Electromagnetic Side-Channel Analysis Methods for Digital Forensics on Internet of Things

Asanka P. Sayakkara

UCD Student Number: 16211801

A thesis submitted to University College Dublin
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

School of Computer Science

Head of School: Associate Professor Chris Bleakley

Primary Supervisor: Assistant Professor Nhien-An Le-Khac

Co-Supervisor: Assistant Professor Mark Scanlon

September – 2020

Abstract

Modern legal and corporate investigations heavily rely on the field of digital forensics to uncover vital evidence. The dawn of the Internet of Things (IoT) devices has expanded this horizon by providing new kinds of evidence sources that were not available in traditional digital forensics. However, unlike desktop and laptop computers, the bespoke hardware and software employed on most IoT devices obstructs the use of classical digital forensic evidence acquisition methods. This situation demands alternative approaches to forensically inspect IoT devices.

Electromagnetic Side-Channel Analysis (EM-SCA) is a branch in information security that exploits Electromagnetic (EM) radiation of computers to eavesdrop and exfiltrate sensitive information. A multitude of EM-SCA methods have been demonstrated to be effective in attacking computing systems under various circumstances. The objective of this thesis is to explore the potential of leveraging EM-SCA as a forensic evidence acquisition method for IoT devices. Towards this objective, this thesis formulates a model for IoT forensics that uses EM-SCA methods. The design of the proposed model enables the investigators to perform complex forensic insight-gathering procedures without having expertise in the field of EM-SCA. In order to demonstrate the function of the proposed model, a proof-of-concept was implemented as an open-source software framework called *EMvidence*. This framework utilises a modular ar-

chitecture following a *Unix philosophy*; where each module is kept minimalist and focused on extracting a specific forensic insight from a specific IoT device. By doing so, the burden of dealing with the diversity of the IoT ecosystem is distributed from a central point into individual modules.

Under the proposed model, this thesis presents the design, the implementation, and the evaluation of a collection of methods that can be used to acquire forensic insights from IoT devices using their EM radiation patterns. These forensic insights include detecting cryptography-related events, firmware version, malicious modifications to the firmware, and internal forensic state of the IoT devices. The designed methods utilise supervised Machine Learning (ML) algorithms at their core to automatically identify known patterns of EM radiation with over 90% accuracy.

In practice, the forensic inspection of IoT devices using EM-SCA methods may often be conducted during triage examination phase using moderately-resourced computers, such as laptops carried by the investigators. However, the scale of the EM data generation with fast sample rates and the dimensionality of EM data due to large bandwidths necessitate rich computational resources to process EM datasets. This thesis explores two approaches to reduce such overheads. Firstly, a careful reduction of the sample rate is found to be reducing the generated EM data up to 80%. Secondly, an intelligent channel selection method is presented that drastically reduces the dimensionality of EM data by selecting 500 dimensions out of 20,000.

The findings of this thesis paves the way to the noninvasive forensic insight acquisition from IoT devices. With IoT systems increasingly blending into the day-to-day life, the proposed methodology has the potential to become the lifeline of future digital forensic investigations. A multitude of research directions are outlined, which can strengthen this novel approach in the future.

To my mother, father, and Upeksha for everything.

Acknowledgements

I would like to thank my supervisor Dr. Nhien-An Le-Khac and co-supervisor Dr. Mark Scanlon for giving me this priceless opportunity and guiding me throughout this journey for the last few years. It has been a pleasure to work with them all this time and they have helped me to become a better researcher. I would also like to thank Dr. Catherine Mooney for serving as the chair of my Research Studies Panel (RSP) and for her invaluable comments that helped me to improve my work. I thank UCD for funding my PhD studies.

Working and studying at UCD was always enjoyable due to my friends who occupied the room A0.12 in computer science building. I thank all of them for keeping me accompanied. Among them, I especially thank Hamed Jahromi and Saad Alabdulsalam for always being there for me and for all the conversations we had about research, life, and everything else.

I would like to acknowledge the support I received from my colleagues Xiaoyu Du, Aikaterini Kanta, and Felix Anda from UCD Forensics and Security Research Group. They spent their time to help me improve my communication skills every time I was preparing for a presentation. I also thank my collaborators Luis Miralles-Pechuán and Quan Le from the Centre for Applied Data Analytics Research (CeADAR), UCD. Their expertise and guidance immensely broadened my knowledge and helped me to improve my work.

Leaving one's homeland and spending a few years abroad doing full-time research is never an easy task without the help of one's family. I would like to thank my wife, Upeksha, for loving me unconditionally and for forgiving me for my absent-mindedness. I also thank my parents and two brothers for their kindness and care.

Contents

Abstract ii

Acknowledgements v

Contents vi

List of Figures xii

List of Tables xvii

List of Publications xix

1 Introduction 1

1.1 Motivation 3

1.2 Research Problem 4

1.2.1 Extraction of Forensic Insights through Electromagnetic Side-Channel Analysis 4

1.2.2 Efficiency of Electromagnetic Side-Channel Analysis Methods 5

1.2.3 Management of the Diversity and Dynamism of Internet of Things Ecosystem 5

1.3 Thesis Approach 6

1.3.1 Designing and Implementing a Methodology as the *EMvidence* Framework 6
1.3.2 Designing and Evaluating Machine Learning-based Methods in the *EMvidence* Framework 7
1.3.3 Designing and Evaluating Methods to Increase the Efficiency of the *EMvidence* Framework 7
1.4 Contributions 7
1.5 Limitations 8
1.6 Thesis Organisation 9

2 Technical Background **10**
2.1 Internet of Things 10
2.2 Digital Forensics 12
2.2.1 Digital Investigation Process 12
2.2.2 Forensics of Internet of Things Devices 13
2.2.3 Encrypted Devices 14
2.3 Digital Signal Processing 15
2.3.1 The Nature of a Signal 15
2.3.2 Analog and Digital Signals 16
2.3.3 Time and Frequency Domains 17
2.3.4 The Visualisation of Signals 19
2.3.5 Tools and Libraries 19
2.4 Software-defined Radio 20
2.4.1 Software-defined Radio Architecture 20
2.4.2 Software-defined Radio Hardware Tools 21
2.4.3 Software-defined Radio Software Tools 23
2.4.4 The Nature of Software-defined Radio Data 24
2.5 Electromagnetic Side-Channel Radiation 25

2.5.1 Sources of Electromagnetic Side-Channels 25
2.5.2 Observation of Electromagnetic Side-Channels 26
2.5.3 Leakage of Critical Information 28
2.6 Machine Learning 29

3 Related Work **31**
3.1 Side-Channel Attacks 31
3.2 Unintentional Electromagnetic Radiation 34
3.2.1 Hardware that Causes Electromagnetic Radiation 34
3.2.2 Sampling Electromagnetic Radiation 35
3.2.3 The Connection between Instructions and Electromagnetic Radiation 36
3.3 Electromagnetic Radiation as a Signature 37
3.3.1 Electromagnetic Radiation as a Hardware Signature . . 37
3.3.2 Electromagnetic Radiation as a Software Signature . . . 38
3.4 Electromagnetic Radiation that Leak Information 41
3.4.1 Observable Electromagnetic Spectrum Patterns 41
3.4.2 Differential Electromagnetic Analysis 42
3.4.3 Analysis of Wireless-powered Devices 45
3.4.4 Countermeasures to Electromagnetic Side-Channel Analysis 47
3.5 Standards and Tools 48
3.6 Current Direction 49
3.6.1 Frequent Cryptographic Operations 50
3.6.2 Combined Side-Channel Attacks 50
3.6.3 File Signatures 51
3.6.4 Packet Analysis of Network Devices 52
3.6.5 Easy Access to Electromagnetic Spectrum 53

3.6.6 Backscatter Channels 54
3.6.7 Advancements in Machine Learning 54

4 Methodology: The Birth of EMvidence 56

4.1 Introduction 56
4.2 A Case Study Scenario 57
4.3 A Forensic Model for Internet of Things 61
4.3.1 Identification of Requirements 62
4.3.2 Planning for Data Acquisition and Analysis 62
4.3.3 Building New Analysis Methods 63
4.3.4 Acquiring Electromagnetic Data 63
4.3.5 Executing Electromagnetic Side-Channel Analysis . . . 63
4.3.6 Reporting Results 64
4.3.7 Overall Workflow 65
4.4 The *EMvidence* Framework 65
4.4.1 Data Acquisition Component 67
4.4.2 Report Generation Component 67
4.4.3 EMvidence Core 68
4.4.4 Implementation Details 68
4.5 Plug-ins for EMvidence 71
4.5.1 Plug-in Behaviour 71
4.5.2 Plug-in Development 73
4.6 Procedure for Data Acquisition 76
4.6.1 Representative Internet of Things Devices 76
4.6.2 Determining Data Acquisition Parameters 77
4.7 Experimental Plan 81
4.7.1 Designing Methods to Acquire Forensic Insights 81
4.7.2 Designing Methods to Increase Efficiency 81

5 Insights from Waves: Machine Learning Methods for EMvidence 82
5.1 Introduction . . . 82
5.2 Considerations for Experiments . . . 83
5.2.1 Types of Useful Insights . . . 84
5.2.2 Machine Learning Algorithms . . . 85
5.2.3 Preprocessing Procedure . . . 86
5.3 Experimental Evaluation . . . 87
5.3.1 The Cryptographic Activities of High-end Internet of Things Devices . . . 87
5.3.2 The Cryptographic Activities of Low-end Internet of Things Devices . . . 92
5.3.3 Firmware Version of Internet of Things Devices . . . 95
5.3.4 Malicious Modifications to the Firmware of Internet of Things . . . 100
5.3.5 Current Behavioural State of an Internet of Things Device 101
5.4 Discussion . . . 104

6 Curse of Dimensionality: Increasing the Efficiency of EMvidence 105
6.1 Introduction . . . 105
6.2 Considerations for Experiments . . . 106
6.3 Approach 1: Minimising Data Production . . . 107
6.3.1 Electromagnetic Data Processing Overhead . . . 107
6.3.2 Electromagnetic Data Storage Overhead . . . 109
6.3.3 Electromagnetic Data Transmission Overhead . . . 111
6.4 Approach 2: Selecting Useful Channels . . . 112
6.4.1 Procedure of Experiments . . . 115
6.4.2 Using 20,000 Channels . . . 118
6.4.3 Principal Component Analysis . . . 120

6.4.4 Channel Selection Based on the Variance 121
6.4.5 Channel Selection based on the Average 122
6.4.6 Applying Average per Class and Variance between the Classes 123
6.4.7 Applying Recursive Feature Elimination 124
6.4.8 Using a Time Window of 50 Timestamps 127
6.4.9 Summary of the Channel Selection Methods 128
6.5 Discussion 129

7 Conclusion & Future Work **131**
7.1 Conclusion 131
7.1.1 Implications of This Work 133
7.1.1 Future of Digital Forensics 133
7.1.2 Legal Acceptability 133
7.1.3 Platform for New Research 133
7.2 Future Work 134
7.2.1 Evaluation of Commonly-used Internet of Things 134
7.2.2 Interoperability between Evidence Sources 134
7.2.3 Management of Electromagnetic Data 135
7.2.4 Hardware Independence of Machine Learning Models . 135
7.2.5 Cryptographic Key Retrieval in Forensic Context 136

Bibliography **137**

Appendices **160**

A List of Abbreviations **160**

B EMvidence User Documentation **165**

List of Figures

1.1 The challenge of IoT forensics. 2
1.2 Application of EM-SCA for forensic inspection of IoT devices. . 3
1.3 The order of addressing research questions and the planned approach to address each question. 6

2.1 Components of an IoT device and a typical IoT network architecture. 11
2.2 Properties of a periodic signal. 16
2.3 Visualisation of a signal as waveform, power spectral density (PSD), and spectrogram. 18
2.4 The architecture of a software defined radio (SDR) platform. . . 20
2.5 RTL-SDR and HackRF One devices with their default antennas. 21
2.6 Near field antennas of varying diameters and lengths along with a semi-rigid RF cable. 22
2.7 GQRX SDR software capturing and visualising EM data from a connected HackRF One device. 23
2.8 Representation of an SDR data sample in polar and Cartesian coordinate systems. 24
2.9 Hardware setup to observe EM radiation from an Arduino device blinking an LED. 27

2.10 Spectrograms of AM demodulated EM radiation acquired from an Arduino device. In (a) and (c), the device is running two unique programs, while (b) depicts the transition period. 28

3.1 The RF-DNA fingerprinting process. 40
3.2 EM analysis of XOR-Cipher algorithm to extract the encryption key. 43

4.1 State machine of the IoT fire warning system's firmware. 58
4.2 Reasoning with the information of IoT device firmware internal state. 60
4.3 A forensic model for IoT forensics using EM-SCA methods. . . 61
4.4 A potential investigative workflow that follows the proposed forensic model. 64
4.5 Major functional components of the *EMvidence* framework and their involvement in the workflow of analysing an IoT device. . . 66
4.6 The output on the Bash terminal when running *EMvidence*. . . 69
4.7 Analysing an EM trace using the *EMvidence*. 70
4.8 A report generated by *EMvidence* after analysing EM data from a device. 71
4.9 A plug-in being called by the *EMvidence* framework's core. . . . 72
4.10 Instrumented and controlled EM signal acquisition. 73
4.11 The workflow of creating a plug-in for *EMvidence*. 75
4.12 Arduino and Raspberry Pi devices with H-loop antennas. 77
4.13 Leakage signals of two representative IoT devices – (a) Raspberry Pi 3 B+ at 1.4 GHz and (b) Arduino Leonardo at 288 MHz (18^{th} harmonic). 78

4.14 PSD of the EM radiation when running Bubble sort algorithm with two different antenna positions on Arduino. 79

5.1 Waveform of the AM demodulated signal at the CPU clock frequency of Raspberry Pi. The AM modulated signal represents the AES encryption performed on the device. Sudden higher peaks are an external interference signal coming from an unknown source. (The three sub-figures depict three zoomed-in scales of the same signal.) 87
5.2 The EM trace acquisition and preprocessing stages in order to classify cryptographic activities using an ML model. 88
5.3 Sample Fourier Transform vectors of cryptographic algorithms that run on Raspberry Pi. 90
5.4 The time it takes to digitally sign and verify a message using different ECC curves on an Arduino device. 93
5.5 Example ECC and non-ECC signals acquired from Arduino device. 94
5.6 Variation of classification accuracy against sliding window length and EM trace length. 95
5.7 Power spectral density (PSD) of the EM radiation from four different Arduino programs that were used for classification. . . . 97
5.8 Confusion matrix of the neural network classifier to detect ten different Arduino programs, which are labelled from 0 to 9. . . . 98
5.9 Confusion matrix of the neural network classifier to detect twenty different Arduino programs, which are labelled from 0 to 19. 99
5.10 The PSD plots of the IoT device's EM signal at different device states. 102

5.11 Confusion matrix of the IoT device state classifier. 103

6.1 The two experimental approaches to increase the efficiency of gathering forensic insights. 106
6.2 The variation of the number of sliding windows produced against the sliding window step size. 108
6.3 The effect of EM trace sample rate to the signal classification accuracy when used with 4 class classifier to identify four different Arduino programs. 110
6.4 The variation of EM data processing overhead against sample rate when used with 4 class classifier to identify four different Arduino programs. 111
6.5 Spectrogram of the observed EM signal from DUT 113
6.6 Waveform of some randomly selected channels of the EM dataset. 114
6.7 The workflow to generate EM traces, identify channels, and finally perform EM-SCA with selected channels. 115
6.8 The series of methods explored for channel selection. 117
6.9 The confusion matrix of classifying programs using all the channels. 119
6.10 Variance (y-axis) of the top 100 eigenvalues (x-values) when applying principal component analysis. 119
6.11 The variance (y-axis) of the 20,000 channels (x-axis). The limit of y-axis is set to 10^{-13} in order to visualise lower values. However, the variance of 5^{th} and 6^{th} channels are 0.000282 and 0.000222 respectively. 121
6.12 Average (y-axis) for each of the 20,000 channels (x-axis). . . . 122
6.13 Variance between the average of each of the classes for all the channels. 124

6.14 The optimal number of features (model with highest performance) for the RFE algorithm is 81 (marked with a red dot). . . 125
6.15 The confusion matrix for the 10 Activities and 81 features, i.e., the optimal number for RFE. 126
6.16 Summary of the experimental results with different techniques. 129

B.1 Login window of the *EMvidence* framework. 166
B.2 Dashboard interface of the *EMvidence* framework. 167
B.3 The interface for uploading EM data into *EMvidence* framework. 168
B.4 The interface for capturing EM data using the *EMvidence* framework. 169
B.5 The interface for analysing EM data using plug-ins on the *EMvidence* framework. 170
B.6 Settings interface of the *EMvidence* framework. 171

List of Tables

5.1 Classification accuracy of cryptographic algorithms. 91
5.2 Private and public key sizes of ECC curves. 92

6.1 Average accuracy per class using the entire 20,000 channels. . 118
6.2 Average accuracy per class for the best 100 PCA components. 120
6.3 Average accuracy per class of the highest 103 channels ordered by variance. 122
6.4 Average accuracy per class of the highest 100 channels ordered by average. 123
6.5 Average accuracy per class after calculating the variance between the average per activity. 125
6.6 Average accuracy per class of the selected 81 channels by RFE. 127
6.7 Result of the experiments when applying a time window of 50 samples. 128

Declaration of Authorship

I hereby certify that the submitted work is my own work, was completed while registered as a candidate for the degree of doctor of philosophy, and I have not obtained a degree elsewhere on the basis of the research presented in this submitted work.

