that even if a classifier has never seen a training example of a specific device, we can perform clustering techniques such as DBSCAN on the output space of a neural network

to perform zero-shot classification of these devices (**RG2**).

In the final chapter of this thesis, we considered implicit information leakages in the form of EM emanations that occur during the computations of cryptographic primitives used in wireless protocols, such as AES. We showed that such leakages can be learned using ML and proposed a novel methodology to train neural networks on EM traces, called CO. Instead of performing a classification of traces, our methodology instead transforms them to an encoding that is suitable for use in a standard CEMA attack, thereby improving its effectiveness compared to previous works while at the same time requiring less complex architectures (**RG2**). Furthermore, we demonstrated that by combining the idea of our methodology with the use of frequency domain features as mentioned in the work of Tiu et al. [261], the requirement having to align the EM traces can be removed (**RG2**). All datasets and tools developed to capture and process the EM traces using SDRs have been op sourced in consideration of **RG3**, thereby allowing the research community to reproduce the acquired results or test for novel EM side-channel attacks in a different context.

To conclude, an overview of all contributions that resulted from this research, including those not discussed in this thesis, is given along with a brief description of their impact in Table 10.0.1.

Table 10.0.1: An overview of all contributions and their impact.

Contribution	Impact
Robyns et al. [215] (**C1**)	CVE-2014-4364, a vulnerability that affects all Apple devices prior to iOS 8, OS X Yosemite, and Apple TV 7. The issue was resolved by Apple by disabling LEAP by default as an alternative EAP authentication method. A tool `peapwn` was released on Github to test for vulnerability to this attack (23 stars, 6 forks). Cited in 19 subsequent academic works.
Robyns et al. [216] (**C2**)	Discovery of a frame injection vulnerability in the 802.11n and following standards, affecting all devices that implement frame aggregation according to the standard. This issue gained attention on Reddit's netsec forum, where it reached the 23rd highest score of all time[a] at the time of discovery. A tool named `aggr-inject` was released on Github to test for vulnerability to this attack (264 stars, 44 forks). Another tool, `scapy-fakeap` was released to allow for easier vulnerability testing (183 stars, 45 forks). Cited in 1 subsequent academic work.
Robyns et al. [217]	Released online tool at `https://wicability.net` that allows the security community to look up the prevalence of a certain IE / security feature in order to assess the impact of discovered vulnerabilities. Cited in 2 subsequent academic works.
Robyns et al. [218] (**C3**, **C4**, **C5**)	New MAC-layer fingerprinting techniques, metrics and datasets. The code is available at the `wifi-mac-tracking` repository (13 stars, 6 forks). The datasets are publicly available on CRAWDAD and Zenodo. Some of the suggested countermeasures are now implemented in mainstream mobile devices. Cited in 13 subsequent academic works.
Robyns et al. [220] (**C6**, **C7**)	The first full description of the LoRa PHY layer and open-source decoder `gr-lora` (189 stars, 52 forks). A dataset of test samples and the code for reproducing the results have been released on the `lora-decoder-paper` repository (5 stars). Cited in 10 subsequent academic works.
Robyns et al. [219] (**C8**)	A novel approach to fingerprint LoRa signals based on PHY-layer features alone, using machine learning and, optionally, zero-shot learning. A virtual machine containing all code, datasets, and scripts to reproduce the results has been published on the `lora-phy-fingerprinting` repository (9 stars, 3 forks). A talk about LoRa and fingerprinting was given at FOSDEM 2018. Cited in 25 subsequent academic works.
Robyns et al. [221] (**C9**)	A novel technique to attack EM traces using ML that can be applied in the frequency domain. Training and test sets released to Github on the `correlation-optimization-paper` repository in both processed and unprocessed form (5 stars). A framework for performing EM analysis with ML called `emma` was also made available (25 stars, 2 forks). A talk about EM attacks using ML was given at FOSDEM 2019. Cited in 3 subsequent academic works.
Di Martino et al. [78]	In context of the GDPR, it is demonstrated that manual verification of subject access request credentials poses a number of security risks due to social engineering and credential spoofing. A talk about these issues was given at the IAPP Europe Data Protection Congress 2019. Cited in 3 subsequent academic works.

[a] See `https://www.reddit.com/r/netsec/comments/3bq96e/vulnerability_in_80211n_standard_allows_remote/`

11

Future Work

Generally, advances to the state-of-the-art of information security research hinge on the identification and exploration of new flaws in the established assumptions of a program's behavior. We have discussed a number of such flaws in the work that was presented in this thesis. However, as hinted on in Chapter 1, the true extent of a program's capabilities remains difficult to determine, since this would require testing *all* possible inputs rather than only a subset of inputs that is derived from functional requirements. In addition, the demand for more and more features in programs implies an increased complexity, which in turn results in a further expansion of the set of possible inputs and hence of the attack surface. *Complexity is the worst enemy of security.* This is not only true for the field of wireless security, but also for web security[1], network security or even physical security. So long as each input of every program has not been tested, there will be potential for researchers to discover new vulnerabilities that could compromise the security or privacy of users. Hence, there will always be potential for future work.

A consequence of the aforementioned issue is that there may be undiscovered vulnerabilities hidden in systems that are widely used today, and so we might ask how to uncover them effectively. For the case of explicit information leakage, and more specifically for wireless protocol design or software implementations, note that the discovery of a vulnerability usually starts from a hypothesis regarding the response of a program given a certain input. For example, one might ask: "How would a Wi-Fi frame aggregation implementation respond if a valid A-MPDU header is embedded in the payload of a frame?". This hypothesis can be

[1] In recent web security research it has for example been demonstrated that by delaying TLS messages in HTTP/2 and measuring their size, encrypted webpages can be fingerprinted [77, 264]. This shows how complex protocol interactions can lead to information leakage.

devised based on, for instance, a random guess, a manual code review or based on similar hypotheses that also lead to vulnerabilities in the past. Therefore, an interesting avenue of future research would be in the domain of fuzzing: generating as much useful hypotheses as possible, without having to explore the entire space of possibilities. It may also be interesting to find new ways of providing formal verification for such complex systems, in order to mitigate attacks. Language-Theoretic Security (LangSec) [227], which we briefly discussed in Section 5.5.4, is an example of one of the more recent attempts at providing such verification by treating expected inputs as a formal language. Similar efforts towards providing verification can be observed in other domains as well, e.g., the Everest project in context of the HTTPS ecosystem [25].

For implicit information leakage such as side-channel attacks, the situation is different. Here, the adversary knows that the physical properties measured through (a combination of) side channels *always* depend on the inputs provided to a program, which constitutes an information leakage vulnerability. To see why this is true, consider a hypothetical adversary who is capable of measuring the voltage on each component and interconnect of an integrated circuit. Theoretically, such a powerful adversary would be able to fully reconstruct the state of a program by observing the measured voltages, regardless of any side-channel measures that were taken. It is interesting to note that the area of "white-box cryptography" research considers this adversarial model, and aims to achieve practical security through the use of various obfuscation techniques that make key extraction computationally intractable [35, 56]. Fortunately, in a practical SCA scenario, adversaries are typically not as powerful, i.e., they are more limited in terms of measurement capabilities and operate in a black-box environment.

A successful defence against implicit information leakage is one which makes it *infeasible*[2]) for an adversary to measure a statistical difference of physical properties between different sensitive inputs, given their resources. Therefore, to test the true limits of a practical adversary, it would be useful to further research improvements regarding the measurement and analysis of implicit leakage[3]. This is evidenced by new sources of side-channel leakage that continue to be discovered over the years, such as the recent "screaming channels" by Camurati et al. [43] or the Meltdown and Spectre attacks in other domains of SCA [138, 154].

[2]Infeasible here does not necessarily mean impossible. For example, an adversary may eventually be able to perform a successful attack, but only after capturing an unrealistic number of traces.

[3]An interesting example in white-box cryptography where the discovery of such improvements leads to a successful attack is the differential computation analysis attack introduced by Bos et al. [35].

Regarding analysis techniques, an interesting and promising avenue seems the use of more advanced ML and DL techniques to extract more complex features from leakage measurements [119, 201]. Since these techniques have outperformed classical approaches at inferring patterns from highly-dimensional data in the multiple computer vision domains [151], their usage may result in similar benefits in the domains of SCA and fingerprinting. More specifically, it would be useful to look at the application of landmark detection in context of SCA. This would remove the requirement to manually add triggers for the separation of signals in a trace set. Another appealing use case can be found in super-resolution techniques: by combining the information from multiple low-resolution leakage traces and trained, high-resolution leakage models, perhaps it is possible to perform successful attacks even when leakage traces are captured at a sample rate significantly lower than the clock rate of the device under attack.

Finally, concerning measurement techniques, one could consider adversaries with more advanced measurement hardware in future work. For example, adversaries that use antenna or probe arrays to respectively fingerprint wireless transceivers and perform side-channel attacks. This would allow to not only extract features from the phase or amplitude of a signal, but also from its angle of arrival, which could contain additional information.