Asanka Sayakkara	Date
(Student Number: 16211801)	

List of Publications

Journal Papers

- Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2020). Facilitating Electromagnetic Side-Channel Analysis for IoT Investigation: Evaluating the EMvidence Framework. Forensic Science International: Digital Investigation. (DFRWS Virtual USA, July 2020)

- Sayakkara, A., Miralles, L., Le-Khac, N-A., & Scanlon, M. (2020). Cutting through the Emissions: Feature Selection from Electromagnetic Side-Channel Data for Activity Detection. Forensic Science International: Digital Investigation. DOI: https://doi.org/10.1016/j.fsidi.2020.300927 (DFRWS Virtual EU, March 2020 - Best Student Paper Award)

- Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2020). EMvidence: A Framework for Digital Evidence Acquisition from IoT Devices through Electromagnetic Side-Channel Analysis. Forensic Science International: Digital Investigation. DOI: https://doi.org/10.1016/j.fsidi.2020.300907 (DFRWS Virtual EU, March 2020 - Best Poster Award)

- Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2019). Leveraging Electromagnetic Side-Channel Analysis for the Investigation of IoT Devices. Digital Investigation, Elsevier. DOI: https://doi.org/10.1016/j.diin.2019.04.012 (DFRWS, Portland, OR, USA, July 2019 - Best Student Paper Award)

- Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2019). A Survey of Electromagnetic Side-Channel Attacks and Discussion on their Case-Progressing Potential for Digital Forensics. Digital Investigation, Elsevier. DOI: https://doi.org/10.1016/j.diin.2019.03.002

Conference Papers

- Du, X., Hargreaves, C., Sheppard, J., Anda, F., Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2020), SoK: Exploring the State of the Art and the Future Potential of Artificial Intelligence in Digital Forensic Investigation, 13th International Workshop on Digital Forensics (WSDF), held at the 15th International Conference on Availability, Reliability and Security (ARES), Virtual Event. DOI: https://doi.org/10.1145/3407023.3407068

- Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2018). Accuracy Enhancement of Electromagnetic Side-channel Attacks on Computer Monitors. The 2nd International Workshop on Criminal Use of Information Hiding (CUING), held at the 13th International Conference on Availability, Reliability and Security (ARES), Hamburg, Germany. DOI: https://dx.doi.org/10.1145/3230833.3234690

- Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2018). Electromagnetic Side-Channel Attacks: Potential for Progressing Hindered Digital Forensic Analysis. International Workshop on Speculative Side Channel Analysis (WoSSCA 2018), held as part of the 32nd European Conference on Object-Oriented Programming (ECOOP) and the 27th International Symposium on Software Testing and Analysis (ISSTA), Amsterdam, Netherlands. DOI: https://dx.doi.org/10.1145/3236454.3236512

Poster Abstracts

- Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2018). Leveraging Electromagnetic Side-Channel Attacks for Digital Forensics. 32nd European

Conference on Object-Oriented Programming (ECOOP) and the 27th International Symposium on Software Testing and Analysis (ISSTA), Amsterdam, Netherlands. (Best Poster Award, UCD CS PhD Event, December 2018).

Doctoral Symposiums

- Sayakkara, A., Le-Khac, N-A., & Scanlon, M. (2018). Leveraging Electromagnetic Side-Channel Attacks for Digital Forensics. Doctoral Symposium held as part of the 32nd European Conference on Object-Oriented Programming (ECOOP) and the 27th International Symposium on Software Testing and Analysis (ISSTA), Amsterdam, Netherlands.

Chapter 1

Introduction

In the field of digital forensics, legal and corporate investigators seek to uncover mysteries using digital sources of evidence. People interact with computing devices while conducting their day-to-day business that leaves unintentional traces of their activities. Such sources of forensic evidence include computer hard disks, network activity logs, removable media, the internal storage of mobile phones and many others [1, 2]. When extracting digital evidence from computing devices, the investigators follow well-established procedures in order to ensure their court-admissibility [3]. A piece of evidence extracted from a device without following standard practices can be challenged in a court of law and consequently be rendered useless to support the investigation [4].

The latest addition to digital forensic evidence sources are Internet of Things (IoT) devices. IoT ecosystem includes a wide variety of devices such as smart-watches, smart TVs, CCTV cameras, medical implants, fitness wearables, etc. [5] With the increasing prevalence of IoT devices in everyday life, it is inevitable to find them in modern crime scenes and digital forensic investigations. For instance, a medical implant, such as a pacemaker, can provide hints in an investigation about a person of interest's physical exertion or stress introduced elevation of heart rate. A fitness wearable, such as a Fitbit, can provide a vital piece of information about the presence and movements of a person in a crime scene. A smart voice assistant device, such as an Amazon Alexa, can provide a vital information about the time its owner came home [6, 7, 8].

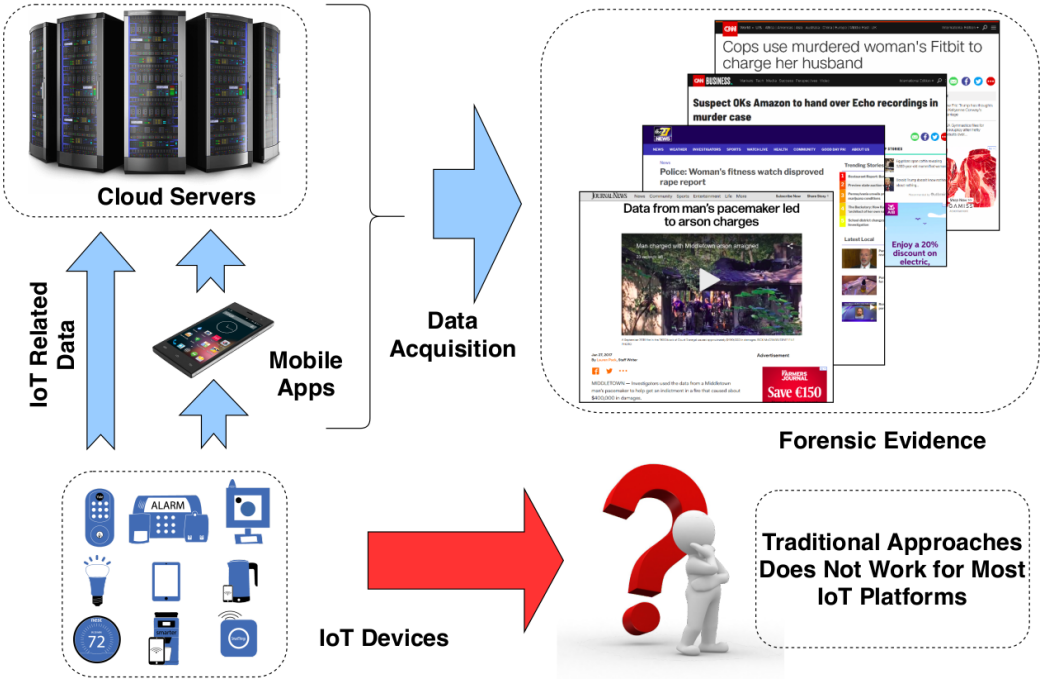

Figure 1.1: The challenge of IoT forensics.

Forensic evidence related to IoT devices is usually found on three different sources: cloud servers, smart mobile phones, and the IoT devices themselves (see Figure 1.1) [9]. Among them, the two former sources are the focus of cloud and mobile forensics domains respectively with developed and reliable methods. In contrast, direct forensic inspection of IoT devices is challenging due to multiple factors. Most IoT products use bespoke hardware and firmware that makes forensic evidence acquisition with classical methods impossible [10]. This situation has lead to the use of invasive techniques, such as physically removing storage chips and inspecting with the Scanning Electron Microscopes (SEM) [11, 12, 13]. The using of invasive methods poses a risk of damaging devices to unrecoverable extents. Furthermore, the vast diversity of IoT products in use and the frequent arrival of new products into the market further complicates the use of such invasive methods [14, 15].

This thesis aims to address this challenge of IoT forensics by identifying a novel methodology to directly inspect IoT devices in a non-invasive manner.

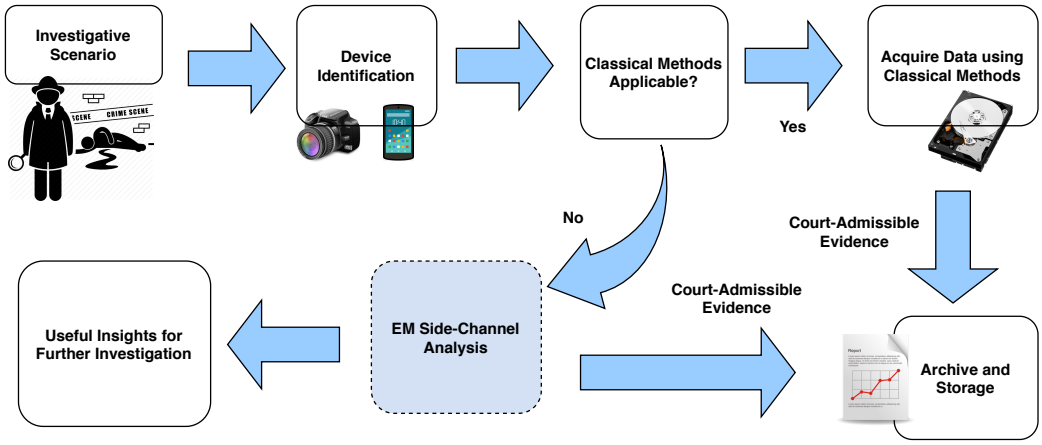

Figure 1.2: Application of EM-SCA for forensic inspection of IoT devices.

1.1 Motivation

Side-channel analysis has been proven to be effective against many security mechanisms on computing systems [16]. Among various side-channel attacks, Electromagnetic Side-Channel Analysis (EM-SCA) is an important class of attacks that exploits the unintentional Electromagnetic (EM) radiation from computers [17, 18, 19]. Using EM-SCA, sensitive information, such as cryptographic keys, have been exfiltrated from computers and IoT devices are no exception. When an investigator encounters IoT devices during a forensic investigation, it should be determined whether the device can be inspected using classical forensic approach, i.e., copying of the volatile and non-volatile storage and inspecting in a forensically-sound manner. The IoT devices that do not meet this condition, require an alternative approach.

This research proposes EM-SCA as an alternative approach for IoT forensics (see Figure 1.2). Due to the nature of EM-SCA, it is indeed a non-invasive approach to deal with IoT forensics. However, EM-SCA has its own limitations in the forensic context. Each EM-SCA method that has been published in the literature is tailor-made for acquiring a specific kind of information from a specific type of device. Equally, EM-SCA is only applicable when an IoT device is up and running during the inspection. Therefore, the practical application of EM-SCA for IoT forensics demands a large collection of tailor-made EM-SCA

methods at the disposal of an investigator, which should be applied as soon as a device is found in order to extract information that are most relevant to the case in hand.

Performance of such a complex procedure is currently impossible for digital forensic investigators. As a result, a need arises to introduce a methodology that enable successful application of EM-SCA for IoT forensics.

1.2 Research Problem

The research presented in this thesis seeks to uncover a methodology that enables the application of EM-SCA to extract forensic insights from IoT devices in digital investigation context. Towards addressing this problem, the following three questions were defined and pursued in this thesis.

1.2.1 Extraction of Forensic Insights through Electromagnetic Side-Channel Analysis

When performing EM-SCA to eavesdrop on computers, a multitude of approaches can be taken. It is necessary to identify methods that are effective at extracting forensic insights from IoT devices at various investigative scenarios.

> Research Question 1:
> What methods are effective at extracting forensic insights from IoT through EM-SCA?

Hypothesis: EM radiation of IoT devices leak sufficient information to provide forensic insight about their internal behaviour. It is possible to design and implement EM-SCA methods to extract such information in order to be used in digital investigation scenarios.

1.2.2 Efficiency of Electromagnetic Side-Channel Analysis Methods

When applying EM-SCA methods for the forensic inspection of IoT devices, they need to be performed within a reasonable time using reasonable amount of computational resources. The fact that EM data are both large in size and wide in dimensionality has a negative impact on this regard.

> Research Question 2:
> What approaches can increase the efficiency of the EM-SCA methods for forensic inspection of IoT?

Hypothesis: Although EM data are highly dimensional and large in size, it is possible to design and implement methods that can increase the efficiency of handling EM data. With such methods, it is possible to acquire forensic evidence from IoT devices in the field by forensic investigators with moderately-resourced computers.

1.2.3 Management of the Diversity and Dynamism of Internet of Things Ecosystem

The precise EM-SCA method, which should be used in a particular scenario, depends on the IoT device in question and the type of information being pursued. Therefore, it is necessary to define a methodology that enables effective management of individual EM-SCA methods in the forensic context.

> Research Question 3:
> What methodology can be used to manage the diversity and dynamism of IoT ecosystem when leveraging EM-SCA to acquire forensic insights?

Hypothesis: It is possible to design a unified methodology to perform EM-SCA in forensic investigation context, regardless of the specific IoT device or the specific forensic insight being extracted. Such a methodology can abstract

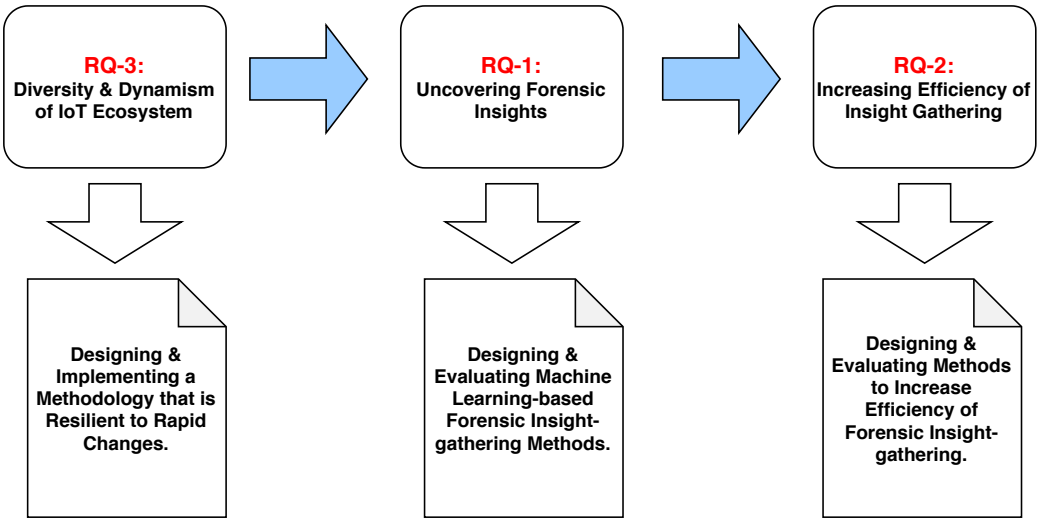

Figure 1.3: The order of addressing research questions and the planned approach to address each question.

out the rapid changes of the IoT ecosystem, enabling forensic investigators to perform EM-SCA seamlessly.

1.3 Thesis Approach

The first two research questions demand the discovery of methods that can be applied to specific types of IoT devices or scenarios. However, the practical usability of these methods depends on a successful answer to the latter question. Therefore, this thesis takes on the third research question as the entry point, which is followed by the first and the second research questions. Figure 1.3 illustrates this order of approaching research questions.

1.3.1 Designing and Implementing a Methodology as the *EMvidence* Framework

In order to address the third research question, a methodology was developed that distributes the complexity of dealing with diverse EM-SCA methods for gathering forensic insights. This methodology is inspired by the *Unix philos-*

ophy [20] of designing systems. A proof-of-concept of the proposed methodology is developed as an open-source software framework called *EMvidence*. This implementation is subsequently used as a platform to work towards answering the other two research questions.

1.3.2 Designing and Evaluating Machine Learning-based Methods in the *EMvidence* Framework

When applying EM-SCA to acquire forensic insights from IoT devices in investigative context, the process should require as minimum human intervention as possible. This is to say that analysis results should not depend on the theoretical or technical expertise of the investigators. In order to meet this requirement, the methodology proposed as part of this work defines the scope of exploring EM-SCA methods that use Machine Learning (ML) techniques. A collection of ML-based EM-SCA methods were designed for this purpose and implemented using the *EMvidence* framework for evaluation.

1.3.3 Designing and Evaluating Methods to Increase the Efficiency of the *EMvidence* Framework

Towards the goal of increasing efficiency of EM-SCA methods in practical investigative scenarios, two potential avenues were experimentally explored. The first is the potential of reducing the amount of EM data production by regulating sample rate as far as it does not negatively affect the functionality and accuracy of EM-SCA methods. The second is the potential to automatically identify the information-leaking channels from a wide band of EM data.