Part V

Appendices

A

Nederlandse Samenvatting

Kwetsbaarheden in de informatieveiligheid van draadloze communicatie worden veroorzaakt door onverwacht gedrag in een systeem dat voortvloeit uit gebreken in het ontwerp of de implementatie van een protocol. Deze kwetsbaarheden kunnen door een aanvaller worden uitgebuit om de confidentialiteit, integriteit of beschikbaarheid van een systeem te schaden, of om de privacy van gebruikers te ondermijnen. In deze thesis wordt gefocust op een specifieke klasse van kwetsbaarheden, namelijk informatielekken, in context van twee draadloze protocollen: Wi-Fi (802.11) en LoRa. Gegeven dat volgens voorspellingen meer dan 454 miljoen Wi-Fi hotspots en 500 000 LoRa gateways operationeel zullen zijn tegen 2020, behoren deze protocollen tot de meest populaire draadloze protocollen die momenteel in gebruik zijn. In deze thesis wordt een onderscheid gemaakt tussen twee types informatielekken, namelijk expliciete en impliciete informatielekken.

Het eerste deel van de thesis behandelt expliciete informatielekken, dewelke hun oorsprong vinden in onvoorziene gebreken in het ontwerp of de implementatie van draadloze protocollen. Specifiek wordt een kwetsbaarheid toegelicht in het 802.1X PEAP protocol, dat als authenticatiemethode gebruikt wordt in WPA2-Enterprise netwerken. Deze kwetsbaarheid staat een aanvaller toe om challenge responses door te sturen vanuit een LEAP handshake als geldige credential voor een aparte PEAP handshake. Zodoende kan een aanvaller toegang verschaffen tot het netwerk zonder in het bezit te zijn van geldige authenticatiegegevens. Er wordt aangetoond dat deze aanval werkt op alle Apple toestellen die zijn uitgebracht vóór het verschijnen van iOS 8, OS X Yosemite en Apple TV 7. Als volgende wordt in de thesis het MAC-layer frame aggregatiemechanisme van Wi-Fi netwerken bestudeerd, dat geïntroduceerd werd in de 802.11n standaard. Het resultaat van deze studie is de ontdekking van een nieuwe kwetsbaarheid, die

een aanvaller toestaat om vanop afstand (bijvoorbeeld over het internet) Wi-Fi frames te injecteren in een open netwerk door het "delimiter scanning" algoritme van Wi-Fi toestellen uit te buiten. Specifiek kan deze aanval worden uitgevoerd door een speciale payload in de applicatielaag van een pakket te voorzien, die lekt naar de lagere lagen van de netwerkstack wanneer de A-MPDU delimiter van een frame wordt beschadigd door incidentele ruis. Ten slotte wordt in het deel over expliciete informatielekken een analyse gemaakt van de informatie die wordt verstuurd in Wi-Fi `Probe Request` frames. Hierbij wordt aangetoond dat deze frames voldoende informatie bevatten om het toestel dat de frame verzond uniek te identificeren, zelfs wanneer privacybeschermende maatregelen zoals MAC-adres randomisatie werden toegepast. De data voor deze analyse werd vergaard in ons onderzoekscentrum en op het muziekfestival Glimps in 2015. Verder worden ook enkele nieuwe technieken besproken om meer transmissies te eliciteren, door bijvoorbeeld varianten van `GAS Request` en `ADDBA Request` frames uit te sturen naar Wi-Fi toestellen. Voor de ontdekte kwetsbaarheden worden een aantal tegenmaatregelen voorgesteld om hun impact te beperken en om de security en privacy van gebruikers te verbeteren, zoals bijvoorbeeld het reduceren van informatie in `Probe Requests` en vermijden om tijdens netwerkscans de SSID uit te sturen.

Het tweede deel van de thesis richt zich op impliciete informatielekken. Deze ontstaan inherent uit meetbare neveneffecten van een software of hardware implementatie van een protocol. Als eerste voorbeeld van zulke informatielekken wordt aangetoond hoe verschillen in de frequentie-afwijking van LoRa symbolen voldoende informatie geven om LoRa toestellen uniek te identificeren op de fysieke laag van de netwerkstack. Aanvullend hierop wordt een open source implementatie van een nieuwe demodulatietechniek voor LoRa voorgesteld, die toestaat om te synchroniseren met LoRa symbolen terwijl aanwezige frequentie-afwijkingen behouden blijven. Via deze techniek werden 8 datasets van LoRa symbolen vergaard met behulp van een Software Defined Radio. Vervolgens wordt voor elk van deze datasets besproken met welke nauwkeurigheid symbolen kunnen worden geclassificeerd volgens zender, door enkel gebruik te maken van hun fysieke radiosignaal. Hiervoor worden SVM, MLP en CNN classifiers uit het domein van machine learning gebruikt. Verder wordt kort aangetoond dat sommige toestellen zelfs correct geclassificeerd kunnen worden zonder te beschikken over transmissies van dit toestel tijdens de trainingsfase. Tot slot worden impliciete informatielekken in de vorm van elektromagnetische (EM) straling beschouwd. Concreet wordt er gekeken naar de EM-straling die lekt tijdens het uitvoeren van AES, een cryptografisch algoritme dat gebruikt wordt in Wi-Fi en LoRa. De toepassing van machine learning en deep learning wordt in deze context onderzocht en een nieuwe methodologie om de geheime AES-sleutel te vinden wordt

tevens beschreven. Deze methodologie vereist slechts enkele minuten trainingstijd op courante hardware omwille van een minder complexe architectuur, terwijl de performantie ten opzichte van state-of-the-art deep learning algoritmes op de ASCAD benchmark dataset wordt verbeterd. Bijkomend wordt de gebruikelijke vereiste om signalen te aligneren in tijd versoepeld door de methodologie toe te passen in het frequentiedomein van de EM-signalen. Hier wordt ten slotte een praktische "proof of concept" van gegeven waarbij gebruik wordt gemaakt van een USRP B210 om een AES-implementatie op een Arduino Duemilanove aan te vallen.

B

Public Datasets and Code

The following datasets and code repositories were released to the public domain, in accordance with **C10**:

scapy-fakeap Fake, monitor-mode wireless AP implementation using Python and Scapy, intended for convenient testing of 802.11 protocols and implementations. `https://github.com/rpp0/scapy-fakeap`

peapwn A proof-of-concept implementation of the LEAP relay attack described in Chapter 4. `https://github.com/rpp0/peapwn`

ath9k-linux Modified ath9k kernel modules that add support for sending custom debugging commands to `ath9k_htc` devices. `https://github.com/rpp0/linux`

open-ath9k-htc-firmware Modified `ath9k_htc` firmware that allows the user to set a fixed data rate, write and read registers or print data to `dmesg` using the kernel modules from **ath9k-linux**. This was used to perform the fixed data rate experiments from Chapter 5. `https://github.com/rpp0/open-ath9k-htc-firmware`

aggr-inject Proof-of-concept implementation of the A-MPDU MAC-layer injection attack detailed in Chapter 5. `https://github.com/rpp0/aggr-inject`

wifi-mac-tracking Python implementation of the bit entropy analysis, deanonymization algorithm, and transmission instigation techniques described in Chapter 6. `https://github.com/rpp0/wifi-mac-tracking`

wifi-mac-tracking (datasets) Datasets used for the experiments described in Chapter 6. `https://zenodo.org/record/545970`

lora-decoder-paper Scripts for reproducing the PDR, SNR and compatibility experiments from Chapter 7. `https://github.com/rpp0/lora-decoder-paper`

lora-decoder-paper (datasets) Datasets for reproducing the PDR and SNR experiment's figures from Chapter 7, to be used in conjunction with the **lora-decoder-paper** scripts. `https://research.edm.uhasselt.be/pro\byns/lora/amcsdlms-datasets.zip`

python-loranode High-level interface for sending serial commands to the RN-2486 LoRa module. `https://github.com/rpp0/python-loranode`

gr-lora Code for the GNU Radio based LoRa demodulator implementation from Chapter 7. `https://github.com/rpp0/gr-lora`

lora-phy-fingerprinting Code and virtual machine with trained models and datasets for reproducing the results and figures from Chapter 8. `https://github.com/rpp0/lora-phy-fingerprinting`

correlation-optimization-paper Code and datasets required for reproducing the results presented in Chapter 9. This includes the AES EM leakage dataset from the Arduino Duemilanove. `https://github.com/rpp0/correlation-optimization-paper`

emma A tool for the distributed analysis of EM leakage, supporting various classical attacks such as CEMA, as well as the CO method described in Chapter 9. `https://github.com/rpp0/emma`

electric-unicorn A tool based on the `unicorn` engine that simulates the EM leakage of a binary executable. `https://github.com/rpp0/electric-unicorn`

C

Scientific Contributions and Publications

The contents of the following scientific contributions are included in this thesis:

[**Robyns et al., 2019**] Pieter Robyns, Peter Quax, Wim Lamotte. Improving CEMA using Correlation Optimization. IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES). Atlanta, USA. 2019 (1), 1-24. (A2)

[**Robyns et al., 2018**] Pieter Robyns, Peter Quax, Wim Lamotte, William Thenaers. A Multi-Channel Software Decoder for the LoRa Modulation Scheme. Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security. Madeira, Portugal. 2018. (C1)

[**Robyns et al., 2017**] Pieter Robyns, Eduard Marin, Wim Lamotte, Peter Quax, Dave Singelée and Bart Preneel. Physical-Layer Fingerprinting of LoRa devices using Supervised and Zero-Shot Learning. Proceedings of the 10th ACM Conference on Security & Privacy in Wireless and Mobile Networks. Boston, MA, USA. 2017. (C1)

[**Robyns et al., 2017**] Pieter Robyns, Bram Bonné, Peter Quax and Wim Lamotte. Non-cooperative 802.11 MAC layer fingerprinting and tracking of mobile devices, Security and Communication Networks. Hindawi. 2017. (A1)

[**Robyns et al., 2015**] Pieter Robyns, Peter Quax and Wim Lamotte. Injection Attacks on 802.11n MAC Frame Aggregation. Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks. New York, NY, USA. 2015. (C1)

[**Robyns et al., 2014**] Pieter Robyns, Bram Bonné, Peter Quax and Wim Lamotte. Exploiting WPA2-enterprise Vendor Implementation Weaknesses

Through Challenge Response Oracles. Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks. Oxford, United Kingdom. 2014. (C1)

The following scientific contributions are not included in this thesis:

[**Di Martino et al., 2019**] Mariano Di Martino, Pieter Robyns, Winnie Weyts, Peter Quax, Wim Lamotte, Ken Andries. Personal Information Leakage by Abusing the GDPR "Right of Access". Proceedings of the Fifteenth Symposium on Usable Privacy and Security (SOUPS). Santa Clara, CA, USA. 2019. (C1)

[**Di Martino et al., 2018**] Mariano Di Martino, Pieter Robyns, Peter Quax, Wim Lamotte. IUPTIS: A Practical, Cache-resistant Fingerprinting Technique for Dynamic Webpages. Proceedings of the 14th International Conference on Web Information Systems and Technologies - Volume 1: WEBIST. Seville, Spain. 2018. (C1)

[**Robyns et al., 2017**] Pieter Robyns, Peter Quax, and Wim Lamotte. Opinion: PHY-Layer Security is no Alternative to Cryptography. Proceedings of the 10th ACM Conference on Security & Privacy in Wireless and Mobile Networks. Boston, MA, USA. 2017. (C1)

[**Robyns et al., 2016**] Pieter Robyns, Bram Bonné, Peter Quax and Wim Lamotte. Poster: Assessing the Impact of 802.11 Vulnerabilities using Wicability. Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks. Darmstadt, Germany. 2016. (C2)

D

Non-academic Publications and Press

The following is a list of non-academic publications and press-related events pertaining to the valorization of research results presented in this thesis.

Belgische student ondekt lek bij Apple (2014) National television item on VTM News about the research presented in Chapter 4. `http://nieuws.vtm.be/binnenland/106218-belgische-student-ontdekt-lek-bij-apple`.

Vulnerability in 802.11n allows remote frame injection (2015) Reddit discussion in `/r/netsec` about the vulnerability described in Chapter 5, which reached the 23rd highest score of all time at the time of publication. `https://shorturl.at/knvC6`.

LoRa software demodulation fingerprinting (2016) Poster presented at the iMinds Superminds conference in Brussels, see Figure D.0.1.

LoRa demodulation and AES EM attacks with SDR (2018) Presentation at FOSDEM 2018 about the work presented in Chapter 7 and Chapter 8. `https://archive.fosdem.org/2018/schedule/event/sdr_lora_aes/`.

Low-cost EM attacks using RTL-SDR and ML (2019) Presentation at FOSDEM 2019 about the work presented in Chapter 9. `https://archive.fosdem.org/2019/schedule/event/sdr_em_sidechannel_attacks/`.

Bedrijven gaan slordig om met onze gegevens (2019) National television item on VTM News about flawed user credential verification methods in context of GDPR subject access requests. This work was performed in collaboration with Mariano Di Martino. `https://m.hln.be/nieuws/binnenland/bedrijven-gaan-slordig-om-met-onze-gegevens~a2210e6b/`.

LoRa Software Demodulation and PHY Layer Fingerprinting

Pieter Robyns*, Eduard Marin†
*UHasselt (EDM) – tUL– iMinds †KU Leuven – ESAT-COSIC – iMinds

What is LoRa?

- LoRa is a modulation method designed for Low-Power Wide-Area Networks (LPWANs)
- Developed and patented by Semtech
- Transmission range between 2 – 15 km and data rate from 0.3 to 50 kbps [2]
- Built-in AES encryption on the MAC layer (LoRaWAN specification [1])
- Typically used in sensor networks with a star topology, where LoRa gateways relay packets to a network server for further processing
- Examples: temperature, parking, pressure, positioning sensors

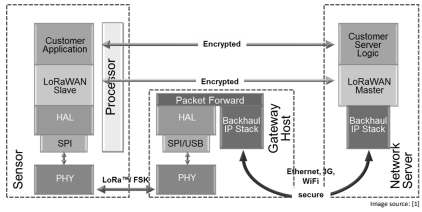


Image source: [3]

PHY Layer Security

- The PHY layer is the lowest layer in the OSI model
- Goal: to provide authentication on this layer without the use of standard cryptographic primitives, comparable to biometric fingerprinting
- Each wireless radio has unique features, which are imparted due to small errors in the manufacturing process
- Use cases exist for both offensive and defensive purposes
- **Offensive**
  - Impersonate a device by cloning its fingerprint
  - Track a device using positioning algorithms
- **Defensive**
  - Implicit identification and authentication of messages
- Challenges:
  - Identify and exploit useful features
  - Defend against spoofing attacks by powerful attackers
  - Fingerprint in noisy environments

References

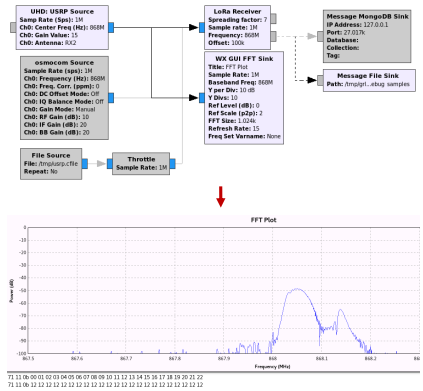
[1] The LoRa Alliance™, *LoRa Alliance™ Technology*, https://www.lora-alliance.org/What-Is-LoRa/Technology, accessed September 2016.

[2] Microchip, *RN2483 (433/868 MHz) LoRa™ Modem*, http://ww1.microchip.com/downloads/en/DeviceDoc/70005219A.pdf, accessed September 2016.

[3] Flaticon, Free vector icons, http://www.flaticon.com/.

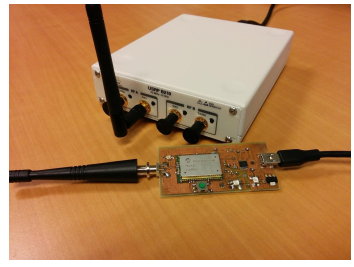
Software Demodulation

- Demodulating LoRa in software:
  - Allows us to access the PHY layer header, sync, and preamble
  - Allows for receiving frames offline / in simulations
  - Removes need for LoRa modem in order to check correct operation of the network
- Challenges:
  - Some implementation details not included in patent → reverse engineering was required (PHY header, sync, coding, whitening)
- Implemented as "GNU Radio" blocks: gr-lora
  - Support for all SFs and CRs
  - Available for free at **https://www.github.com/rpp0/gr-lora**



Demo

- Receiver: Software Defined Radio (SDR). Any SDR can be used e.g. USRP, HackRF, RTL-SDR, etc.
- Transmitter: RN2483 LoRa radio chip on custom module



Icons designed by "Madebyoliver" and distributed by Flaticon [3].

*Contact: {name}.{surname}@uhasselt.be
†Contact: {name}.{surname}@kuleuven.be

Figure D.0.1: Poster presented about our LoRa decoder at the iMinds Superminds 2016 conference in Brussels.

References

[1] 3GPP. LTE Release 15. https://www.3gpp.org/release-15, 2019. Accessed: 3 June 2019.

[2] 3GPP. LTE Release 13. https://www.3gpp.org/release-13, 2019. Accessed: 3 June 2019.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[4] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[5] Michael Abbott-Jard, Harpal Shah, and Ashish Bhaskar. Empirical evaluation of Bluetooth and WiFi scanning for road transport. In *36th Australasian Transport Research Forum (ATRF)*, page 14, 2013.

[6] ABI Research. 161 Million Consumer Wi-Fi Access Points Shipped in 2013; 802.11ac Sales Rapidly Accelerating. https://www.abiresearch.com/press/1391-million-consumer-wi-fi-access-\points-shipped-. Accessed: 2015.

[7] Adafruit. Adafruit Feather 32u4 specification. https://www.adafruit.com/product/3078, 2017. Accessed: 11 January 2017.

[8] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-Channel(s). In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems -*

CHES, pages 29–45, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-36400-9.

[9] Islam Alyafawi, Desislava C Dimitrova, and Torsten Braun. Real-time passive capturing of the GSM radio. In *IEEE International Conference on Communications (ICC)*, pages 4401–4406. IEEE, 2014.

[10] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. On the reliability of wireless fingerprinting using clock skews. In *Proceedings of the Third ACM Conference on Wireless Network Security*, pages 169–174. ACM, 2010.

[11] Cédric Archambeau, Eric Peeters, F-X Standaert, and J-J Quisquater. Template attacks in principal subspaces. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 1–14. Springer, 2006.

[12] Nadarajah Asokan, Valtteri Niemi, and Kaisa Nyberg. Man-in-the-middle in tunnelled authentication protocols. In *Security Protocols*, pages 28–41. Springer, 2005.

[13] Atheros Communications, Malinen, J. wpa_supplicant configuration options. `https://w1.fi/cgit/hostap/plain/wpa_supplicant/wpa_supplicant.conf`, . Accessed: February 10, 2016.

[14] Atheros Communications, Malinen, J. wpa_supplicant official source code repository. `https://www.w1.fi/cgit/hostap/tree/src/p2p/p2p_sd.c?id=fb09ed338919db09f3990196171fa73b37e7a17f#n384`, . Accessed: February 10, 2016.