1.4 Contributions

The research presented in this thesis makes the following contributions:

1. Introduces EM-SCA as an alternative window to acquire forensic insights from IoT devices for digital forensic investigation purposes.

2. Designs and experimentally evaluates a set of ML-assisted EM-SCA methods that can provide forensic insights from EM radiation of IoT devices in digital forensic context.

3. Using empirical evaluations, identifies two methods to increase the efficiency of ML-assisted EM-SCA methods: the reduction of sample rate to a certain degree and the selection of information-leaking frequency channels intelligently.

4. Presents an IoT forensics model that enables the investigators to utilise a large collection of EM-SCA methods without theoretical or technical expertise on each individual EM-SCA method.

5. Implements an open-source software framework named *EMvidence* that follows the model presented in this work. The *EMvidence* framework features EM data collection, analysis and also can serve as a platform for future research and development in the field of EM-SCA for IoT forensics.

1.5 Limitations

The following limitations exists in the work presented in this thesis:

- In order to prevent information leakage through EM radiation, some IoT devices can employ hardware and software side-channel leakage mitigation techniques. This work only considers hardware and software that do not employ such techniques.

- In the experiments of this work, two types of representative IoT devices were used. While the two chosen platforms sufficiently emulate a wide range of real-world IoT hardware, future research should consider off-the-shelf IoT devices that are currently in general use.

1.6 Thesis Organisation

The rest of this thesis is organised as follows. In Chapter 2, the technical background of concepts required to follow this thesis are presented in detail. A comprehensive literature survey on the topic is presented in Chapter 3, which covers a broad range of related work in the field of EM-SCA. Chapter 4 illustrates the methodology for addressing high-level research problem. The EM-SCA methods for acquiring forensic insights from IoT devices are explored in Chapter 5. Chapter 6 is dedicated to the empirical evaluation of methods for increasing efficiency of EM-SCA methods. Finally, Chapter 7 concludes the thesis by summarising the results and highlighting the future work.

Chapter 2

Technical Background

This chapter introduces a high-level technical background of the areas that are necessary to follow later chapters. Section 2.1 introduces the nature of IoT devices from both hardware and software perspective that are important to consider when designing tools to inspect them in forensic contexts. Section 2.2 introduces the background of digital forensics and the challenge it faces with the emergence of IoT. The approach taken by this research requires the reader to be familiar with certain Digital Signal Processing (DSP) concepts and terminologies. Section 2.3 covers fundamental ideas around DSP depending on their relevance to this thesis research. EM data acquisition and analysis are at the centre of the research presented in this thesis. As the data acquisition is performed using Software-defined Radio (SDR) tools, Section 2.4 introduces the technical concepts and tools that are necessary for this purpose. Furthermore, Section 2.5 introduces key ideas and procedures related to EM-SCA. Finally, Section 2.6 briefly introduces key ideas of machine learning related to this research providing references for further reading.

2.1 Internet of Things

Internet of Things (IoT) is a term that refers to a broad spectrum of computing devices [21, 22]. Depending on the application, the hardware and software components of these devices may vary. However, at a high-level, an IoT device

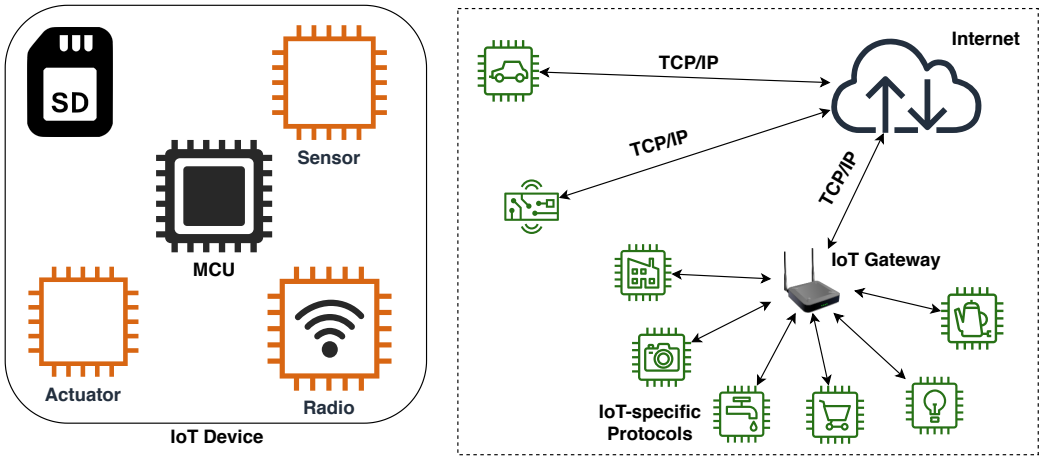

Figure 2.1: Components of an IoT device and a typical IoT network architecture.

consists of a few important hardware components that are of interest to EM-SCA. The heart of an IoT device is a Microcontroller Unit (MCU). An MCU is a tiny computer with various peripheral devices packed into a single Integrated Circuit (IC) chip. It stores and executes the firmware. The second most important component is the radio transceiver that helps the MCU to communicate with the outside world through a wireless network. Depending on the intended application, an IoT device may contain a number of sensors and actuators that are controlled by the MCU. Similarly, some IoT applications require the device to store a considerable amount of data outside the on-board flash ship and therefore, a Secure Digital (SD) card slot or an on-board flash chip may also present (see Figure 2.1). Meanwhile, System-on-Chip (SoC) hardware are single IC chips that integrates both MCU and radio transceiver into the same package [23]. The use of SoCs on IoT devices helps to save the space and the energy consumption, which are essential factors for an IoT platform.

Although the very name of IoT hints that it refers to devices that are connected to the internet, an IoT device is not necessarily connected to the internet directly. Communicating directly through the Internet with remote servers is an expensive move from multiple aspects [24]. From a computational resource point of view, direct Internet connectivity requires a functioning TCP/IP protocol stack running on the MCU of the device, which is memory and processing intensive. From an energy consumption point of view, direct Internet connec-

tivity requires support for protocols such as WiFi (IEEE 802.11) that are not designed to be energy efficient. Many IoT devices are battery-operated and attempt to minimise energy consumption to maintain a longer endurance. Due to these reasons, most IoT device applications use intermediate protocols that are designed to be energy efficient and better suit for IoT contexts, such as Zigbee, Bluetooth, Bluetooth Low Energy (BLE), Z-Wave, and Thread [25, 26].

2.2 Digital Forensics

2.2.1 Digital Investigation Process

A typical digital forensic investigation starts when a law enforcement encounters an electronic device in a crime scene or seized it from a person under the investigation. These devices can vary from traditional personal computers and mobile devices to IoT devices, such as smart home devices and wearables. The seized devices are usually handed over to a digital forensic laboratory where specialists perform the investigation on the device [4]. Initially, pictures and notes are taken about the physical conditions of the device. For personal computers, the investigation mainly focuses on the data stored in the non-volatile memory, i.e., the hard disk or solid state drive. A forensically-sound disk image is acquired, which is analysed using specialised software tools to identify the pertinent information.

The sole purpose of acquiring a disk image from the device under investigation is to prevent the investigative procedure from inadvertently making changes to the device. Popular tools such as EnCase [27] and The Sleuth Kit [28] are designed to extract information from disk images. In contrast to personal computers, the forensic analysis of mobile devices typically requires specialised hardware tools due to the fact that different makes and models of mobile devices have different internal structures. Even though there are various commercial tools available for mobile devices, they need to be updated each time a new device model comes into the market. The maintainers of commercial tools for forensic evidence acquisition on mobile devices are struggling to keep up with the highly dynamic ecosystem of mobile devices [29].

When the digital evidence is presented to a court of law as a part of an investigation, the evidence acquisition procedure can be thoroughly questioned and challenged. This is due to the fact that legal processes follow strict procedures to ensure the fairness to all parties involved. As a result, digital forensic evidence acquisition procedures are demanded to be documented and auditable. Current digital evidence acquisition procedures, practices and tools in use are time-tested to be resilient against such legal challenges. Therefore, whenever a completely new way of acquiring digital evidence is introduced, it has to be thoroughly scrutinised to face reliability challenges in a court of law.

2.2.2 Forensics of Internet of Things Devices

Traditionally, digital forensic investigators deal with personal computers or mobile devices as the principal digital evidence sources [2]. However, the emergence of the IoT has revolutionised the potential for digital forensics by opening up new sources of evidence [10]. IoT devices are ubiquitous in everyday life and collect a large volume of information that can be useful in a forensic investigation [30]. For example, a fitness wearable can contain highly precise information regarding the movements of the owner, which can assist in identifying where the person was at a particular point in time. Similarly, a smart TV or a smart light bulb may contain information regarding the usage patterns of the owner and might hint at the presence of the owner in a premises at a particular time.

While IoT devices can provide valuable data for digital investigations, acquisition of data from IoT devices is not a straightforward task. An IoT device is a special purpose device designed to perform a specific task. Several manufacturers produce these devices often with bespoke hardware and software designs. As a result, IoT devices lack standard interfaces and forensic acquisition methods. Therefore, IoT-focused digital forensic tools are extremely limited. In fact, many IoT devices are not usable in investigations due to unavailability of support from commercial vendors or open-source projects. The large variety of IoT devices in the market makes it virtually impossible to support all of them within a limited tool set. This can often result in a device

requiring a memory chip-off procedure in order to access its data [13]. However, with the increasing application of lightweight cryptographic algorithms in IoT devices, such a physical access into the device may not be a viable way to acquire forensic data [31]. Furthermore, some IoT devices may not store any data on-board at all. Instead, the data they produce are delivered to an associated smartphone or a cloud server, rendering the direct forensic inspection of such IoT devices a futile exercise.

2.2.3 Encrypted Devices

Due to the increasing concerns regarding security and privacy among communities, modern computer systems are designed and shipped with built-in security mechanisms. Popular smartphones, such as iOS and Android based devices, encrypt their internal storage in order to protect user data from third parties [29]. Each of the mainstream PC operating systems, such as Mac OS, Windows, and Linux, provide built-in hard disk encryption. Meanwhile, network communications, both wired and wireless, employ strong packet encryption mechanisms. Modern computer hardware have made the handling of encrypted data an everyday possibility in consumer, industrial and military applications [32]. Computer devices seized at a crime scene containing encrypted data poses a huge challenge to the investigation. The IoT ecosystem is no exception to this data encryption trend making the challenge of digital forensic investigations on IoT devices even more challenging.

Whenever encryption is involved in the storage of a device being investigated, forensic tools are unable to extract information [10]. From the investigator's perspective, a very limited number of workarounds are potentially viable. The obvious approach can be asking the device owner for the decryption key or password. However, if the device owner is not cooperative, this approach is not viable. Another possible approach can involve seeking the assistance of the device vendor to unlock the access to data using whatever the capabilities the vendor holds. However, many recent cases indicate that even the device vendors does not have access to the encrypted data storage on devices they produce. Under these circumstances, forensic investigations may end up un-

able to collect the required evidence from the devices they have seized [33].

For a more comprehensive background on the domain of digital forensics, please refer to the two textbooks by Eoghan Casey [3] and Xiaodong Lin [34].

2.3 Digital Signal Processing

Digital Signal Processing (DSP) is a broad field that is focused on mathematical methods and algorithms to process signals on computers. A signal is a function that can describe how a particular parameter is related with another parameter in a system [35]. In different words, a signal represents a quantity that varies against another quantity. The former quantity is called *dependent variable*, while the latter quantity is called the *independent variable*. For example, the variation of temperature inside a building over time is a signal, where temperature is the dependent variable while time is the independent variable.

2.3.1 The Nature of a Signal

Signals that have a pattern repeating over a specific time period are called periodic signals. Such periodic signals contain some important properties that help to uniquely distinguish them. Frequency (f) is the number of repeating patterns occur in the signal within a time period of a second, measured in Hertz (Hz). The distance between two consecutive peaks is called wavelength (λ) of the signal, measured in meters (m). For a given time instance, the value of the dependent parameter of the signal is called amplitude (A), which can be measure in decibel (dB). Phase (ϕ) is the current shift of the signal from a reference position. A periodic signal with frequency f produces further signals with positive integer multiples of f, called harmonics. The harmonics of a wave is numbered according to this integer multiple such as 1st harmonic (the fundamental frequency f itself), 2nd harmonic ($2f$), 3rd harmonic ($3f$), etc. [35].

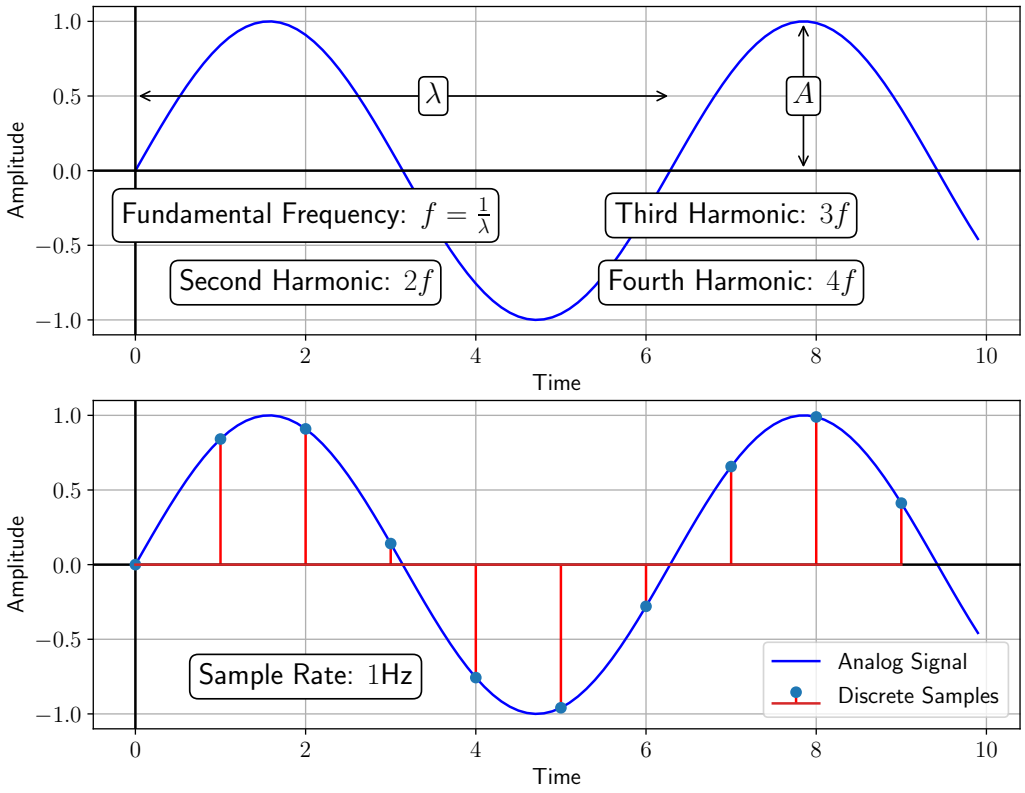

Figure 2.2: Properties of a periodic signal.

2.3.2 Analog and Digital Signals

A signal can be either *continuous* or *discrete*. A continuous signal has continuous values for both its dependent and independent variables. In contrast, a discrete signal has only discrete, i.e., countable, values for its variables. Another categorisation is *analog* and *digital* signals. An analog signal is a continuous signal. Most signals found in the nature are analog signals. Meanwhile, a *digital signal* is a special kind of discrete signal that has discrete values for both dependent and independent variables.

In order to process signals on modern computers, the signals has to be in digitised form. That means, analog signals have to be converted into digital signals before they are fed into computers. The digitisation of an analog signal is called Analog-to-Digital Conversion (ADC), which consists of two steps; *sampling* and *quantisation*. By sampling a signal, the continuous independent

variable of the signal, i.e., time for most signals, is converted into a discrete variable. Sampling is achieved by capturing the value (a sample) of the dependent variable over predefined intervals of the independent variable (see Figure 2.2). The number of samples taken per second is called *sample rate* of the ADC. Meanwhile, the quantisation is achieved by converting a captured continuous sample value into one of the predefined discrete set of values. The number of discrete set of values available for the quantisation process depends on the number of digits used to represent a sample.

Due to the nature of sampling and quantisation, the conversion of an analog signal into a digital signal causes loss of information. Due to the limited number of bits available to represent a sample, the number of potential discrete values available is finite. Therefore, multiple continuous values in the analog signal can be mapped into the same discrete value in the digital signal causing information loss. This is called *quantisation error*. Increasing the size of a sample, i.e., bit length of a sample, can minimise quantisation error. Similarly, due to the discrete time intervals used, poor sample rates can lose information of the analog signal and cause errors, such as aliasing, at later stages. Therefore, sample rate has to be at least as twice as the highest frequency of the analog signal, in order to prevent aliasing errors. This phenomena is often referred as Nyquist sampling theorem [35].