[15] Azzy's Electronics. LoRaWAN RN2483/RN2903 breakout board specification. `https://www.tindie.com/products/DrAzzy/lorawan-rn2483rn2903-breakout-board-assembled/`, 2017. Accessed: 11 January 2017.

[16] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. Acoustic Side-Channel Attacks on Printers. In *USENIX Security symposium*, pages 307–322, 2010.

[17] Paramvir Bahl and Venkata N Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 775–784, 2000.

[18] Alessandro Barenghi, Gerardo Pelosi, and Yannick Teglia. Information leakage discovery techniques to enhance secure chip design. In *International Workshop on Information Security Theory and Practices (IFIP)*, pages 128–143. Springer, 2011.

[19] Andrea Barisani and Daniele Bianco. Sniffing Keystrokes with Lasers / Voltmeters. *BlackHat USA, August*, 2009. `https://www.blackhat.com/presentations/bh-usa-09/BARISANI/BHUSA09-Barisani-Keystrokes-SLIDES.pdf`.

[20] Andrea Barisani and Daniele Bianco. Fully arbitrary 802.3 packet injection: maximizing the Ethernet attack surface. *BlackHat USA, August*, 2013. `http://dev.inversepath.com/download/802.3/whitepaper.txt`.

[21] K. Bauer, H. Gonzales, and D. McCoy. Mitigating Evil Twin Attacks in 802.11. In *IEEE International Performance, Computing and Communications Conference (IPCCC)*, pages 513–516, Dec 2008. doi: 10.1109/PCCC.2008.4745081.

[22] George Becker, J Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, G Kenworthy, T Kouzminov, A Leiserson, M Marson, Pankaj Rohatgi, et al. Test vector leakage assessment (TVLA) methodology in practice. In *International Cryptographic Module Conference*, volume 1001, page 13, 2013.

[23] Richard Bellman. A new type of approximation leading to reduction of dimensionality in control processes. *Journal of Mathematical Analysis and Applications*, 27(2):454–459, 1969.

[24] Elyes Ben Hamida, Guillaume Chelius, and Jean Marie Gorce. Impact of the physical layer modeling on the accuracy and scalability of wireless network simulation. *Simulation*, 85(9):574–588, 2009.

[25] Karthikeyan Bhargavan, Barry Bond, Antoine Delignat-Lavaud, Cédric Fournet, Chris Hawblitzel, Catalin Hritcu, Samin Ishtiaq, Markulf Kohlweiss, Rustan Leino, Jay Lorch, et al. Everest: Towards a verified, drop-in replacement of HTTPS. In *2nd Summit on Advances in Programming Languages (SNAPL)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[26] Giuseppe Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[27] Bastian Bloessl, Christoph Leitner, Falko Dressler, and Christoph Sommer. A GNU radio-based IEEE 802.15.4 testbed. *GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN)*, pages 37–40, 2013.

[28] Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. An IEEE 802.11 a/g/p OFDM Receiver for GNU Radio. In *Proceedings of the Second Workshop on Software Radio Implementation Forum*, pages 9–16. ACM, 2013.

[29] Bastian Bloessl, Christoph Sommer, Falko Dressler, and David Eckhoff. The scrambler attack: A robust physical layer attack on location privacy in vehicular networks. In *International Conference on Computing, Networking and Communications (ICNC)*, pages 395–400. IEEE, 2015.

[30] Bluetooth SIG, Inc. Bluetooth Core Specification v5.0, December 2016.

[31] Josh Blum. LoRa SDR project. `https://github.com/myriadrf/LoRa-SDR`, 2017. Accessed: 10 October 2017.

[32] Lilian Bohy, Michael Neve, David Samyde, and Jean-Jacques Quisquater. Principal and Independent Component Analysis for Crypto-systems with Hardware Unmasked Units. In *Proceedings of e-Smart*. Citeseer, 2003.

[33] Bram Bonné, Arno Barzan, Peter Quax, and Wim Lamotte. WiFiPi: Involuntary tracking of visitors at mass events. In *IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, 2013.

[34] Bram Bonné, Peter Quax, and Wim Lamotte. Your Mobile Phone is a Traitor! – Raising Awareness on Ubiquitous Privacy Issues with SASQUATCH. *International Journal on Information Technologies & Security*, 6(3), 2014.

[35] Joppe W Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen. Differential computation analysis: Hiding your white-box designs is not enough. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 215–236. Springer, 2016.

[36] Sergey Bratus, Cory Cornelius, David Kotz, and Daniel Peebles. Active behavioral fingerprinting of wireless devices. In *Proceedings of the First ACM Conference on Wireless Network Security*, pages 56–61. ACM, 2008.

[37] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer, 2004.

[38] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, pages 116–127. ACM, 2008.

[39] Julien Brouchier, Tom Kean, Carol Marsh, and David Naccache. Temperature attacks. *IEEE Security & Privacy*, 7(2):79–82, 2009.

[40] Johnny Cache. Fingerprinting 802.11 implementations via statistical analysis of the duration field. `http://www.uninformed.org/?v=5&a=1&t=pdf`, 2006. Accessed: 22 November 2019.

[41] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.

[42] Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. Device-to-device communications with Wi-Fi Direct: overview and experimentation. *IEEE Wireless Communications*, 20(3):96–104, 2013.

[43] Giovanni Camurati, Sebastian Poeplau, Marius Muench, Tom Hayes, and Aurélien Francillon. Screaming channels: when electromagnetic side channels meet radio transceivers. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 163–177. ACM, 2018.

[44] Aldo Cassola, William Robertson, Engin Kirda, and Guevara Noubir. A Practical, Targeted, and Stealthy Attack Against WPA Enterprise Authentication. In *Proceedings of NDSS*, volume 2013, 2013.

[45] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi. Long-Range Communications in Unlicensed Bands: the Rising Stars in the IoT and Smart City Scenarios. *IEEE Wireless Communications*, 23, October 2016.

[46] Adrian Chadd. Atheros ath9k transmit path documentation. `https://github.com/erikarn/ath9k-docs/blob/master/ath9k-xmit.txt`. Accessed: 2015.

[47] Gayathri Chandrasekaran, John-Austen Francisco, Vinod Ganapathy, Marco Gruteser, and Wade Trappe. Detecting identity spoofs in IEEE 802.11e wireless networks. In *Global Telecommunications Conference (GLOBECOM)*, pages 1–6. IEEE, 2009.

[48] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[49] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[50] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES*, pages 13–28, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-36400-9.

[51] Periklis Chatzimisios, Anthony C Boucouvalas, and Vasileios Vitsas. Performance analysis of IEEE 802.11 DCF in presence of transmission errors. In *IEEE International Conference on Communications*, volume 7, pages 3854–3858. IEEE, 2004.

[52] M. Chernyshev, C. Valli, and P. Hannay. On 802.11 Access Point Locatability and Named Entity Recognition in Service Set Identifiers. *IEEE Transactions on Information Forensics and Security*, 11(3):584–593, March 2016. ISSN 1556-6013.

[53] François Chollet et al. Keras. `https://keras.io`, 2015.

[54] François Chollet et al. Keras Input layer implementation. Github. `https://github.com/keras-team/keras/blob/9080613dbc6f0840d7544bccc416121f0864a7fd/keras/engine/input_layer.py#L114`, 2019. Accessed: 17 September 2019.

[55] Omar Choudary and Markus G Kuhn. Efficient Template Attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 253–270. Springer, 2013.

[56] Stanley Chow, Philip Eisen, Harold Johnson, and Paul C Van Oorschot. White-box cryptography and an AES implementation. In *International Workshop on Selected Areas in Cryptography*, pages 250–270. Springer, 2002.

[57] Mark Ciampa. *CWNA Guide to Wireless LANs*. Cengage Learning, 2012. ISBN 978-1133132172.

[58] Cisco. Dictionary Attack on Cisco LEAP Vulnerability. `http://www.cisco.com/en/US/tech/tk722/tk809/technologies_security_notice09186a00801aa80f.html`, 2003.

[59] Cisco Systems. Number of public Wi-Fi hotspots worldwide from 2016 to 2022 (in millions). Statista - The Statistics Portal. https://www.statista.com/statistics/677108/global-public-wi-fi-hotspots/, 2019. Accessed: 3 July 2019.

[60] Cherita L Corbett, Raheem A Beyah, and John A Copeland. Using active scanning to identify wireless NICs. In *IEEE Information Assurance Workshop*, pages 239–246. IEEE, 2006.

[61] Cherita L Corbett, Raheem A Beyah, and John A Copeland. Passive classification of wireless NICs during active scanning. *International Journal of Information Security*, 7(5):335–348, 2008.

[62] Mathieu Cunche. I know your MAC Address: Targeted tracking of individual using Wi-Fi. *Journal of Computer Virology and Hacking Techniques*, 10(4):219–227, 2014.

[63] Mathieu Cunche, Mohamed-Ali Kaafar, and Roksana Boreli. Linking wireless devices using information contained in Wi-Fi probe requests. *Pervasive and Mobile Computing*, 11:56–69, 2014.

[64] Felipe Cunha, Leandro Villas, Azzedine Boukerche, Guilherme Maia, Aline Viana, Raquel AF Mini, and Antonio AF Loureiro. Data communication in VANETs: protocols, applications and challenges. *Ad Hoc Networks*, 44: 90–103, 2016.

[65] Adrian Dabrowski, Katharina Krombholz, Johanna Ullrich, and Edgar R Weippl. QR Inception: Barcode-in-Barcode Attacks. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 3–10. ACM, 2014.