2.3.3 Time and Frequency Domains

A signal can be represented and processed in two major forms, namely the time domain and the frequency domain. When a signal is in the time domain, the independent variable is time while the dependent variable is the magnitude of the signal. When in the frequency domain, the independent variable is frequency components of the signal while the dependent variable is magnitude of each frequency component. A signal can consists of multiple frequency components that we cannot see from the time domain. In order to convert a time domain signal into the frequency domain, we have to decompose and extract individual frequency components of the original signal. This decomposition can be performed through a Fourier transform. The most commonly used al-

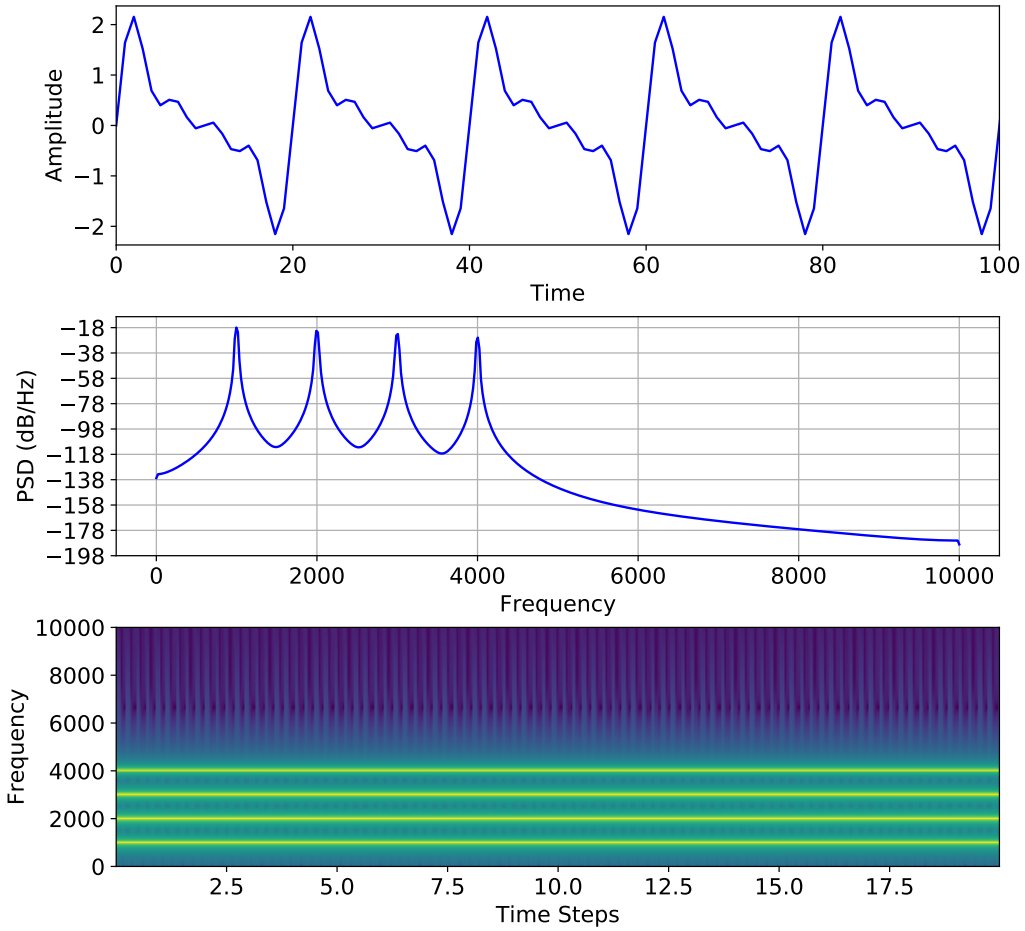

Figure 2.3: Visualisation of a signal as waveform, power spectral density (PSD), and spectrogram.

gorithm for a Fourier transform is the Fast Fourier Transform (FFT) that is fast and efficient compared to others [35, 36]. Furthermore, instead of performing FFT over the entire signal, small consecutive segments of the signal over time axis can be used to perform the Fourier decomposition called Short-term Fourier Transform (STFT). The result of STFT contains frequency information for each segment and time information across segments.

2.3.4 The Visualisation of Signals

Visualisation of signals can be done in various ways depending on the specific information that need to be conveyed. The most simplest form is plotting the waveform of the signal in time domain. In a waveform diagram, the x-axis represents time while the y-axis represents amplitude of the signal. Although the shape of the signal is visible in waveform, the frequency related information are not visible. An FFT plot can be used to illustrate the frequency information of a signal where the x-axis represents frequency while the y-axis represents magnitude of each frequency component of the signal. Power Spectral Density (PSD) plots are another way similar to FFT plots where power spectral density is represented in the y-axis. However, an FFT or PSD plot lacks time information. In order to visualise both time and frequency information, a spectrogram can be used. A spectrogram has time information in one axis while frequency information is depicted in the other axis. The magnitude variation of each frequency component over time is represented by a color coding. Figure 2.3 represents three visualisation of a signal, i.e., waveform, PSD, and spectrogram, that has four frequency components mixed inside namely, 1 kHz, 2 kHz, 3 kHz, and 4 kHz.

2.3.5 Tools and Libraries

When using DSP in applications and research, dedicated tools are necessary. DSP can be performed on either dedicated hardware platforms or on general purpose computers with the help of specialised software. The advancements of Very Large-scale Integration (VLSI) technologies have enabled the possibility of building dedicated computing hardware to perform special tasks, with the help of hardware systems such as Field-programmable Gate Arrays (FPGA) [37]. As a result, special-purpose DSP processors can be built to perform application-specific tasks. Although such hardware-implemented DSP tools are extremely efficient, they are expensive and require specialised technical expertise. In contrast, various software tools and libraries in different programming languages are available to facilitate DSP on general purpose

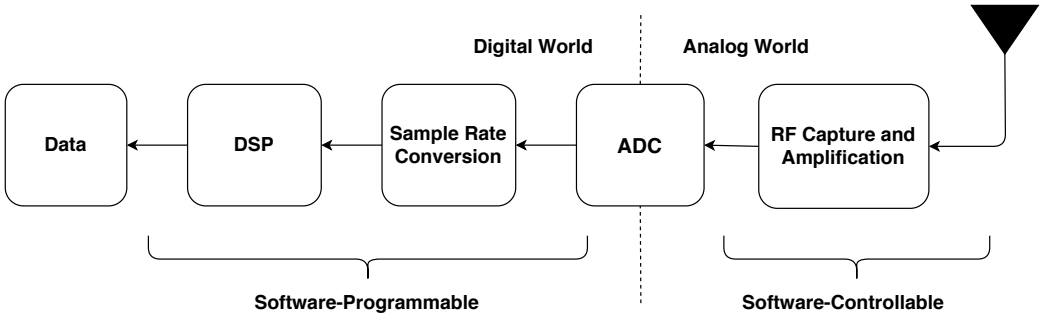

Figure 2.4: The architecture of a software defined radio (SDR) platform.

computers. In Python language, numpy, scipy, and matplotlib libraries collectively provide DSP capabilities [36]. All the experiments presented in this thesis research use the aforementioned Python libraries to process EM data.

2.4 Software-defined Radio

2.4.1 Software-defined Radio Architecture

Radio technology facilitates the transmission and the reception of information using electromagnetic radiation. Traditionally, radio transceivers have analog physical layers handling all types of processing. With the inception of digital data processing capability, modern analog physical layers of radios can be software-controlled. Taking it further, Software-defined Radio (SDR) moves most of analog physical layer functionalities of a radio device into digital domain [38] making them software-programmable. Consequently, SDR platforms can be used to build radio applications that are more flexible and adaptable to changes [39]. Figure 2.4 illustrates the architecture of an SDR platform that operates in between analog and digital worlds. The analog components of an SDR are software-controllable, meaning that certain setting of analog components can be controlled by software, such as the base-band frequency of the receiver and the level of signal amplification.

Figure 2.5: RTL-SDR and HackRF One devices with their default antennas.

2.4.2 Software-defined Radio Hardware Tools

The first component of an SDR platform that faces the analog world is its hardware. As described in Subsection 2.4.1, the hardware side of SDR is bare minimum and provide only the EM signal capture, amplification, and ADC functionalities. Various SDR hardware devices are available in the market that differ from each other due to their supported frequency range, sample rate, and amplification capability.

RTL-SDR is a repurposed digital TV and radio receiver that has been shown to be usable as an SDR hardware [40]. Costing less than €15, it is the least expensive SDR device. Different models of RTL-SDR devices consist of different frequency ranges and sample rates. Generally, their frequency range is about 22 MHz to 1 GHz. Similarly, sample rate of an RTL-SDR device is somewhere close to 3.2 MHz. HackRF One is a much more powerful SDR device [41]. It has a operating frequency range of 1 MHz to 6 GHz, enabling it to be used in a wide variety of radio applications. It has a maximum sample rate of 20 MHz. In contrast to RTL-SDR, HackRF One is a half-duplex device and therefore, it can be used for both reception as well as transmission of radio signals. It supports a wide variety of antenna types depending on the application. Both RTL-SDR and HackRF One devices are USB-powered and

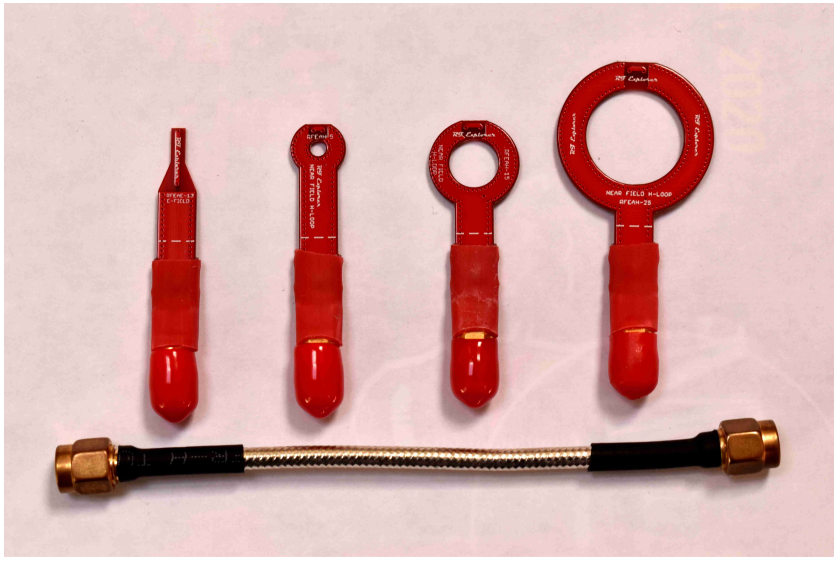

Figure 2.6: Near field antennas of varying diameters and lengths along with a semi-rigid RF cable.

able to deliver their data to software running on a computer via the same USB port. Therefore, it is convenient to use them on general purpose computers. Figure 2.5 depicts the devices with their default antennas.

Depending on the nature of the EM signal, the type of the antenna that should be used to capture signals differs [42]. H-loops are a type of antennas that is useful to capture weak EM radiation from small regions of electronic circuits. They are designed to behave as an inductive coil capturing the EM field in the vicinity. The diameter of an H-loop antenna decides its spatial resolution and the sensitivity. Throughout the experiments of this thesis, a near-field H-loop antenna kit from RF Explorer was used (see Figure 2.6) [43]. The antenna kit consists of SubMiniature version A (SMA) female connectors making them compatible with the HackRF One SDR device. Furthermore, the characterised frequency range of the antenna kit is between 1 MHz to 7 GHz, which sufficiently covers the EM side-channel signal frequencies. When placing the antenna in close proximity to a specific region of an electronic circuitry, a semi-rigid RF cable can be used in between the antenna and the SDR device to hold the setup without human intervention.

While the near-field H-loop antennas can help to localise a signal being

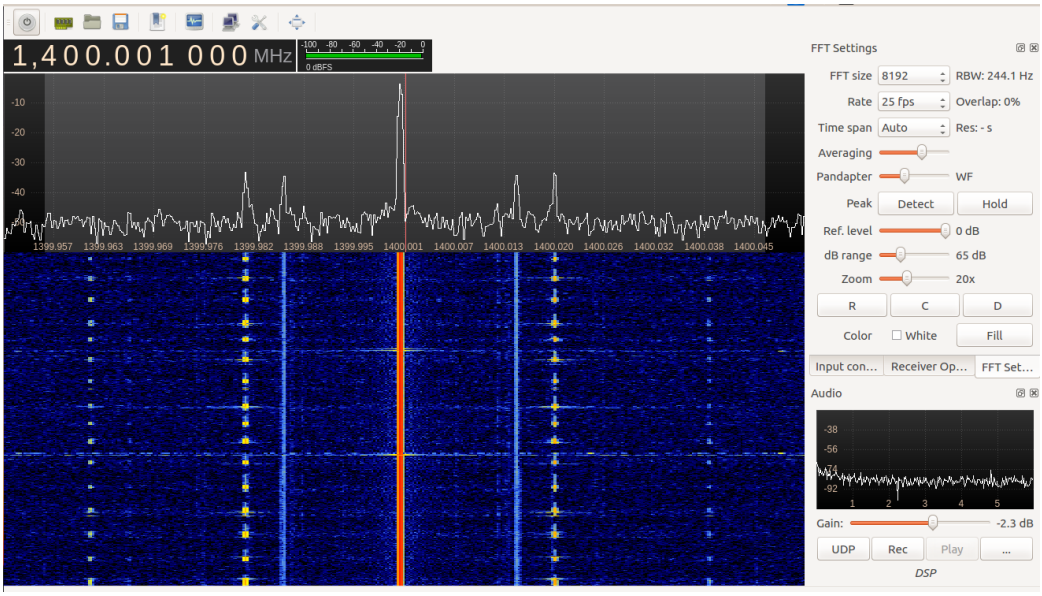

Figure 2.7: GQRX SDR software capturing and visualising EM data from a connected HackRF One device.

emitted from a specific location of an electronic circuit, it is possible to use directional antennas and signal amplifiers to observe EM radiation from electronic circuits at longer distances up to several meters [44].

2.4.3 Software-defined Radio Software Tools

The EM data samples captured by SDR hardware need to be processed in order use them in applications. DSP software libraries that were discussed previously can be used for this purpose. However, there are dedicated software tools and libraries that facilitate the configuration and proper control of SDR hardware. In order to capture data through SDR hardware and perform the basic signal demodulation and visualisation, tools such as GQRX SDR [45] and SDR# (SDR Sharp) [46] can be used (see Figure 2.7). For better control over SDR hardware and data produced by them, much advanced software libraries are necessary. GNU Radio is an open-source software library that supports a wide variety of SDR hardware and provide a rich set of EM data processing functions [47]. It is possible to build SDR-based radio applications entirely on software by programming in Python by using GNU Radio libraries for Python.

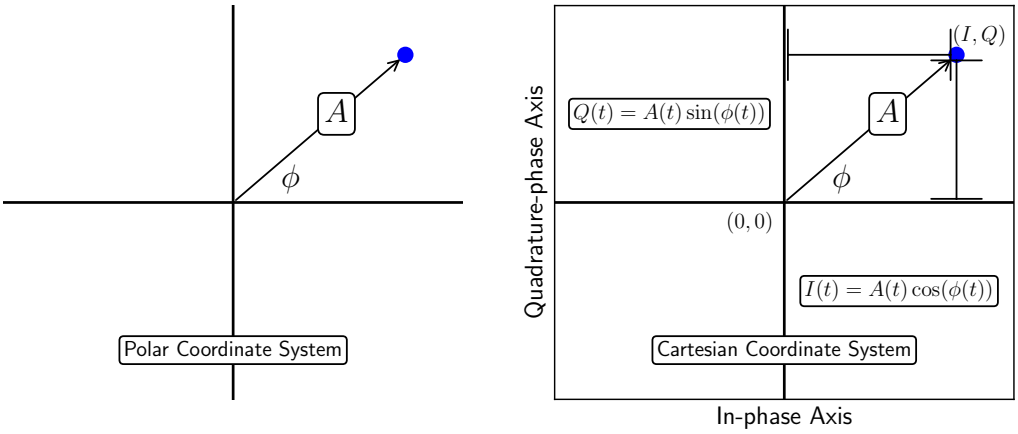

Figure 2.8: Representation of an SDR data sample in polar and Cartesian coordinate systems.

In addition to that, GNU Radio library provides a GUI tool called GNU Radio Companion (GRC) that allows creating SDR applications as Python programs through a drag-and-drop interface [48].

2.4.4 The Nature of Software-defined Radio Data

Due to different bit lengths allocated and the encoding used, the data samples produced by each SDR hardware device can differ from each other. However, when using GNU Radio library to handle the SDR device, the EM data samples that come through the library have a consistent format. The resulting samples of GNU Radio library are in In-phase/Quadrature-phase (I/Q) data format, which can be described further as follows.

The Equation 2.1 represents a sine wave where A is amplitude, f is frequency, t is time, and ϕ is phase.

$$A\cos(2\pi ft + \phi) \tag{2.1}$$

According to the equation, the amplitude, frequency, and phase collectively can represent a signal. However, when a spontaneous state of a signal needs to be represented, the amplitude (A) and phase (ϕ) alone are sufficient, if they are placed in a polar coordinate system. Such a representation of a

specific point of a signal with A and ϕ in a polar coordinate system can be converted into a Cartesian coordinate system where the same information are represented by using the coordinates of the two axes. The horizontal axis of this new Cartesian coordinate system is labeled as the *In-phase* (I) axis while the vertical axis is labeled as the *Quadrature-phase* (Q) axis (see Figure 2.8). This means, a signal sample captured by the SDR can be represented using two coordinates (I, Q) and they are usually stored and processed as a complex number. The real component of the complex number represents the I coordinate, while the imaginary component of the complex number represents the Q coordinate [49].

In GNU Radio library, two 32 bit (4 byte) floating point values are used to represent a complex I/Q sample. Therefore, each EM data sample is a 8 bytes long complex value. When capturing EM data using an SDR hardware and GNU Radio library, these I/Q samples are produced in a continuous stream with a rate equal to the sample rate of the SDR hardware.

Due to the Nyquist sampling theorem, the sample rate has to be twice as large as the frequency of the interested signal in order to prevent aliasing. This can pose a challenge to signal acquisition devices when attempting to capture a high-frequency signal. However, this limitation negatively affects only to real-valued sampling. The use of I/Q sampling helps SDR devices to overcome this challenge. Firstly, they shift the centre of frequency band of interest to baseband, i.e., 0 Hz, at the analog hardware front-end using a local oscillator. Secondly, they produce complex I/Q samples. By doing so, SDRs can use a sample rate that is lower than the signal frequency it is capturing [50]. Furthermore, the sample rate of SDR devices are equal to the signal bandwidth they capture.

2.5 Electromagnetic Side-Channel Radiation

2.5.1 Sources of Electromagnetic Side-Channels

EM waves can be generated from electrical and electronic systems without the intention of the designers when conductors in a circuit accidentally behave as

antennas [51, 52]. Electronic circuits that perform high-speed switching operations are especially susceptible to generating unintentional EM noise due to their higher frequencies. Among them, digital electronic components used on computers are well known sources of EM noise since they employ high-speed clocks to carry out their internal operations [53]. CPU, memory chips, data and address bus lines, various ports such as USB and Ethernet are examples of EM radiation sources on a typical computer system. Among these, EM radiation from the CPU is well-known to leak information about the internal activities of the CPU including data being handled.

Unlike the CPU units used on PCs, the MCU units used on IoT devices are designed to consume less power and hence, they cause weaker EM radiation compared to their PC counterparts [53]. While any IoT device can be designed with a unique MCU, there is an important commonality of components. There are only few common architectures used for MCU in most IoT devices, e.g., ARM, AVR, and MSP430. This means, EM radiation patterns identified from a particular processor chip should be applicable across many IoT devices that employ them.

2.5.2 Observation of Electromagnetic Side-Channels

Observation of unintentional EM radiation from computing devices can be made using traditional signal analysis hardware, such as oscilloscopes and spectrum analysers. An alternative is SDR, where fast ADCs are used to digitise EM signals and feed into software for processing and visualisation. Compared to the traditional options, SDRs provide more flexibility and ease of use to non-signal analysis professionals. In order to observe EM radiation from a target device's CPU, the EM radiation frequency needs to be determined. The clock frequency of the CPU is the most fundamental frequency for EM radiation. Furthermore, the harmonics of this fundamental frequency can also contain the desired information. Therefore, the exact choice of the frequency depends on what has the highest amplitude with the least amount of external interference.

A simple hardware set-up can be used to demonstrate the unintentional

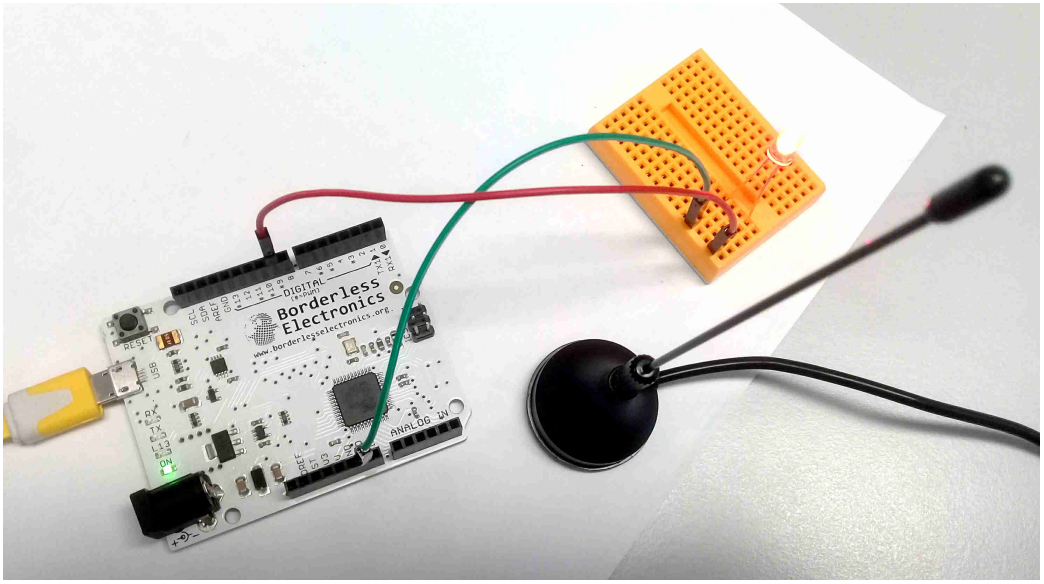

Figure 2.9: Hardware setup to observe EM radiation from an Arduino device blinking an LED.

EM signals generated by an IoT device observed using an SDR platform. An Arduino prototyping board is loaded with a simple program to blink an LED connected to it via its general purpose I/O pins. An antenna connected to an RTL-SDR dongle (an SDR with low sample rate) is placed close to the Arduino board in order to receive unintentional EM signals emitted from the board (see Figure 2.9). The Arduino board consists of an MCU that operates at 16 MHz. However, the RTL-SDR dongle cannot be tuned to frequencies below 22 MHz. Therefore it was tuned to the second harmonic of the Arduino's clock speed, i.e., 32 MHz. Figure 2.10 illustrates the spectrograms of three different EM signal observations. When two different LED blinking patterns are performed by two different programs separately, the Arduino emitted two completely different EM signal patterns as can be seen from Figure 2.10 (a) and Figure 2.10 (c). The transition of Arduino device from running first program to the second is depicted in Figure 2.10 (b).

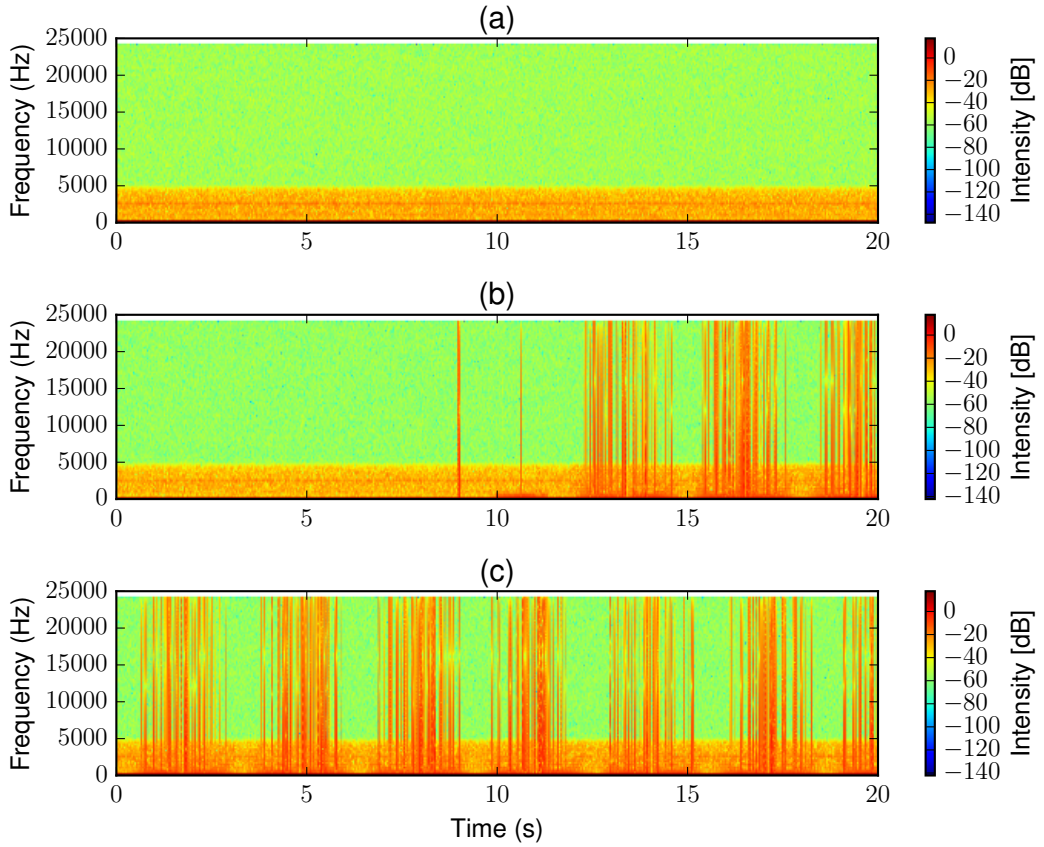

Figure 2.10: Spectrograms of AM demodulated EM radiation acquired from an Arduino device. In (a) and (c), the device is running two unique programs, while (b) depicts the transition period.

2.5.3 Leakage of Critical Information

When performing an EM-SCA attack, the information about internal operations of the device being attacked are modulated into the radiation signal in various ways. This exact method of leaking data through an EM side-channel is called the *leakage model*. It is necessary to assume a specific leakage model when performing an EM-SCA. From the inception of side-channel cryptographic key recovery attacks, the major leakage model that has been explored is *Hamming weight* leakage model. In multiple publications by Kocher et al., it has been shown that the Hamming weight of the data being handled by a CPU gets modulated into the side-channel – hence the name of the leakage model [54, 55]. Another closely associated model that often gets considered is the *Hamming*

distance leakage model where the number of bits that gets flipped is assumed to be modulated into the EM radiation [56]. Further improvements have even lead to the modelling of the exact bit transitions, which can be either 0 $\rightarrow$ 1 or 1 $\rightarrow$ 0, called *switching distance* leakage model [17].

$$A_{leak\ t} = \alpha HW(P_t \oplus K_t) + \eta_t \tag{2.2}$$

$$F_{observe} = F_{clock} \pm F_{leak} \tag{2.3}$$

When attacking a cryptographic algorithm with the intention of retrieving the encryption key under Hamming weight model, it is assumed that in a specific point in the execution of the algorithm, the Hamming weight of the cryptographic key bits are exposed [57]. For example, consider a simple cipher where a plaintext, P, is associated with a key, K, through `XOR` operations to generate the ciphertext. The amplitude of the information leaking EM signal $A_{leak\ t}$ can be modelled with the Hamming weight leakage model as shown the Equation (2.2). HW is the Hamming weight function while P_t and K_t are the plaintext and key bytes `XOR`-ed at the time instance t. α is a positive integer used as a scaling factor while η_t is the noise at time t. When a signal gets modulated with a carrier wave, such as CPU clock or on-board radio transmitter signal through the amplitude, it causes side-bands to occur between the carrier wave [58]. For example, consider the clock frequency of a CPU to be F_{clock} and the frequency of the leakage signal to be F_{leak}. This causes the observation of a frequency bandwidth $F_{observe}$ that spans from $(F_{clock} - F_{leak})$ to $(F_{clock} + F_{leak})$ as illustrated in the Equation (2.3).

2.6 Machine Learning

The domain of Machine Learning (ML) deals with building computer algorithms that can learn from data without a need for explicit programming [59]. ML algorithms can be basically categorised into two principal classes as supervised and unsupervised learning algorithms. In supervised learning, the algorithm is given a sufficiently large set of labeled data to learn. Once learned, the

algorithm can be exposed to new data where it can figure out the appropriate labels based on past experiences. In unsupervised learning, the algorithm is exposed to data without prior labelling and it figures out relations between different aspects of input data on its own. Meanwhile, problems that can be solved using ML can again be categorised into two classes as regression and classification problems. In regression problems, the objective is to predict a value of a parameter based on input data. As the name implies, classification problems deal with classifying data into predefined or self-learned group of classes.

The choice of the ML algorithm to use when trying to solve a problem depends on a multitude of factors such as the class of the problem, the amount of data available, and the nature of those data. Linear regression, support vector machines (SVM), decision trees, and neural networks are to name a few of ML algorithms in use [60]. Regardless of the algorithm, the application of ML algorithms on a problem typically follows a similar process. The first step of applying ML in a problem is the careful study of the problem and the availability of data. Then, an ML model is trained using a training dataset. A trained ML model is evaluated using further testing and validating data in order to identify whether it sufficiently represents the real world. If the evaluation process turns out that the model is not accurate enough, the reasons need to be identified to start model training process once again from the beginning.

When training and testing ML models to solve real-world problems, ML algorithms need to be implemented as computer software. In order to ease the life of researchers and developers, a wide variety of software libraries have been developed that implement almost all the published and recognised ML algorithms. Some of the most popular ML software libraries include Scikit-learn [61], PyTorch [62], and TensorFlow [63] – all of them are supported in Python language. In the research presented in this work Python-based libraries are preferred due to the ease of integrating those trained models with our larger software framework, which is implemented on Python.

A more detailed account of the topic is given by Christopher M. Bishop [60] and Aurélien Géron [59].

Chapter 3

Related Work

This chapter introduces the related work of this research topic covering a wide body of literature. The organisation of this chapter is as follows. Section 3.1 discusses side-channel attacks in general. Section 3.2 introduces how EM radiation are generated from computing devices and how they can be capture for analysis. Such EM radiation captured from computing devices can be used for two different purposes from the perspective of information security and forensics. The first is the possibility of using EM radiation as a signature to uniquely identify hardware and software. This topic is discussed in Section 3.3. The second is the possibility of exfiltrating information through the EM side-channel, covered in Section 3.4. Section 3.5 explores the literature on the standards related to EM side-channel attacks. Furthermore, it introduces some of the tools available for EM-SCA attacks. Finally, Section 3.6 introduces the recent advancements in the field that may shape the future direction of EM-SCA.

3.1 Side-Channel Attacks

The domain of side-channel attacks spans over a wide variety of techniques. Each side-channel attack on a computer system focuses on a specific unintentional information leakage through either hardware or software. The amount of memory and cache spaces shared between different software, the time a

program takes to respond to different inputs, the sound different components of computer hardware make, the amount of electric current a computer system draws, and the EM radiation a computer hardware emits are examples of such side-channels. This section provides an overview of side-channel attacks in general.

Computer programs contain conditional branches and loops in order to handle input and produce the intended output. Depending on the input values, the execution path of a program can vary, which may result in a different program execution time. It has been shown that the execution time of encryption algorithms can reveal information regarding the input values provided to it, which includes cryptographic keys [64]. For example, the square and multiplication segment in RSA algorithm checks whether a key bit is `0` or `1` before moving into multiplication operations. Therefore, observation of large number of execution times with the same key and different input data can lead to uncovering cryptographic key bits effectively [64, 65, 66].

In environments where multiple Virtual Machines (VM) run on the same hardware, e.g., cloud infrastructure, cache-based side-channel attacks are possible [67]. While each VM has its own virtual resources, many of them are mapped into shared physical resources including shared cache memories. It has been shown that an attacker running a VM on a virtualised environment can spy on a victim VM through the shared cache storage, which can lead to the extraction of sensitive information such as cryptographic keys [68].

It has been shown that acoustic emanations from various components and peripherals of computer systems can be used to exfiltrate information [69]. Genkin et al. showed that it is possible to distinguish between CPU operations by listening to acoustic emanations resulting in an attack on the cryptographic keys of the RSA algorithm [70].

Computer displays and their video cables have been identified to leak information about the content being displayed on the screen through EM radiation. Such leakages from Cathode-ray Tube (CRT) displays have been known for several decades [71, 72]. Video information provided to a computer display has synchronisation information to recognise between different lines of pixels and different frames, which are called horizontal and vertical synchroni-

sations. By recognising this synchronisation information in the EM radiation, an attacker can reconstruct the images being displayed [73, 74].

In [54, 55], Kocher et al. were the first to introduce power consumption-based side-channel attack: Simple Power Analysis (SPA) and Differential Power Analaysis (DPA). In SPA, power consumption variation over time is sampled for a target computing device while it is performing a particular activity. The waveform of the power consumption data, when plotted against time, contains patterns that correspond to the instructions executed on the target device. If SPA can reveal the sequence of instruction operations of a cryptographic algorithm, it follows that the sequence depends on the data being handled by the algorithm (due to conditional branching). Designing code to minimise data dependent branching, which does not show characteristic power consumption patterns for specific operations, can prevent attackers from recognising what is being executed on the device [75].

DPA is a technique that can be custom-tailored for specific encryption algorithms. Kocher et al. used the DPA technique against the Data Encryption Standard (DES) algorithm [54]. The technique was able to guess the encryption key accurately, given sufficient cipher texts and the power traces corresponding to those encryption operations. Kocher et al. claim that they have used DPA to reverse engineer various unknown algorithms and protocols on devices. Furthermore, they state that it may be possible to automate this reverse engineering process and also use the same techniques with EM radiation in addition to power consumption.