[66] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[67] Boris Danev and Srdjan Capkun. Transient-based identification of wireless sensor nodes. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 25–36. IEEE Computer Society, 2009.

[68] Boris Danev, Thomas S. Heydt-Benjamin, and Srdjan Čapkun. Physical-layer Identification of RFID Devices. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 199–214, Berkeley, CA, USA, 2009. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855768.1855781.

[69] Boris Danev, Heinrich Luecken, Srdjan Capkun, and Karim El Defrawy. Attacks on physical-layer identification. In *Proceedings of the Third ACM Conference on Wireless Network Security*, pages 89–98. ACM, 2010.

[70] Boris Danev, Srdjan Capkun, Ramya Jayaram Masti, and Thomas S. Benjamin. Towards practical identification of HF RFID devices. *ACM Transactions on Information and System Security (TISSEC)*, 15(2):7:1–7:24, jul 2012. ISSN 1094-9224. doi: 10.1145/2240276.2240278. URL `http://doi.acm.org/10.1145/2240276.2240278`.

[71] Boris Danev, Davide Zanetti, and Srdjan Capkun. On physical-layer identification of wireless devices. *ACM Computing Surveys (CSUR)*, 45(1):6, 2012.

[72] Douglas SJ De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[73] Amine Dehbaoui, Victor Lomne, Philippe Maurine, Lionel Torres, and Michel Robert. Enhancing electromagnetic attacks using spectral coherence based cartography. In *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*, pages 135–155. Springer, 2009.

[74] Alan DeKok and Adam Sulmicki. Cisco LEAP protocol description. `http://freeradius.org/rfc/leap.txt`, 2001.

[75] Johannes Demel, Sebastian Koslowski, and Friedrich K Jondral. A LTE receiver framework using GNU Radio. *Journal of Signal Processing Systems*, 78(3):313–320, 2015.

[76] Loh Chin Choong Desmond, Cho Chia Yuan, Tan Chung Pheng, and Ri Seng Lee. Identifying unique devices through wireless fingerprinting. In *Proceedings of the First ACM Conference on Wireless Network Security*, pages 46–55. ACM, 2008.

[77] Mariano Di Martino, Peter Quax, and Wim Lamotte. Realistically Fingerprinting Social Media Webpages in HTTPS Traffic. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ACM, 2019.

[78] Mariano Di Martino, Pieter Robyns, Winnie Weyts, Peter Quax, Wim Lamotte, and Ken Andries. Personal Information Leakage by Abusing the GDPR "Right of Access". In *Fifteenth Symposium on Usable Privacy and Security (SOUPS)*, 2019.

[79] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol. RFC 5246, IETF, August 2008. URL `http://tools.ietf.org/html/rfc5246`.

[80] Tim Dierks and Eric Rescorla. RFC 5246: The Transport Layer Security (TLS) Protocol. *The Internet Engineering Task Force*, 2008.

[81] Dragino. LoRa/GPS HAT specification. `http://wiki.dragino.com/index.php?title=Lora/GPS_HAT`, 2017. Accessed: 11 January 2017.

[82] Ekahau. Asset tracking & management. `http://www.ekahau.com/real-time-location-system/solutions/healthcare/asset-tracking-management`. Accessed: January 19, 2016.

[83] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[84] Eurostat. Statistics database - individuals - mobile internet access [isoc_ci_im_i]. `https://ec.europa.eu/eurostat/web/digital-economy-and-society/data/database`, 2019. Accessed: 3 June 2019.

[85] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering: Design Principles and Practical Applications*. Wiley Publishing, 2010. ISBN 0470474246, 9780470474242.

[86] FIRST.Org, Inc. Common Vulnerability Scoring System v3.0. Technical Report v1.9, FIRST. URL `https://www.first.org/cvss/cvss-v30-specification_v1.9.pdf`.

[87] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie V Randwyk, and Douglas Sicker. Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting. In *USENIX Security*, 2006.

[88] Julien Freudiger. How Talkative is Your Mobile Device?: An Experimental Study of Wi-Fi Probe Requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '15, pages 8:1–8:6, New York, NY, USA, 2015. ISBN 978-1-4503-3623-9.

[89] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES*, pages 251–261, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44709-2.

[90] Matthew S. Gast. *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly, 2005. ISBN 978-0-596-10052-0.

[91] Matthew S. Gast. *802.11n: A Survival Guide*. O'Reilly, 2012. ISBN 978-1-449-31204-6.

[92] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[93] Catherine H Gebotys and Brian A White. EM analysis of a wireless Java-based PDA. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(4):44, 2008.

[94] Catherine H Gebotys, Simon Ho, and Chin Chi Tiu. EM analysis of Rijndael and ECC on a wireless Java-based PDA. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 250–264. Springer, 2005.

[95] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *International Cryptology Conference*, pages 444–461. Springer, 2014.

[96] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. Stealing keys from PCs using a radio: cheap electromagnetic attacks on windowed exponentiation. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 207–228. Springer, 2015.

[97] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get your hands off my laptop: physical side-channel key-extraction attacks on PCs. *Journal of Cryptographic Engineering*, 5(2):95–112, Jun 2015. ISSN 2190-8516. doi: 10.1007/s13389-015-0100-7. URL `https://doi.org/10.1007/s13389-015-0100-7`.

[98] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. ECDH key-extraction via low-bandwidth electromagnetic attacks on PCs. In *Cryptographers' Track at the RSA Conference*, pages 219–235. Springer, 2016.

[99] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 195–200. ACM, 2005.

[100] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 15–29. Springer, 2006.

[101] Boris Ginzburg and Alex Kesselman. Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11 n. In *Sarnoff Symposium*, pages 1–5. IEEE, 2007.

[102] Stuart Golden, Steve S Bateman, et al. Sensor measurements for Wi-Fi location with emphasis on time-of-arrival ranging. *IEEE Transactions on Mobile Computing*, 6(10):1185–1198, 2007.

[103] Travis Goodspeed. Phantom Boundaries and Cross-layer Illusions in 802.15. 4 Digital Radio. 2014.

[104] Travis Goodspeed and Sergey Bratus. 802.11 Packets in Packets, A Standard Compliant Exploit of Layer 1. In *28th Chaos Communications Congress*, pages 1–60, 2011.

[105] Travis Goodspeed, Sergey Bratus, Ricky Melgares, Rebecca Shapiro, and Ryan Speers. Packets in Packets: Orson Welles' In-Band Signaling Attacks for Modern Radios. In *WOOT*, pages 54–61, 2011.

[106] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. A testing methodology for side-channel resistance validation. In *NIST Non-Invasive Attack Testing Workshop*, 2011.

[107] Google. Privacy: MAC randomization. `https://source.android.com/devices/tech/connect/wifi-mac-randomization`, 2019. Accessed: 22 November 2019.

[108] Claire Goursaud and Jean-Marie Gorce. Dedicated networks for IoT: PHY/MAC state of the art and challenges. *EAI Endorsed Transactions on Internet of Things*, 2015.

[109] F. Gray. Pulse code communication, mar 1953. URL `https://www.google.com/patents/US2632058`. US Patent 2,632,058.

[110] Marco Gruteser and Dirk Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications*, 10(3):315–325, 2005.

[111] Sylvain Guilley, Philippe Hoogvorst, and Renaud Pacalet. Differential power analysis model and some results. *Smart Card Research and Advanced Applications VI*, pages 127–142, 2004.

[112] Fanglu Guo and Tzi-cker Chiueh. Sequence number-based MAC address spoof detection. In *Recent Advances in Intrusion Detection*, pages 309–329. Springer, 2005.

[113] A. Gupta, R. Cozza, and CK Lu. Market Share analysis: Mobile phones, worldwide, 4Q13 and 2013. *Gartner*, 2014.

[114] Mordechai Guri, Assaf Kachlon, Ofer Hasson, Gabi Kedma, Yisroel Mirsky, and Yuval Elovici. GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies. In *USENIX Security Symposium*, pages 849–864, 2015.

[115] Jeyanthi Hall, Michel Barbeau, and Evangelos Kranakis. Radio frequency fingerprinting for intrusion detection in wireless networks. *IEEE Transactions on Defendable and Secure Computing*, 2005.

[116] Jinsong Han, Chen Qian, Panlong Yang, Dan Ma, Zhiping Jiang, Wei Xi, and Jizhong Zhao. Geneprint: generic and accurate physical-layer identification for UHF RFID tags. *IEEE/ACM Transactions on Networking*, 24 (2):846–858, 2016.

[117] Changhua He and John C Mitchell. Analysis of the 802.11 i 4-way handshake. In *Proceedings of the 3rd ACM Workshop on Wireless Security*, pages 43–50. ACM, 2004.

[118] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[119] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. Cryptology ePrint Archive, Report 2019/143, 2019. `https://eprint.iacr.org/2019/143`.

[120] Guido R Hiertz, Dee Denteneer, Philips Lothar Stibor, Yunpeng Zang, Xavier Pérez Costa, and Bernhard Walke. The IEEE 802.11 universe. *IEEE Communications Magazine*, 48(1):62–70, 2010.

[121] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293, 2011.

[122] Ken Hutchison. Wireless Intrusion Detection Systems. *SANS Institute InfoSec Reading Room*, October 2004.

[123] Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. In *International Conference on Smart Card Research and Advanced Applications*. Springer, 2013.