While various side-channel attacks are possible on computer systems, it is possible to increase the advantages achievable by combining multiple side-channels that leak different kinds of information together into a single attack [76]. For example, the power analysis and EM analysis can be performed together in order to reduce the errors and improve the accuracy of inferring the leaked information from a computer system.

3.2 Unintentional Electromagnetic Radiation

EM radiation is the underlying technology for numerous of wireless communication. Meanwhile, it is a well known fact that electronic devices generate EM radiation on unintended frequencies as a side effect of their internal operations [53]. Such unintended EM radiation are regulated by government agencies, such as Federal Communications Commission (FCC) in the USA, due to the possible interference they can make on legitimate wireless communication and the potential health issues they can cause to the users of these devices. However, it is not possible to entirely avoid such radiation and the equipment manufacturers attempt to minimise it as much as possible [77]. This section discusses how EM signals are generated from different components of a computer system, what kind of information they can carry, and what types of methods and tools can be used to capture these signals.

3.2.1 Hardware that Causes Electromagnetic Radiation

As derived from Maxwell's equations [51], EM waves can be generated by electric currents varying over time. Characteristics of the EM waves being generated, such as frequency, amplitude, and phase, depend on the nature of the time varying electric current. Based on this principle, modern communication systems generate oscillating currents on antennas that generate EM waves that propagate over free space to be captured by another antenna with appropriate properties. The fact that modern computer systems have a large number of components that depend on electric pulses or alternating currents for their operations, leave the space for EM waves to be generated at unexpected frequencies without the intention of the system manufacturer.

There are multiple computer components that operate in a coordinated, sequential fashion according to clock signals. Among them, both the CPU and the RAM are important. The CPU performs a cycle of fetching, decoding and executing instructions while the RAM maintains data and instructions when the device is powered on. The EM radiation from these components carry a significant amount of side-channel information regarding the events related to

software execution and data handling. On most IoT devices, the CPU and the RAM are included in the MCU making them the most important EM source.

3.2.2 Sampling Electromagnetic Radiation

The EM radiation frequencies of a device is unpredictable due to its dependability on various hardware characteristics. Therefore, it is difficult to have a universal purpose device, which can be used to observe EM radiation from a computer and interpret side-channel information. It has been shown that small magnetic H-Loop antennas can be used for this purpose of picking EM radiation from computing devices [17]. When EM signals are picked up by an H-Loop antenna, it requires digital sampling before the data can be used for analysis. Theoretically, the sample rate of the equipment should be twice that of the maximum EM frequency required to be captured – referred to as Nyquist frequency [35]. For this reason, the EM signal sampling equipment must have a very high sample rate. The most commonly used equipment, with high sample rates to capture EM signals, are oscilloscopes and spectrum analysers. The digitised data these devices capture can be later analysed in signal analysis software. However, access to such devices for information security professionals is not very common [78].

SDR are getting increasingly popular among wireless hackers, hobbyists, and security enthusiasts who are interested in accessing the RF spectrum. An SDR consists of a minimal hardware component that can be tuned to a range of RF frequencies and digitise signals with a fast ADC. These digitised samples are processed entirely on software [39]. A wide variety of SDR hardware and software platforms are available [40, 41, 79, 47]. Due to the great flexibility provided by software, SDR platforms have become a perfect candidate for research in EM-SCA. An SDR can be used to scan through a wide range of frequencies to locate potential EM radiation from a computer system.

3.2.3 The Connection between Instructions and Electromagnetic Radiation

As a result of executing instructions in different combinations by the CPU, EM signal patterns are emitted at various frequencies and amplitudes. Depending on the sequence of instructions, i.e., the exact program being executed, the output of EM noise from the CPU varies significantly. Due to this, systematically modelling and predicting possible EM signal characteristics of a computer processor is a difficult task. In order to identify unintentional EM radiation of a computer processor, the most practical method is scanning a large frequency spectrum for suspected EM signals and subsequently trying to interpret these identified signals for potential side-channel information. This arduous approach is a time consuming task that requires manual inspection by a human user.

Callan et al. introduced a metric, called Signal AVailability for an ATtacker (SAVAT), that measures the power of emitted EM signal when a CPU is executing a specific pair of instructions (A and B). The authors show that different selections of A and B instruction pairs emit different SAVAT values, i.e., signal power [80, 81]. An improvement to the SAVAT technique is a method called Finding Amplitude-modulated Side-channel Emanations (FASE). The key idea behind the FASE technique is as follows. When a program activity is alternating at a frequency (f_{alt}) that affects any periodic EM signal originating from any source at a frequency f_c, it is possible to observe two side-band signals at $f_c - f_{alt}$ and $f_c + f_{alt}$ between the f_c signal. Further improvements to SAVAT technique enabled the possibility of identifying both amplitude and frequency modulated EM radiation from CPUs [82, 83, 84]. While it is evident from the existing studies that the EM side-channel leakage is available across various type of CPUs, further studies are necessary to identify the effect of different CPU architectures to the EM radiation.

3.3 Electromagnetic Radiation as a Signature

Due to the manner the EM radiation are generated by the CPU of a device, these radiation patterns correlate with the specific hardware and software settings of the source device. This section discusses the use of this correlation as a signature for the hardware and software of computers.

3.3.1 Electromagnetic Radiation as a Hardware Signature

Despite of the software components available on a computing device, it is important to investigate whether the hardware alone can provide a recognisable EM radiation pattern. Such a capability can lead to profiling of hardware devices and components in order to uniquely identify them purely based on their EM radiation. It has been shown that the simple EM signal acquisition device, the RTL-SDR, can be used to uniquely profile computing devices. Laput et al. used a similar device to acquire EM signals, which were successfully applied to an SVM classifier to uniquely distinguish the EM source device [85]. This possibility has lead to the idea that EM radiation from an electronic device owned by a person can be used as an authentication token of the person instead of relying on conventional methods such as Radio Frequency Identification (RFID) tags [86, 87].

This uniquely distinguishable EM radiation patterns of a known electronic device can help to identify any potential alteration that may have applied to it. For example, when a known electronic device is altered at the hardware fabrication level for a malicious purpose such as accessing stored data or eavesdropping on user's activities, the hardware modification results in a changed EM radiation pattern. Such changes in radiation pattern can be used to uniquely identify the device [88]. Similarly, a genuine electronic device can be replaced with a counterfeit with a malicious objective. It has been shown that even when the counterfeit hardware attempts to follow the design of the genuine device, it still creates distinguishably different EM radiation patterns compared to their original product [89].

3.3.2 Electromagnetic Radiation as a Software Signature

When software runs on different computing devices, it is clear that the hardware EM radiation are influenced by the software instructions being executed. It is important to consider this influence from two different aspects. The first is how uniquely the EM radiation of different software running on the same hardware platform can be recognised. This can be used to pin point to the exact software running on a device. The second aspect is how unique the same software program is when it is running across various hardware platforms. It facilitates the unique detection of a specific piece of software.

When software systems are being developed, requirements arise to debug their behaviour or find ways to increase the performance. Instrumenting the software by applying logging events and break points are the most common ways to identify where complex software is not performing as expected. These instrumentation affect the performance of the software being inspected in addition to the overhead of their placement each time a copy of the software needs to be inspected. It has been shown that unintended EM radiation of the CPU can be used to inspect software execution sequences without having to instrument the software each time it is required to be inspected [90, 91, 92, 93].

The capability to detect a software code execution sequence has opened up the opportunity to identify when a computing device is running a software code not intended by the manufacturer or the owner due to various reasons. One possible scenario can be the software bugs or hardware faults that cause an IoT device to execute unexpected instruction sequences. Another possible scenario can occur when an IoT device is under an attack causing it to run malware or an unintended part of the device's genuine software. Both Stone et al. [94, 95] and Nazari et al. [96] showed that such abnormal deviations of the software code executions on computing devices can be detected using the EM radiation patterns.

The simplest form of representing EM side-channel radiation data is the waveform of the signal in the time domain. Stone et al. built *matched-filter classifiers* that utilise the correlation between known EM signal waveform vectors with unknown EM signal waveform vectors to detect software activities on

MCUs used in embedded devices [94]. In this work, the software activities considered were individual CPU operations such as `mov`, `add` and `sub` instructions. In order to generate matched-filter templates, an assembly program that executed a particular CPU operation continuously was used to trigger a GPIO pin value. This GPIO trigger was separately probed in order to identify the boundaries of EM radiation signals in order to extract the matched-filter template trace. Stone et al. continued to demonstrate that instead of using the time-domain signal as a feature vector, it is more effective to use *Hilbert transformation* of the EM radiation signals [95]. The advantage of this approach is that, when calculating the correlation of two signals, Hilbert-transformed vectors perform better than time-domain vectors for the same Signal-to-Noise Ratio (SNR) of signals. As in their previous work with time-domain signals, templates were generated for individual instructions running on a target device and these were used with a correlation-based classifier to detect when a device was executing anomalous codes.

When using machine learning algorithms to detect patterns in time series data, such as EM radiation, Recurrent Neural Networks (RNN) can play a major role. An RNN is a specific type of neural network where the parameters generated during one-time instance are reused as input to the network again in consecutive time instance [97]. It enables the learning of patterns that occur in data sequences along the time domain. Wang et al. evaluated the effectiveness of Long Short-term Memory (LSTM), a variant of RNN, against traditional Multi-layer Perceptron (MLP) neural networks in classifying EM radiation signals from various embedded devices [98]. They show that both MLP and LSTM networks perform well in detecting the behaviour of the firmware running on their target device. Meanwhile, Han et al. demonstrated the potential of using LSTM networks in identifying control flow of Programmable Logic Controllers (PLC) in industrial environments [92]. Their work indicates that sliding window sampling of EM signals can be effectively used to track the control flow of a program with sufficient resolution to identify malfunctions.

In addition to the time-domain signals and Hilbert-transformed signals, another alternative format of representing EM radiation signals is using a Radio Frequency Distinct Native Attributes (RF-DNA) fingerprint. RF-DNA fin-

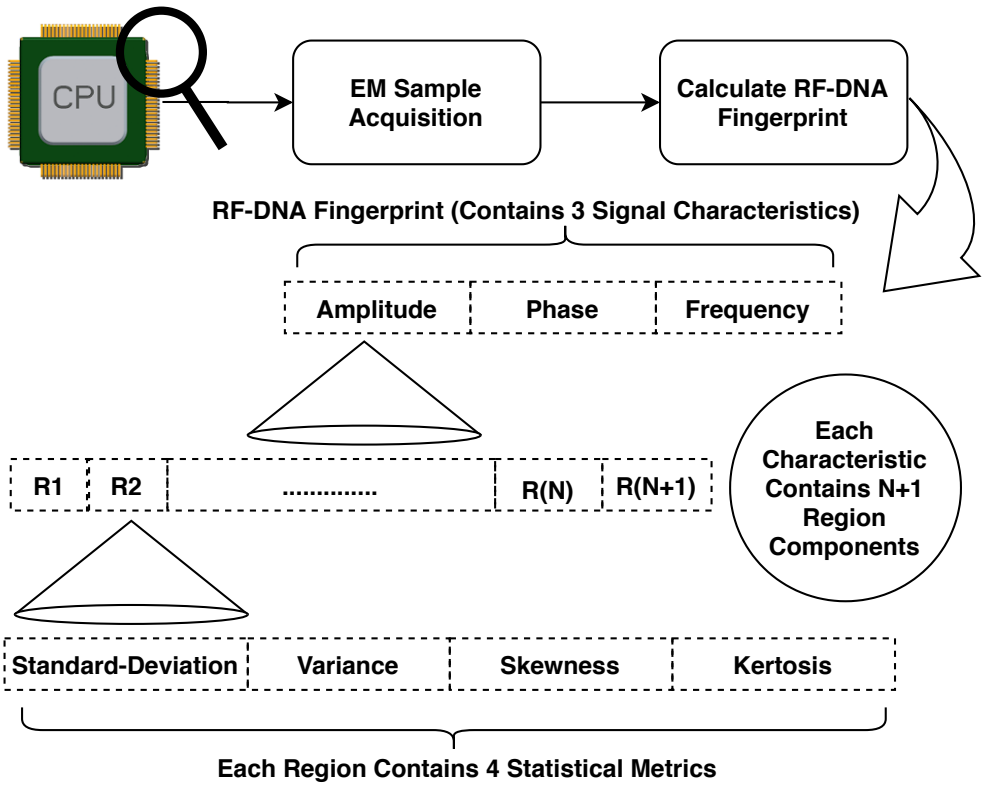

Figure 3.1: The RF-DNA fingerprinting process.

gerprinting is a technique to fingerprint the radio signals transmitted by various device families including WiFi, Bluetooth, Zigbee, GSM, RADAR antennas, etc. This technique has been used to identify rogue devices in a deployment through using their RF signals without physically inspecting them [99, 100, 101, 102]. Deppensmith et al. showed that the RF-DNA technique can be reliably applied to unintentional EM radiation fingerprinting on computing devices [103]. Lukacs et al. used Multiple Discriminant Analysis (MDL) in order to reduce the dimensionality of RF-DNA fingerprints before applying them into a Maximum Likelihood (ML) classifier to identify known radio transmitters used in radar systems [102]. Similarly, Bihl et al. showed that MDL can help in identifying most important features from RF-DNA fingerprints [104]. However, the evaluations performed by Stone et al. on MCU-based IoT devices indicates that further study is necessary to conclude the most reliable format to represent unintentional EM signals [105].

Figure 3.1 illustrates the structure of an RF-DNA fingerprint. When calculating it, the EM signals emitted from a device on a selected frequency is captured, filtered, and amplified appropriately to produce a clean trace. From this acquired time-domain EM trace, the three signal characteristics; *Amplitude*, *Phase* and *Frequency*, are separately considered for further processing. Each signal characteristic is broken into N equally sized regions and then for each region, four statistical metrics; *standard deviation*, *variance*, *skewness*, and *kurtosis* are calculated. Furthermore, the signal itself is again considered as a one entire region, i.e., the $(N+1)^{\text{th}}$ region, to calculate the statistical metrics. As Figure 3.1 illustrates, each of these calculated statistical metrics are arranged in a single vector that becomes the RF-DNA fingerprint of the originally acquired EM trace from a device.

3.4 Electromagnetic Radiation that Leak Information

This section dives into the question of what information is contained in an EM radiation trace of a particular computing system. From a digital forensic perspective, the kind of software running on IoT devices and the data being handled by each software application are potentially of significant interest. If the EM-SCA cannot reveal all of data being handled by an IoT device platform, extracting critical information, e.g., cryptographic keys, can progress the forensic analysis.

3.4.1 Observable Electromagnetic Spectrum Patterns

While there exists a wide variety of MCUs used on IoT devices, Sohaib et al. have shown that it is still viable to perform EM side-channel attacks on them [106]. When considering information leakage from an EM radiation trace, it has been shown that the simple visual observation of a trace can reveal a significant amount of information for side-channel analysis. Visually inspecting the time-domain signal in waveform is the first observational technique. This

approach is called Simple Electromagnetic Analysis (SEMA), which evolved from the SPA introduced by Kocher et al. [54]. The second way of performing visual observations is by transforming the EM trace into the frequency domain and plotting it as a spectrogram. This enables observation of different signal patterns distributed over multiple frequencies.

Multiple publications have demonstrated the effectiveness of SEMA approach in extracting critical data from computers such as cryptographic keys from the EM radiation. The *El Gamal* and RSA algorithms implemented using GnuPG library were attacked by observing critical CPU operations [107]. Furthermore, Elliptic Curve-based Cryptography (ECC) algorithms such as Elliptic Curve-based Diffie Hellman (ECDH) and Elliptic Curve-based Digital Signature Algorithm (ECDSA) are identified to be vulnerable to EM side-channel attacks with SEMA approach [108, 109, 110]. Due to the low computational overhead in ECC algorithms, most mobile devices and IoT platforms employ ECC algorithms to secure data. This indicates that such devices can be inspected through EM side-channels to access cryptographically protected data.

3.4.2 Differential Electromagnetic Analysis

Differential Electromagnetic Analysis (DEMA) – a variant of DPA – uses the amplitude variation of EM radiation of a CPU to discover variables used in an executing program, such as encryption algorithms [54, 55]. When a bit in a CPU register is flipped from `0` to `1` or vice versa, it consumes an amount of energy, which is reflected in the corresponding EM radiation amplitude. Some CPUs may emit a higher EM signal when switching a register bit from `0` to `1` than vice versa since that operation can lead to a higher energy consumption [17]. Due to this, when the content of a complete CPU register is modified, it is possible to identify the *hamming distance* between the previous and new state of the register using the resulting EM radiation. Since every instruction running on a CPU affects the values on different registers, this means that attackers can identify instructions being executed and intermediate variables used based on the EM observation.

A variant of DEMA is Correlation Electromagnetic Analysis (CEMA) [56,

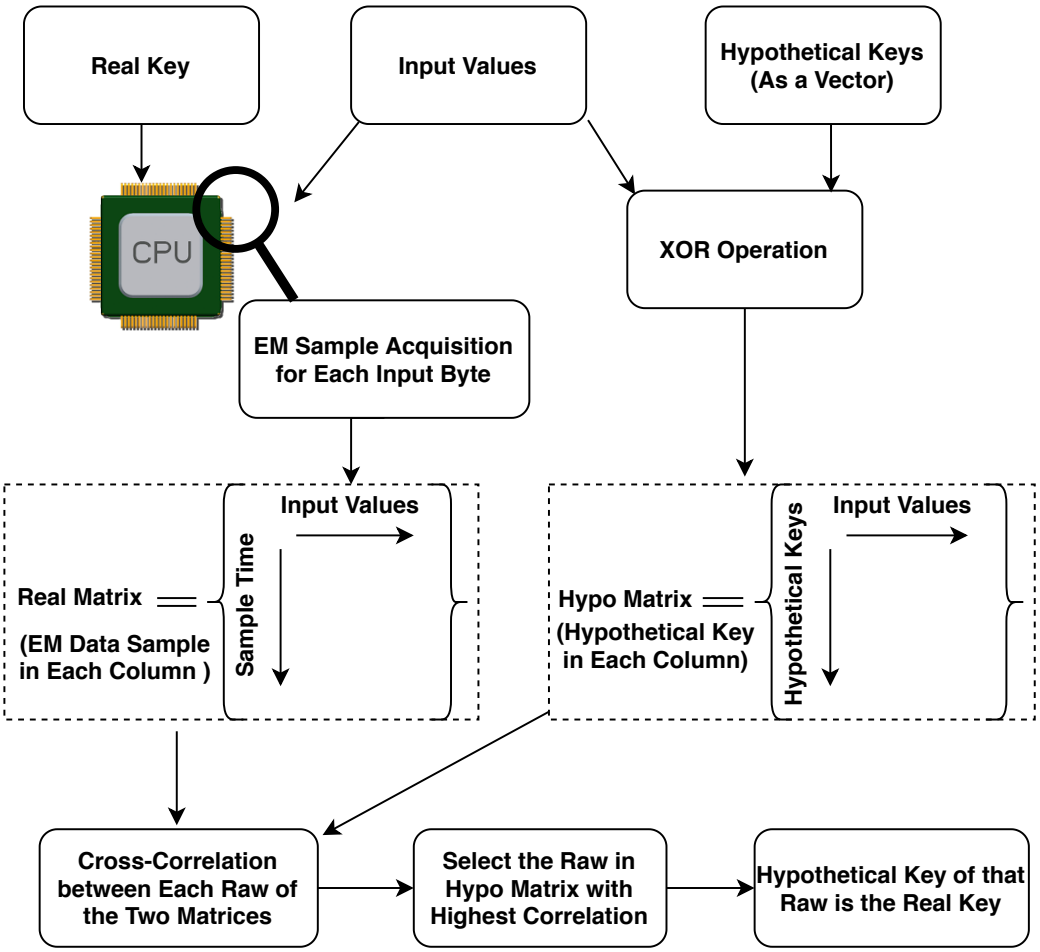

Figure 3.2: EM analysis of XOR-Cipher algorithm to extract the encryption key.

57]. Figure 3.2 illustrates how a CEMA attack is used to identify the key of a simple XOR-cipher. Initially, a large set of input data bytes and all the possible key bytes are used to perform XOR-operations and the hamming distances of the resulting values are stored in a matrix called *Hypothetical Matrix*, as shown in Figure 3.2. The objective is to find the hypothetical key of the matrix that generates matching hamming distances for the same input values. For this, the input values are fed to the software running on the CPU where the XOR operation is performed with an unknown key. EM radiation are sampled for each input value to generate the *Real Matrix*, where each column contains a EM signal sample for its corresponding input. Calculating cross-correlations between rows of the two matrices can find the row in the *Hypothetical Ma-*

trix that provides the best correlation of hamming distances. The hypothetical key corresponding to this row is likely the encryption key used for the XOR operation inside the CPU.

Standard encryption algorithms, such as DES and AES, employ XOR operations at various stages in their functionality using chunks of the encryption key and input data. Therefore, attacks of the type of DEMA and CEMA are possible by attacking each chunk of the key being used with the XOR operations. Such an attack reveals parts of the encryption key, which have to be combined at the end. However, in a real-world setting, the attacker may not have enough EM radiation samples of the encryption operations to calculate the correct part of the key used. This results in lists of possible key chunks for each segment of the encryption key used in the algorithm. The problem of identifying the correct parts of the key to build the complete encryption key is called the *Key Enumeration Problem* which can be solved within a reasonable computational overhead [111].

Quisquater et al. practically demonstrated that EM analysis is a viable option to the aforementioned power analysis attack on computer CPUs [112]. By precisely moving the EM probe over an MCU, the authors were able to build an accurate 3-dimensional EM signature of the chip running an idle loop. It was shown that the radiation spectrum of each processor used in these experiments were sufficiently unique to use as a distinguishable feature for processor identification without applying an EM fingerprinting technique, such as RF-DNA. These experiments were performed in a Faraday cage to minimise external noise effects and the EM radiation were captured using a small magnetic loop antenna (diameter $\approx$ 3 mm). An oscilloscope digitised the signal for analysis. Gandolfi et al. applied DEMA to extract encryption keys from three different chips used on smartcards namely: COMP128, DES, and RSA [113]. According to their study, SNR of EM radiation from these chips is higher than the SNR of power consumption analysis. Therefore, it leads to the extraction of more information from the DEMA technique as compared to simple power analysis attacks [114].

While the XOR operations are targeted as the weak point in many attacks to DES and AES algorithms, it is not the only manner to successfully attack

encryption algorithms. Asymmetric key encryption algorithms, such as RSA, can be attacked by identifying the individual modular exponentiation operations performed within the algorithm through EM radiation [115].

Due to the modern electronic multimedia distribution model, e.g., the model for distributing music, movies, and books, end-users can become the attackers as well. These users might have the malicious intention of breaking encryption or sharing Digital Rights Management (DRM) protected data. Since the victim device is completely under the control of the attacker in such a scenario, unlimited physical access is available to the hardware and software through various side-channel attacks. White Box Cryptography (WBC) was introduced as a solution to this; whereby cryptographic algorithms and keys are combined with random code and random data to create an obfuscation that makes side-channel attacks more difficult. However, it has been shown that EM side-channel attacks, such as DEMA, are still capable of extracting encryption keys despite of the application of WBC techniques [116].

Recently, Camurati et al. made an important discovery, which extended the previously known capabilities of EM-SCA of cryptographic operations on IoT devices [58, 117]. It was shown that mixed-signal processors such as SoCs, which contains a radio transceiver and a CPU on the same silicon die, can cause long distance EM leakages. This occurs when the CPU noise gets modulated into the radio transceiver's radiation, extending the range of the CPU EM side-channel. As the usage of SoCs is getting increasingly popular on IoT devices, this latest type of EM side-channel leakage, named as *screaming channels* has hugely increased the potential attack surface.

3.4.3 Analysis of Wireless-powered Devices

Unlike traditional computing devices that have their own power sources to run CPU operations, wireless-powered devices, e.g., passive RFIDs, depend on an external RF field provided by the device's reader for power [118]. IoT devices are ideal candidates to be powered by wireless means. EM-SCA on such devices is challenging due to the presence of a strong RF field from the reader, since it obfuscates the weak EM radiation of the devices themselves. However,

RFID-based devices are being used in critical systems, such as secure access control to buildings and electronic payments, where cryptographic operations are performed on-board to verify the authenticity of the RFID device. Therefore, investigating the EM-SCA capability on such devices is important from both security and forensic standpoints.

Hutter et al. demonstrated the capability to perform EM side-channel analysis on RFID-based devices using a custom-made RFID tag as a proof of concept [119]. Using this custom set-up, the authors were able to recover the AES key used in a challenge response protocol between RFID tag and its reader. In order to avoid the disturbance from RF field of the reader device, RFID circuitry was placed outside the reader's RF field while power harvesting antenna is kept inside the reader's RF field. The two components were connected through a sufficiently long wire. While this enabled the measurement of the EM radiation from the RFID circuitry without interference, it is not possible to follow a similar approach in a regular RFID tag. This is because the antenna and RFID circuitry are inseparable by any reasonable means.

Kasper et al. performed EM-SCA attacks on RFID-based smart-cards in a more realistic setting. Their research employed commercially available smart-cards and performed the attacks within RF field of the RFID reader [120]. When the RFID smart-card is consuming more energy, amplitude of the RFID reader's RF field becomes lower. Meanwhile, when the RFID smart-card is consuming less energy, the amplitude of the RFID reader's field is higher. Accordingly, the power consumption of the RFID tag is reflected in the amplitude of the RFID readers carrier frequency. Therefore, it is possible to observe two side-bands around carrier frequency of the RFID reader reflecting this phenomena. Kasper et al. used this signal as the side-channel to attack the internal operations of the RFID reader. A computer-controlled USB-oscilloscope and a computer-connected custom-made RFID reader was used to attack an RFID tag while capturing EM fluctuations using a small RF loop probe. This set-up was used to perform a CPA attack to extract the symmetric keys used in DES and 3-DES implementations on the RFID tag successfully.

Souvignet and Frinken suggested that power analysis attacks can be used to extract data from smart-cards used by malicious skimmer devices as a

method to identify victims in a forensic investigation. However, it requires physically tapping into the device being investigated leading to potential inadvertent tampering of evidence [121]. A recent work by Xu et al. demonstrated that RFID-based smart-cards that employ side-channel attack mitigation techniques, such as *head and tail protection*, are not effective enough against EM-SCA attacks [122]. In their work, encryption keys used for 3-DES algorithm were demonstrated to be recoverable. In light of this attack vector, it is important to note that wireless-powered IoT devices are also susceptible to threats from EM-SCA-based attacks.

3.4.4 Countermeasures to Electromagnetic Side-Channel Analysis

As EM-SCA has been shown to be successful on recovering data from computing devices, various countermeasures have also been explored to counteract it on both software and hardware levels [75]. Masking variables by using random values alongside the operations is a basic software-based countermeasure, which has been proven to be not effective enough against EM-SCA attacks [123, 124]. Various other approaches exist such as randomising the operation sequences or lookup tables of algorithms [125, 126], avoiding instruction pairs executing adjacently that are known to emit distinguishable EM patterns [80, 81], and accessing critical data using pointers instead of values [127]. These approaches require further studies to see how effective they are against EM-SCA attacks.

Quisquater et al. suggests several hardware level countermeasures to these attacks [112]. Actions that can be taken by hardware designers includes minimising metal parts in a chip to reduce EM radiation, the use of Faraday cage like packaging, making the chips less power consuming (which leads to less unintentional radiation), asynchronism (i.e., designing the chips not to use a central system clock and instead operate asynchronously), and the use of dual line logic (i.e., using two lines that in combination of two bits represents a state instead of a single line that simply represent `0` or `1` states). Furthermore, it has been shown that it is possible to mathematically model an electronic

chip during the design phase to identify and avoid potential information leakages through EM side-channels [128, 129].

3.5 Standards and Tools

The concerns from software perspective of EM radiation from IoT devices are mostly concentrated towards wireless communication technologies such as WiFi, Bluetooth, and proprietary IoT protocols, e.g., Zigbee [130, 131]. Meanwhile, the unintentional EM radiation minimisation is generally left to those involved in the hardware design and manufacturing process. The term Electromagnetic Compatibility (EMC) refers to a device's unintentional EM radiation that can affect the functionality of other devices and the health of humans who are exposed to it [53]. The Federal Communications Commission (FCC), the Food and Drug Administration (FDA), the International Electrotechnical Commission (IEC), and the European Union (EU) are examples of authorities concerned with EMC regulation [132, 133]. However, regarding the question of EM side-channel information leakage from general purpose electronic devices, there are no such rules to govern the manufacturers. Instead, only guidelines exist, which may or may not be followed [134, 135].

Once a hardware device's design is completed and manufacturing commences, it is a challenging task to apply mitigation steps if the EMC tests reveal that it does not meet requirements. In the worst case scenario, the minimisation of EM radiation may require a complete rework of the PCB used in the device or a replacement of a critical electronic component. Due to the potential for costly manufacturing disruption, the minimisation of EM radiation needs to be ensured from the designing phase. Due to the fact that EM side-channel information leakage is not a problem limited to the hardware manufacturers, a joint effort by both hardware and software developers is necessary to establish standards.

In order to ease the job of information system security professionals to assess side-channel vulnerabilities of embedded systems, it is necessary to have tools. The Test Vector Leakage Assessment (TVLA) is a technique that can be

used to assess the resistance of cryptographic implementations against hardware side-channel attacks [136]. TempestSDR is a software tool that can be used with a large variety of hardware platforms, e.g., the Universal Software Radio Peripheral (USRP) [79] or HackRF [41], to eavesdrop on computer monitors by capturing the EM signals emitted by the video cables [137].

Multiple commercial and open source products exist that can be used to break the encryption on MCU-based IoT devices. ChipWhisperer [138, 139] is a widely used tool among security professionals and academic researchers to perform cryptographic key recovery attacks. It consists of a collection of open-source trace acquisition hardware and data analysis software components. Similarly, Riscure Inspector [140, 141] is a fully fledged commercial product that comes with software and hardware components to perform various power analysis and EM-SCA attacks to embedded devices including smart-cards. Blanco et al. presented a side-channel trace acquisition framework called SCAP, which is targeted at general purpose computing devices (including mobile devices). While the framework does not perform side-channel attacks currently, the objective is to provide a platform to build future analysis tools [142]. Such tools enable IoT system developers to test the robustness of their hardware against physical side-channel attacks and identify information leakage.

While these tools are focused on information security objectives, this thesis specifically focuses on the needs of digital forensic use cases. Therefore, it combines a large collection of EM-SCA methods under a unified methodology to acquire forensic insights from IoT devices.

3.6 Current Direction

With the current challenges in digital forensics and the state-of-the-art of EM-SCA, it is important to identify the future potential impact for digital forensics from these attacks. This section highlights some of the potential ways this impact may occur in the future under several key themes. Many of these approaches are already starting to be realised and others are ambitious predictions that can prove significantly beneficial.

3.6.1 Frequent Cryptographic Operations

While the increasing application of cryptographic protection on computing devices poses a challenge to traditional digital forensics, it can open up new opportunities to EM-SCA attacks [10]. EM-SCA attacks require a large number of traces acquired from a target device while the device is performing cryptographic operations using a single key. It has been demonstrated that such attacks are viable under laboratory conditions. However in most PC operating systems, it is rare to find practical situations where an attacker can observe EM radiation from a device for an extended period of time (since cryptographic operations typically occur less often than in the laboratory experimental conditions). The most common encryption occurring on many personal devices are Secure Socket Layer (SSL)-based web traffic [143].

Encrypted storage is becoming commonplace in both desktop and mobile devices. Recent versions of mobile operating systems, e.g., Android and iOS, secure their internal storage using encryption. Critical information necessary for digital forensic investigation can be inaccessible due to being stored in an encrypted form [31, 144, 145]. This includes encrypted emails, encrypted instant messenger applications, encrypted files, and encrypted storage partitions. However, it is highly likely that a mobile device is powered on when seized by law enforcement. Access to encrypted file systems causes an increased number of cryptographic CPU operations. Live data forensic techniques can help to perform investigations on such devices [146]. However, forensic investigators often encounter powered on, but locked, devices. As long as the device is reading and writing to the encrypted storage, EM radiation should reflect the cryptographic operations on the device. Therefore, an attacker can straightforwardly force the victim device to perform cryptographic operations in order to acquire side-channel traces for key extraction.

3.6.2 Combined Side-Channel Attacks

Instead of using a single side-channel attack in isolation, combinations of multiple side-channel attacks directed towards a single computer system can

prove more fruitful. It has been proven that power and EM-SCA can be combined to achieve better results [76]. There can be some operations of the CPU that are more clearly reflected in the device's power consumption than in the EM radiation and vice versa.

Sometimes, combining conventional attacks, e.g., spyware and worms, with EM-SCA attacks can provide new kinds of compound attacks that are difficult to counteract. For example, malware running on a victim computer can aid an EM-SCA attacker to extract additional information over the EM side-channel alone. This can be achieved through running specially selected instruction sequences on the CPU to intentionally emit encoded EM signals. Yang et al. [147] illustrated a mechanism to intentionally modulate EM radiation of electronic and electro-mechanical devices to exfiltrate data from the device to an external receiver. This hints at the potential for employing these unintentional EM side-channels to intentionally and covertly transmit data wherever necessary.

There are two potential avenues for malware assisted EM-SCA attacks. Firstly, malicious JavaScript can be embedded in a website, using Cross-site Scripting (XSS) or otherwise, and read the contents of a user's screen and encode that information into deliberate CPU EM radiation. Furthermore, TEMPEST style attacks on computer monitors can be combined with other attacks to increase the attack surface for air-gapped computer equipment [148]. For example, malware running on a target computer could read local files and encode that information into the computer's video output. Image steganography techniques can be used to hide the encoded data from the human user's view [149]. Meanwhile, a TEMPEST style attack can be performed on the computer's monitor in order to extract the video frames ultimately leaking data to the attacker.