[124] i-SCOOP. Wireless Internet of Things connectivity: LPWAN IoT forecasts 2017. https://www.i-scoop.eu/internet-of-things-guide/iot-spending-2020/wireless-iot-lpwan-forecasts-predictions-2017/, 2017. Accessed: 10 October 2017.

[125] IBM. Shorten the runway to smarter aircraft with AI-enabled IoT solutions. https://www.ibm.com/internet-of-things/explore-iot/vehicles/aircraft, 2019. Accessed: 7 June 2019.

[126] IEEE Computer Society. Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements. IEEE 802.11e Standard, 2005.

[127] IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE 802.11 Standard, September 2012.

[128] IEEE Computer Society. Amendment 5: Preassociation Discovery. IEEE 802.11aq Standard, June 2018.

[129] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[130] Suman Jana and Sneha K Kasera. On fast and accurate detection of unauthorized wireless access points using clock skews. *IEEE Transactions on Mobile Computing*, 9(3):449–462, 2010.

[131] Ira Ray Jenkins, Rebecca Shapiro, Sergey Bratus, Travis Goodspeed, Ryan Speers, and David Dowd. Speaking the Local Dialect: Exploiting differences between IEEE 802.15. 4 Receivers with Commodity Radios for fingerprinting, targeted attacks, and WIDS evasion. In *Proceedings of the 7th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 63–68, 2014.

[132] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[133] Pantea Kiaei, Darius Mercadier, Pierre-Evariste Dagand, Karine Heydemann, and Patrick Schaumont. SKIVA: Flexible and Modular Side-channel and Fault Countermeasures. Cryptology ePrint Archive, Report 2019/756, 2019. https://eprint.iacr.org/2019/756.

[134] Mikkel Baun Kjærgaard. A taxonomy for radio location fingerprinting. In *Location-and Context-Awareness*, pages 139–156. Springer, 2007.

[135] Mikkel Baun Kjærgaard, Mads Vering Krarup, Allan Stisen, Thor Siiger Prentow, Henrik Blunck, Kaj Grønbæk, and Christian S Jensen. Indoor positioning using Wi-Fi – how well is the problem understood? In *International Conference on Indoor Positioning and Indoor Navigation*, volume 28, page 31st, 2013.

[136] Matt Knight. Reversing LoRa: Exploring Next-Generation Wireless. GRCon, 2016. URL `https://static1.squarespace.com/static/54cecce7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf`.

[137] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48405-9.

[138] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *arXiv preprint arXiv:1801.01203*, 2018.

[139] Tadayoshi Kohno, Andre Broido, and Kimberly C Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[140] Boris Köpf and David Basin. An information-theoretic model for adaptive side-channel attacks. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 286–296. ACM, 2007.

[141] Memduh Köse, Selçuk Taşcıoğlu, and Ziya Telatar. The Effect of Transient Detection Errors on RF Fingerprint Classification Performance. In *Recent Advances in Circuits, Systems and Automatic Control*, pages 89–93, 2015.

[142] David Kotz, Tristan Henderson, and Chris McDonald. CRAWDAD: A Community Resource for Archiving Wireless Data At Dartmouth. `http://http://crawdad.org/`. Accessed: March 20, 2017.

[143] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999134.2999257`.

[144] Piotr Krysik. GNU Radio blocks and tools for receiving GSM transmissions. https://github.com/ptrkrysik/gr-gsm, 2017. Accessed: 10 October 2017.

[145] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

[146] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 646–651. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL http://dl.acm.org/citation.cfm?id=1620163.1620172.

[147] Larry Dignan. IBM launches Watson tools for agriculture. https://www.zdnet.com/article/ibm-launches-watson-tools-for-agriculture/, May 2019. Accessed: 7 June 2019.

[148] Laurent Butti. NetGear WG311v1 Wireless Driver 2.3.1 - 10 SSID Heap Buffer Overflow Vulnerability. http://www.exploit-db.com/exploits/29167/. Accessed: 2015.

[149] Quoc V. Le. A tutorial on deep learning part 1: nonlinear classifiers and the backpropagation algorithm, December 2015.

[150] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks*, pages 255–258, 1995.

[151] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[152] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *Journal of Cryptographic Engineering*, Apr 2017. ISSN 2190-8516. doi: 10.1007/s13389-017-0162-9. URL https://doi.org/10.1007/s13389-017-0162-9.

[153] Yuxia Lin and Vincent WS Wong. WSN01-1: frame aggregation and optimal frame size adaptation for IEEE 802.11n WLANs. In *Global Telecommunications Conference*, pages 1–6. IEEE, 2006.

[154] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *arXiv preprint arXiv:1801.01207*, 2018.

[155] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1067–1080, 2007.

[156] LoRa Alliance. LoRa Alliance home page. `https://www.lora-alliance.org/`, 2019. Accessed: 3 June 2019.

[157] Yao Lu. Unsupervised Learning on Neural Network Outputs. *CoRR*, abs/1506.00990, 2015. URL `http://arxiv.org/abs/1506.00990`.

[158] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[159] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016.

[160] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer US, Boston, MA, 2007. ISBN 978-0-387-38162-6. doi: 10.1007/978-0-387-38162-6_5. URL `https://doi.org/10.1007/978-0-387-38162-6_5`.

[161] Steve Mann and Simon Haykin. The Chirplet Transform: A Generalization of Gabor's Logon Transform. *Vision Interface '91*, pages 205–212, June 3-7 1991. ISSN 0843-803X.

[162] Eduard Marin, Dave Singelée, Flavio D Garcia, Tom Chothia, Rik Willems, and Bart Preneel. On the (in) security of the latest generation implantable cardiac defibrillators and how to secure them. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 226–236. ACM, 2016.

[163] Moxie Marlinspike. Divide and Conquer: Cracking MS-CHAPv2. `https://www.cloudcracker.com/blog/2012/07/29/cracking-ms-chap-v2/`, 2012.

[164] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. A study of MAC address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies*, 2017(4):365–383, 2017.

[165] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. Profiling power analysis attack based on MLP in DPA contest V4.2. In *39th International Conference on Telecommunications and Signal Processing (TSP)*, pages 223–226. IEEE, 2016.

[166] Mariano Di Martino, Pieter Robyns, Peter Quax, and Wim Lamotte. IUPTIS: A Practical, Cache-resistant Fingerprinting Technique for Dynamic Webpages. In *Proceedings of the 14th International Conference on Web Information Systems and Technologies - Volume 1: WEBIST*, pages 102–112. INSTICC, SciTePress, 2018. ISBN 978-989-758-324-7. doi: 10.5220/0007226501020112.

[167] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 145–167. Springer, 2019.

[168] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. Defeating MAC address randomization through timing attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 15–20. ACM, 2016.

[169] Thomas S Messerges, Ezzy A Dabbish, and Robert H Sloan. Investigations of power analysis attacks on smartcards. In *Proceedings of the USENIX Workshop on Smartcard Technology*, pages 17–17. USENIX Association, 1999.

[170] Olivier Meynard, Denis Réal, Sylvain Guilley, Florent Flament, Jean-Luc Danger, and Frédéric Valette. Characterization of the electromagnetic side channel in frequency domain. In *International Conference on Information Security and Cryptology*, pages 471–486. Springer, 2010.

[171] Marco Mezzavilla, Sourjya Dutta, Menglei Zhang, Mustafa Riza Akdeniz, and Sundeep Rangan. 5G mmWave Module for the ns-3 Network Simulator. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 283–290. ACM, 2015.

[172] Microchip Technology Inc. Low-Power Long Range LoRa Technology Transceiver Module. `http://ww1.microchip.com/downloads/en/DeviceDoc/50002346A.pdf`, 2015. Accessed: 10 October 2017.

[173] Microsoft. Cryptobinding. `http://msdn.microsoft.com/en-us/library/cc238384.aspx`.

[174] David P Montminy. *Enhancing Electromagnetic Side-Channel Analysis in an Operational Environment*. PhD thesis, Air Force Institute of Technology, 2013.

[175] David P Montminy, Rusty O Baldwin, Michael A Temple, and Mark E Oxley. Differential electromagnetic attacks on a 32-bit microprocessor using software defined radios. *IEEE Transactions on Information Forensics and Security*, 8(12):2101–2114, 2013.

[176] ABM Musa and Jakob Eriksson. Tracking unmodified smartphones using Wi-Fi monitors. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 281–294. ACM, 2012.

[177] Sashank Narain, Amirali Sanatinia, and Guevara Noubir. Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proceedings of the 7th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 201–212. ACM, 2014.

[178] Christoph Neumann, Olivier Heen, and Stéphane Onno. An empirical study of passive 802.11 device fingerprinting. In *32nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 593–602. IEEE, 2012.

[179] Andrew Ng and Kian Katanforoosh. CS229 lecture notes: Deep Learning, October 2018.

[180] Ludwig Nussel. The Evil Twin problem with WPA2-Enterprise. *SUSE Linux Products GmbH*, 2010.

[181] Colin O'Flynn and Zhizhang David Chen. Side channel power analysis of an AES-256 bootloader. In *28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 750–755. IEEE, 2015.

[182] Tamoghna Ojha, Sudip Misra, and Narendra Singh Raghuwanshi. Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges. *Computers and Electronics in Agriculture*, 118:66–84, 2015.

[183] Aaron Van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel Recurrent Neural Networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1747–1756, 2016.

[184] Michael Ossmann. Unambiguous Encapsulation. `https://www.mail-archive.com/langsec-discuss@mail.langsec.org/msg00000.html`, 2013. Accessed: 2015.

[185] Michael Ossmann and Dominic Spill. Unambiguous Encapsulation - Separating Data and Signaling. *Great Scott Gadgets Technical Report 2014-03-1*, 2014.

[186] Colin O'Flynn and Zhizhang Chen. Power analysis attacks against IEEE 802.15.4 nodes. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 55–70. Springer, 2016.

[187] C David Page and Sriraam Natarajan. Encyclopedia of Machine Learning and Data Mining. (2):1–24, 2014. doi: 10.1007/978-1-4899-7502-7. URL `http://link.springer.com/10.1007/978-1-4899-7502-7`.

[188] Ashwin Palekar, Dan Simon, Joe Salowey, Hao Zhou, Glen Zorn, and S. Josefsson. Protected EAP Protocol (PEAP). Work in Progress 6, IETF, March 2003. URL `http://tools.ietf.org/html/draft-josefsson-pppext-eap-tls-eap-06`.

[189] Ashwin Palekar, Dan Simon, Joe Salowey, Hao Zhou, Glen Zorn, and S. Josefsson. Protected EAP Protocol (PEAP) Version 2. Work in Progress 10, IETF, October 2004. URL `http://tools.ietf.org/html/draft-josefsson-pppext-eap-tls-eap-10`.

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

[191] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 802.11 user fingerprinting. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, pages 99–110. ACM, 2007.

[192] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. SoK: Security and privacy in machine learning. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE, 2018.

[193] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[194] Eldad Perahia. IEEE 802.11n development: history, process, and technology. *IEEE Communications Magazine*, 46(7):48–55, 2008.

[195] Stjepan Picek, Ioannis Petros Samiotis, Annelie Heuser, Jaehun Kim, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. Cryptology ePrint Archive, Report 2018/004, 2018. https://eprint.iacr.org/2018/004.

[196] Pieter Robyns. MAC frame aggregation injection implementation. https://github.com/rpp0/aggr-inject, . Accessed: April 2015.

[197] Pieter Robyns. Packet trace of the Beacon injection experiment. http://research.edm.uhasselt.be/~probyns/traces/beacon_inj.tar.gz, . Accessed: April 2015.

[198] Pieter Robyns. Packet trace of the remote host scan experiment. http://research.edm.uhasselt.be/~probyns/traces/inj_host_scan.tar.gz.tar.gz, . Accessed: April 2015.

[199] Thor S Prentow, Henrik Blunck, Kaj Gronbaek, and Mikkel B Kjærgaard. Estimating common pedestrian routes through indoor path networks using position traces. In *IEEE 15th International Conference on Mobile Data Management (MDM)*, volume 1, pages 43–48. IEEE, 2014.

[200] Emmanuel Prouff and Matthieu Rivain. A Generic Method for Secure SBox Implementation. In *International Workshop on Information Security Applications*, pages 227–244. Springer, 2007.

[201] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, page 53, 2018. URL http://eprint.iacr.org/2018/053.

[202] Pycom Ltd. LoPy 1.0 specification sheet. https://www.pycom.io/wp-content/uploads/2016/09/lopySpecsheetFinal.pdf, 2016. Accessed: 10 January 2017.

[203] Qualcomm Atheros. Atheros ath9k_htc transmit path documentation. https://github.com/qca/open-ath9k-htc-firmware/blob/master/target_firmware/wlan/if_owl.c#L1343. Accessed: 2015.

[204] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security*, E-SMART '01, pages 200–210, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42610-8. URL http://dl.acm.org/citation.cfm?id=646803.705980.

[205] Benjamin W Ramsey, Michael A Temple, and Barry E Mullins. PHY foundation for multi-factor ZigBee node authentication. In *Global Communications Conference (GLOBECOM)*, pages 795–800. IEEE, 2012.

[206] Benjamin W. Ramsey, Tyler D. Stubbs, Barry E. Mullins, Michael A. Temple, and Mark A. Buckner. Wireless infrastructure protection using low-cost radio frequency fingerprinting receivers. *International Journal of Critical Infrastructure Protection*, 8:27–39, 2015. ISSN 18745482. URL `http://dx.doi.org/10.1016/j.ijcip.2014.11.002`.

[207] K. Bonne Rasmussen and S. Capkun. Implications of radio fingerprinting on the security of sensor networks. In *Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm*, pages 331–340, Sept 2007. doi: 10.1109/SECCOM.2007.4550352.

[208] Christian Rechberger and Elisabeth Oswald. Practical Template Attacks. In *Information Security Applications*, pages 440–456, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31815-6.

[209] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-time Object Detection. In *Proceedings of the IEEE Conference On Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[210] Saeed Ur Rehman, Kevin Sowerby, and Colin Coghill. RF fingerprint extraction from the energy envelope of an instantaneous transient signal. In *Australian Communications Theory Workshop (AusCTW)*, pages 90–95. IEEE, 2012.

[211] Reilly Dunn. IoT Applications in the Automotive Industry. `https://www.iotforall.com/iot-applications-automotive-industry/`, November 2018. Accessed: 7 June 2019.

[212] KA Remley, CA Grosvenor, RT Johnk, DR Novotny, PD Hale, MD McKinley, A Karygiannis, and E Antonakakis. Electromagnetic signatures of WLAN cards and network security. In *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology*, pages 484–488. IEEE, 2005.

[213] Matthieu Rivain. On the exact success rate of side channel analysis in the Gaussian model. In *International Workshop on Selected Areas in Cryptography*, pages 165–183. Springer, 2008.

[214] Pieter Robyns. Online Resource 1: pcap file of the stimulus frame response conditions experiment. `http://research.edm.uhasselt.be/~probyns/mactracking/stimulus_experiment_results.pcap`, 2016.

[215] Pieter Robyns, Bram Bonné, Peter Quax, and Wim Lamotte. Exploiting WPA2-enterprise vendor implementation weaknesses through challenge response oracles. In *Proceedings of the 7th ACM Conference on Security & Privacy in Wireless and Mobile networks*, pages 189–194. ACM, 2014.

[216] Pieter Robyns, Peter Quax, and Wim Lamotte. Injection attacks on 802.11 n MAC frame aggregation. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, page 13. ACM, 2015.

[217] Pieter Robyns, Bram Bonné, Peter Quax, and Wim Lamotte. Assessing the Impact of 802.11 Vulnerabilities Using Wicability. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '16, pages 217–218, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4270-4. doi: 10.1145/2939918.2942421. URL `http://doi.acm.org/10.1145/2939918.2942421`.

[218] Pieter Robyns, Bram Bonné, Peter Quax, and Wim Lamotte. Noncooperative 802.11 MAC layer fingerprinting and tracking of mobile devices. *Security and Communication Networks*, 2017.

[219] Pieter Robyns, Eduard Marin, Wim Lamotte, Peter Quax, Dave Singelée, and Bart Preneel. Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning. In *Proceedings of the 10th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 58–63. ACM, 2017.

[220] Pieter Robyns, Peter Quax, Wim Lamotte, and William Thenaers. A Multi-Channel Software Decoder for the LoRa Modulation Scheme. In *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security*. SCITEPRESS, 2018.

[221] Pieter Robyns, Peter Quax, and Wim Lamotte. Improving CEMA using Correlation Optimization. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 1–24, 2019.

[222] Volker Roth, Wolfgang Polak, Eleanor Rieffel, and Thea Turner. Simple and Effective Defense Against Evil Twin Access Points. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 220–235, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-814-5. doi: 10.1145/1352533.1352569. URL `http://doi.acm.org/10.1145/1352533.1352569`.

[223] RTL-SDRangelove. Github project page. `https://github.com/hexameron/rtl-sdrangelove`, 2017. Accessed: 10 October 2017.

[224] Ehsan Saeedi, Md Selim Hossain, and Yinan Kong. Side channel analysis of an elliptic curve crypto-system based on multi-class classification. In *6th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2015.

[225] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, February 1978. doi: 10.1109/TASSP.1978.1163055.

[226] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[227] Len Sassaman, Meredith L Patterson, Sergey Bratus, and Michael E Locasto. Security applications of formal language theory. *IEEE Systems Journal*, 7(3):489–500, 2013.

[228] Harshad Sathaye, Domien Schepers, Aanjhan Ranganathan, and Guevara Noubir. Wireless attacks on aircraft instrument landing systems. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 357–372, Santa Clara, CA, Aug 2019. USENIX Association. ISBN 978-1-939133-06-9. URL `https://www.usenix.org/conference/usenixsecurity19/presentation/sathaye`.

[229] Oliver Schimmel, Paul Duplys, Eberhard Boehl, Jan Hayek, Robert Bosch, and Wolfgang Rosenstiel. Correlation power analysis in frequency domain. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*, 2010.

[230] Timothy M Schmidl and Donald C Cox. Robust frequency and timing synchronization for OFDM. *IEEE Transactions on Communications*, 45 (12):1613–1621, 1997.

[231] Bruce Schneier, Mudge, and David Wagner. Cryptanalysis of Microsoft's PPTP Authentication Extensions. *CQRE '99*, October 1999.

[232] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[233] O.B.A. Seller and N. Sornin. Low power long range transmitter, aug 2014. URL https://www.google.com/patents/EP2763321A1?cl=en. EP Patent App. EP20,130,154,071.

[234] Semtech. Semtech Acquires Wireless Long Range IP Provider Cycleo. http://investors.semtech.com/releasedetail.cfm?ReleaseID=655335, March 2012. Accessed: 9 October 2017.

[235] Semtech. SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver Datasheet. http://www.semtech.com/images/datasheet/sx1272.pdf, 2015. Accessed: 10 October 2017.

[236] Semtech. LoRa Modulation Basics. http://www.semtech.com/images/datasheet/an1200.22.pdf, 2015. Accessed: 9 October 2017.

[237] Semtech. LoRa overview page. https://www.semtech.com/lora, 2019. Accessed: 3 July 2019.

[238] Adi Shamir and Eran Tromer. Acoustic cryptanalysis. *presentation available from http://www.wisdom.weizmann.ac.il/~tromer*, 2004.

[239] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[240] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019.

[241] Sigfox. Sigfox home page. https://www.sigfox.com/, 2019. Accessed: 3 June 2019.

[242] Bertrik Sikken. Decoding LoRa. https://revspace.nl/DecodingLora, 2017. Accessed: 10 October 2017.

[243] Raul Siles. EAP dumb-down attack. In *RootedCON 2013*, pages 27–28. DinoSec, 2013.

[244] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013.

[245] Karen Simonyan, Sander Dieleman, Andrew Senior, and Alex Graves. WaveNet: A Generative Model for Raw Audio. pages 1–15, 2016.

[246] Arvind Singh, Monodeep Kar, Jong Hwan Ko, and Saibal Mukhopadhyay. Exploring power attack protection of resource constrained encryption engines using integrated low-drop-out regulators. In *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 134–139. IEEE, 2015.

[247] George Sklivanitis, Adam Gannon, Stella N Batalama, and Dimitris A Pados. Addressing next-generation wireless challenges with commercial software-defined radio platforms. *IEEE Communications Magazine*, 54(1): 59–67, 2016.

[248] Dionysios Skordoulis, Qiang Ni, Hsiao-Hwa Chen, Adrian P Stephens, Changwen Liu, and Abbas Jamalipour. IEEE 802.11 n MAC frame aggregation mechanisms for next-generation high-throughput WLANs. *IEEE Wireless Communications*, 15(1):40–47, 2008.

[249] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943, 2013.

[250] Robin Sommer and Vern Paxson. Outside the closed world: on using machine learning for network intrusion detection. In *IEEE Symposium on Security & Privacy*, pages 305–316. IEEE, 2010.

[251] N Sornin, M Luis, T Eirich, T Kramp, and O Hersent. LoRaWAN™ Specifications. *LoRa™ Alliance*, 2015.

[252] François-Xavier Standaert. How (not) to Use Welch's T-test in Side-Channel Security Evaluations. In *International Conference on Smart Card Research and Advanced Applications*, pages 65–79. Springer, 2018.

[253] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Advances in Cryptology - EUROCRYPT*, pages 443–461, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-01001-9.

[254] Stratistics Market Research Consulting Pvt Ltd. Wireless Connectivity - Global Market Outlook (2017-2023). `https://www.researchandmarkets.com/research/9g7hcj/global_wireless`, 2018. Accessed: 29 April 2019.

[255] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[256] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, 13(1): 63–78, 2017.

[257] The Linux Foundation. TX A-MPDU aggregation. `https://www.kernel.org/doc/htmldocs/80211/aggregation.html`. Accessed: 2015.

[258] The Radicati Group. Forecast number of mobile users worldwide from 2019 to 2023 (in billions). Statista - The Statistics Portal. `https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/`, 2019. Accessed: 3 June 2019.

[259] The Things Network. Gateway coverage map. `https://www.thethingsnetwork.org/map`, 2017. Accessed: 10 October 2017.

[260] Sébastien Tiran, Sébastien Ordas, Yannick Teglia, Michel Agoyan, and Philippe Maurine. A model of the leakage in the frequency domain and its application to CA and DA. *Journal of Cryptographic Engineering*, 4(3): 197–212, 2014.

[261] C Tiu. A New Frequency-Based Side Channel Attack for Embedded Systems. Master's thesis, University of Waterloo, 2005.

[262] O. Ureten and N. Serinken. Wireless security through RF fingerprinting. *Canadian Journal of Electrical and Computer Engineering*, 32(1):27–33, 2007. ISSN 0840-8688. doi: 10.1109/CJECE.2007.364330.

[263] Oktay Ureten and Nur Serinken. Wireless security through RF fingerprinting. *Canadian Journal of Electrical and Computer Engineering*, 32(1): 27–33, 2007.

[264] Tom Van Goethem, Mathy Vanhoef, Frank Piessens, and Wouter Joosen. Request and conquer: Exposing cross-origin resource size. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 447–462, 2016.

[265] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA*, pages 104–119, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-19074-2.

[266] Mathy Vanhoef and Frank Piessens. Advanced Wi-Fi attacks using commodity hardware. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 256–265. ACM, 2014.

[267] Mathy Vanhoef and Eyal Ronen. Dragonblood: Analyzing the Dragonfly handshake of WPA3 and EAP-pwd. In *IEEE Symposium on Security & Privacy*. IEEE, 2020.

[268] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 413–424. ACM, 2016.

[269] Mathy Vanhoef, Nehru Bhandaru, Thomas Derham, Ido Ouzieli, and Frank Piessens. Operating Channel Validation: Preventing Multi-Channel Man-in-the-Middle Attacks Against Protected Wi-Fi Networks. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 34–39. ACM, 2018.

[270] Angelos Vlavianos, Lap Kong Law, Ioannis Broustis, Srikanth V Krishnamurthy, and Michalis Faloutsos. Assessing link quality in IEEE 802.11 wireless networks: Which is the right metric? In *Personal, Indoor and Mobile Radio Communications*, pages 1–6, 2008.

[271] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir. Fingerprinting Wi-Fi Devices Using Software Defined Radios. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 3–14, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4270-4. doi: 10.1145/2939918.2939936. URL `http://doi.acm.org/10.1145/2939918.2939936`.

[272] Martin Vuagnoux and Sylvain Pasini. Compromising Electromagnetic Emanations of Wired and Wireless Keyboards. In *USENIX security symposium*, pages 1–16, 2009.

[273] K Wang, M Faulkner, J Singh, and I Tolochko. Timing synchronization for 802.11 a WLANs under multipath channels. In *Australasian Telecommunication Networks and Applications Conference (ATNAC)*, 2003.

[274] Weightless. Weightless home page. `http://www.weightless.org/`, 2019. Accessed: 3 June 2019.

[275] Wi-Fi Alliance. Wi-Fi CERTIFIED Passpoint. `https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-passpoint`, . Accessed: January 27, 2016.

[276] Wi-Fi Alliance. Product Finder search results. https://www.wi-fi.org/product-finder-results?sort_by=default&sort_order=desc&categories=4&capabilities=1, . Accessed: February 9, 2016.

[277] Wi-Fi Alliance. Wi-Fi 6 home page. https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-6, 2019. Accessed: 3 June 2019.

[278] Wi-Fi Alliance. Wi-Fi HaLow home page. https://www.wi-fi.org/discover-wi-fi/wi-fi-halow, 2019. Accessed: 3 June 2019.

[279] Wi-Fi Alliance Technical Committee, P2P Task Group. Wi-Fi Peer-to-Peer (P2P) v1.5. Technical Specification, Wi-Fi Alliance, August 2014.

[280] WiGLE.net. Statistics. https://wigle.net/stats. Accessed: March 9, 2016.

[281] Wireless Geographic Logging Database. Wi-Fi network statistics. https://wigle.net/stats. Accessed: 2015.

[282] Joshua Wright. FreeRADIUS-WPE. http://www.willhackforsushi.com/?page_id=37, 2008.

[283] R. Yamasaki, A. Ogino, T. Tamaki, T. Uta, N. Matsuzawa, and T. Kato. TDOA location system for IEEE 802.11b WLAN. In *IEEE Wireless Communications and Networking Conference*, volume 4, pages 2338–2343, March 2005.

[284] Zhimin Yang, Adam C. Champion, Boxuan Gu, Xiaole Bai, and Dong Xuan. Link-layer Protection in 802.11I WLANS with Dummy Authentication. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 131–138, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-460-7. doi: 10.1145/1514274.1514294. URL http://doi.acm.org/10.1145/1514274.1514294.

[285] Josh Yavor. The BYOD PEAP Show. In *DefCon 21*. iSEC Partners, 2013.

[286] Xin Ye. Side Channel Leakage Analysis - Detection, Exploitation and Quantification. 2015.

[287] Rama K Yedavalli and Rohit K Belapurkar. Application of wireless sensor networks to aircraft control and health management systems. *Journal of Control Theory and Applications*, 9(1):28–33, 2011.

[288] Jihwang Yeo and Ashok Agrawala. Packet error model for the IEEE 802.11 MAC protocol. In *Personal, Indoor and Mobile Radio Communications*, volume 2, pages 1722–1726. IEEE, 2003.

[289] Jong-Hoon Youn, Hesham Ali, Hamid Sharif, Jitender Deogun, Jason Uher, and Steven H Hinrichs. WLAN-based real-time asset tracking system in healthcare environments. In *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, page 71. IEEE, 2007.

[290] YongBin Zhou and DengGuo Feng. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. *IACR Cryptology ePrint Archive*, page 388, 2005.

[291] G. Zorn. Microsoft PPP CHAP Extensions, Version 2. RFC 2759, IETF, January 2000. URL `http://tools.ietf.org/html/rfc2759`.

[292] G. Zorn and S. Cobb. Microsoft PPP CHAP Extensions. RFC 2443, IETF, October 1998. URL `http://tools.ietf.org/html/rfc2433`.