3.6.3 File Signatures

Many types of digital multimedia content including images, audio, and video files are stored in a compressed format for the efficient storage and distribution [150]. As a result, when a computer starts playing an audio/video file in a

specific format, e.g., MPEG-2 Audio Layer III, AAC, MPEG-4, etc., or attempts to display a compressed image format, e.g., JPEG, GIF, etc., corresponding decompression software has to process the content. Since the software's execution path will be governed by the media file content, the instruction execution sequence will also depend on the media file. Therefore, it is possible that the CPU might emit EM patterns unique to a specific file being handled. This could potentially lead to the ability to identify the files being handled by a device.

While there have been attempts to make EM radiation signatures for hardware devices and specific software running on them for profiling purposes, such as RF-DNA technique [103], the possibility of profiling specific media files using the EM radiation caused by them is a potential avenue for the future exploration. Searching for a known file, such as known illegal content, in a target device is a challenge that the digital forensics community has been attempting to solve in efficient and effective ways because the manual comparison is often overly arduous for the expert investigators [151]. When a device is handling a file, passive observations of EM radiation can help to profile the file being handled by the device. This can be later be compared with a known set of file signatures to confirm the access or processing of a specific file on the target device.

3.6.4 Packet Analysis of Network Devices

There are a wide variety of special purpose computers being used in various specialised application environments including network routers and switches. There can often be an operational need to investigate a live network. In such cases, it is necessary to run network analysis software tools on specific interfaces at host computers [152]. Analysing the network solely based on the traffic going through routers and switches in order to observe live events is a challenging task. In such situations, the EM radiation of routers and switches are able to provide an approximate picture of the workload and traffic on the network [153]. It has been shown that EM radiation observed from Ethernet cables can lead to identifying the MAC addresses of the frames being handled by the networking devices [154]. In that demonstration, attackers has used a

technique similar to SEMA.

When IP packets are being switched at routers, the router has to update certain fields in the packet including TTL and the header checksum. After updating these fields, the router forwards the packet to the relevant network interface. If the EM radiation patterns of the router forwarding a packet to an interface and processing a packet are distinguishable, there are opportunities to perform interesting analysis on routers by observing their EM radiation. Packets that contains a specific payload, such as malware that comes from or addressed to a specific host, and network based attacks, e.g., Denial of Service (DoS) attacks, can be identifiable. Similarly, an attacker could gather EM radiation from a router to eavesdrop on the data being delivered through a wired network. Such possibilities are important from a digital forensic perspective when network analysis tools cannot be attached to a live system for analysis.

3.6.5 Easy Access to Electromagnetic Spectrum

EM-SCA attacks traditionally involve expensive hardware including RF probes, oscilloscopes, spectrum analysers, and data acquisition modules. Such devices are mostly used in EM-insulated laboratory environments. Moreover the configuration and operation of these devices require specialised domain knowledge. Information security specialists and digital forensic analysts might not have access to such hardware and might not posses the specialised knowledge required for their operation. While hobbyist attempts have been made to build such tools for lower costs, such efforts come with a penalty of lower precision and accuracy. This situation places a significant barrier to the wide adoption of EM-SCA.

Recent advancements in SDR hardware enable new opportunities for accessing radio spectrum for non-specialists. Affordable SDR hardware and freely available software libraries can be used to process and decode various wireless communication protocols. The ever-increasing processing power and memory capacity on personal computers supports the use of SDR software tools at high sample rates. EM-SCA attackers have recently started to

use SDR tools as a more affordable alternative to the expensive RF signal acquisition hardware. Following this trend, digital forensic analysis should be possible through the leveraging of EM side-channels detected on SDR-based hardware and software platforms.

3.6.6 Backscatter Channels

The unintentional EM radiation from computing devices can cause interference to other radio signals in the vicinity. This phenomena is evident in laptop computers which have been shown to modulate signals from commercial AM radio stations [155]. IoT devices already use this interference phenomena to communicate purposefully with other devices by modulating the ambient RF signals, which is called *backscatter* communication technology [156]. There are various carrier wave sources that have been tested in the literature for this purpose, such as TV transmission stations and WiFi access points [157, 158, 159, 160]. The potential of using this backscatter phenomena to eavesdrop on internal CPU operations of IoT devices by listening to ambient RF sources needs further exploration.

3.6.7 Advancements in Machine Learning

Recent advances that have been made in the area of Artificial Intelligence (AI) have demonstrated promising applications to many other domains across computer science. Various tasks where human intuition was required to perform decision making are now being replaced with machine learning and deep learning based algorithms. Software libraries and frameworks are becoming increasingly available in order to assist the building of applications that have intelligent capabilities. Examples include the automated detection of malicious programs [161], image manipulation [162], and anomaly detection in network traces [163].

EM-SCA techniques, such as SEMA and spectrogram pattern observations, that previously required human intervention can be automated through the development of AI algorithms. Wang et al. [98] applied deep learning al-

gorithms such as MLP and LSTM to detect anomalies in the code of simple IoT devices such as Arduino and Raspberry Pi through the power consumption side-channel. Therefore, it can be possible to extract better information from EM traces than the current manual observations are capable of achieving. Several examples that were discussed in previous sections already leverage AI techniques to recognise EM trace patterns, which strongly hints the future role ML algorithms can play in EM-SCA for digital forensics [85, 164, 90, 91, 96, 105, 165].

Chapter 4

Methodology: The Birth of EMvidence

4.1 Introduction

The high-level problem tackled by this research is enabling digital forensics of IoT devices through their EM side-channel radiation. Towards this goal, three research questions were identified that need to be addressed to solve the challenge of forensic insight gathering from IoT devices. First of all, a question was raised whether machine learning methods can be used to extract forensically-useful insights from such devices. Secondly, a question was raised regarding the efficiency of such ML-based insight-gathering methods as it is an absolute necessity to make such methods usable in practical digital forensic scenarios. Thirdly, the attention was drawn to the diverse and dynamic nature of IoT device ecosystem and the challenge it poses to manage the methods that are to be discovered to gather forensically-useful insights.

A careful reasoning between the three research questions highlights that the methods to gather insights and to do it efficiently will not be so useful if they are targeting a limited set of IoT devices, which will be obsolete in a few years time. Therefore, the long-term success of the answers to first and second research questions directly rely on a sufficient answer for the third research question on managing diversity and dynamism. Considering this im-

portant correlation between the research questions, the solutions presented in this thesis, first of all, draws a high-level outline of a model to tackle third research question. Based on this model, a software framework is designed and implemented. Later, this thesis moves on to address the first and second research questions by implementing and testing methods that are integrated into the aforementioned software framework eventually.

The rest of this chapter is organised as follows. Section 4.2 discusses the need for a EM-SCA-based forensic investigation model using an example scenario where the requirements that should be met by such a model are illustrated. Section 4.3 details the proposed model for EM-SCA-based forensic investigation. Section 4.4 presents the design and the implementation of a framework called *EMvidence*, which is based on the proposed model. The plug-ins of the *EMvidence* framework are further detailed in Section 4.5. The procedure of EM data acquisition from IoT devices were illustrated in Section 4.6. Finally, Section 4.7 briefly highlights the experiments presented in the following two chapters using the *EMvidence* framework.

4.2 A Case Study Scenario

Investigating an Arson Attack:

Consider a fire warning system deployed inside a research laboratory[1]. The device is designed to run independently without communication with the outside world most of the times. Its firmware is programmed to run by transferring between a predefined set of states. Figure 4.1 illustrates the state machine of the device firmware. The device remains powered up continuously. It can be powered off only by physically opening its casing and turning an internal switch, which is protected by a physical lock. Therefore, shutting the fire warning system off is only possible for personnel with proper credentials.

The device firmware basically has 5 internal states when its up and running. Initially, it stays on *Smoke Detection* state, where it keeps reading a

[1]This application is inspired by real-world smart fire warning systems, such as Google's Nest Protect – `https://store.google.com/ie/product/nest_protect_2nd_gen`.

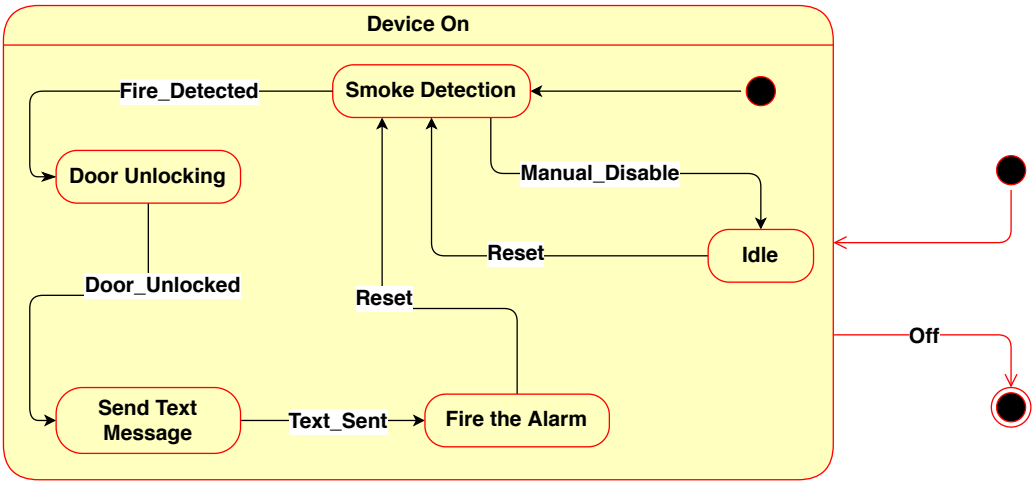

Figure 4.1: State machine of the IoT fire warning system's firmware.

smoke detecting sensor input. Whenever a potential fire is detected, the device sends a message to the smart door lock in the research laboratory to unlock the door so that any people inside the premises can escape to the outside. Then, the device sends a text message to a predefined person to notify the incident, i.e., *Send Text Message* state. Finally, the device moves to *Fire the Alarm* state where it remains firing an alarm. From here, the device can be turned back to *Smoke Detection* state by pressing a reset button. Furthermore, pressing another button, the device can be switched to *Idle* state where the device does nothing but stands on alert for any further moves from the user. The device does not keep records of its internal activities locally or remotely in any form, other than the text message it sends out to notify a fire.

Consider law enforcement entering into the research laboratory due to a report on a fire in the building. The firefighters have already diffused the fire. According to their report, the fire warning system has not been ringing when they arrive at the scene. They assume that a fault in the IoT device has prevented it from ringing. The authorised person who was supposed to receive a notification via a text message has not received it either. However, there is a suspicion that an insider who is knowledgeable about the IoT device has disabled the device, placing it into *Idle* state before performing an arson attack. An important piece of clue to answer the mystery is hanging on the volatile

memory of the IoT device firmware.

The Current Approach:

As IoT devices are usually designed to be always on, it is highly possible to have a seized IoT device still operational. Before attempting to acquire any non-volatile storage data physically from the device, the device must be switched off in order to prevent any physical damage to the data. The longer the device is running, the higher the risk of contaminating device data. If the device is connected to the network, it can potentially receive remote commands to wipe its internal storage. Therefore, following the usual practice, the investigators take the fire warning device into custody, turn the device off, and transfer it to a digital forensic laboratory for inspection [166]. If the device stores data that are not encrypted and the device has a standard interface, the data can be extracted using existing forensic evidence acquisition methods [167]. This particular IoT device lacks these standard interfaces, forcing investigators to take more risky approaches, such as chip-off forensics. Mistakes during such operations could inadvertently destroy the device itself. However, since the device in question in this particular scenario does not store any useful information, nothing of useful value to the investigation can be found from the device at the end.

The Ideal Approach:

Upon arriving at the scene, the investigators should be able to recognise IoT devices available in the surrounding environment. Once the IoT device that belongs to the intruder detection system is noticed, the make and model of the device must be identified by visual inspection. The next step should be to operate a portable system closer to the target IoT device in order to pick EM radiation from the device and analyse it immediately to produce a report. The analysis should be able to recognise the firmware running on the device and the current status of it. If the analysis revealed that the device is currently in the *Idle* state, that indicates that someone from the inside of the building has

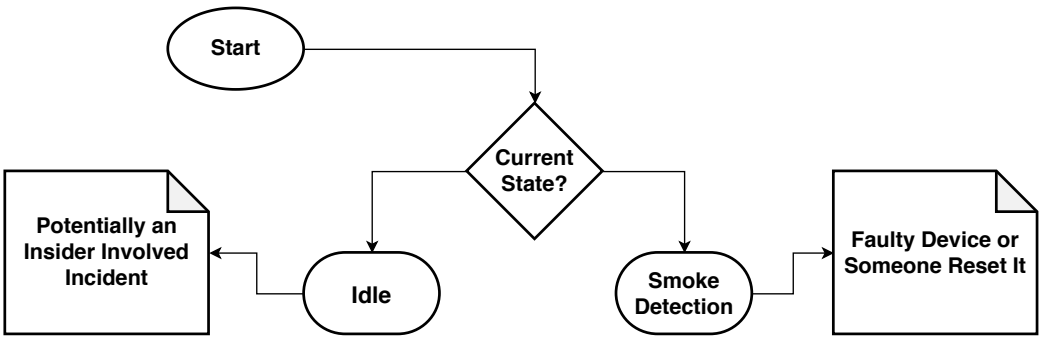

Figure 4.2: Reasoning with the information of IoT device firmware internal state.

disabled the fire detection system, pointing to an insider job. If it was found that the device is in *Smoke Detection* mode, that can point to two possibilities. It is either the system never detected the occurrence of fire due to a technical fault or someone has reset it when the alarm started to ring. In such a case, evidence from other sources such as testimonies from neighbours can help to verify whether anyone actually heard the alarm ringing even for a brief period of time.

Manually switching the device to *Idle* state means, an insider who knows the presence of the system carried out the arson attack. Causing the system to trigger first and then resetting it later indicates that the criminal was not an insider as they were not aware of the presence of the system (see Figure 4.2). The insights gathered from the EM-SCA of the fire detection system combined with other sources of evidence can finally point to uncover the mystery of the arson attack – whether the criminal was an insider or an intruder. Some similar systems may even contain certain internal flash storage that may leave non-volatile clues about the behaviour of the system. Due to the unavailability of standard interfaces to the system, inspecting the flash storage by performing a chip-off can be a viable option as well. However, even when chip-off forensics is being considered, it may be a good approach to first try an EM-SCA inspection on the device to gather as much information as possible before attempting a chip-off.

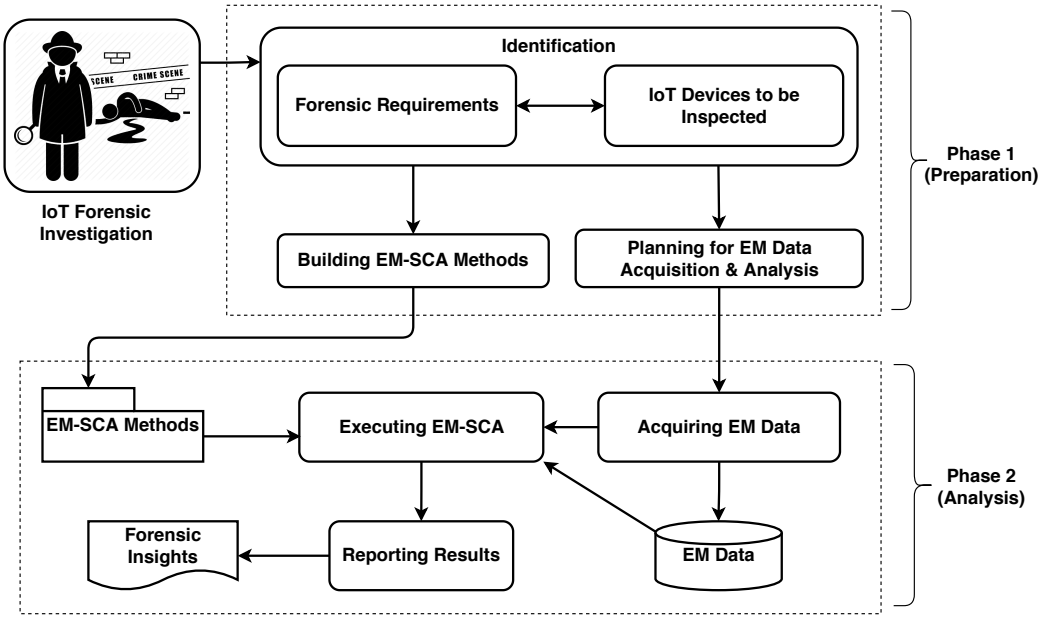

Figure 4.3: A forensic model for IoT forensics using EM-SCA methods.

4.3 A Forensic Model for Internet of Things using Electromagnetic Side-Channel Analysis

Due to the potential existence of a variety of EM-SCA methods to acquire forensic insights from a large diversity of IoT devices, the complexity of the task demands a standard procedure for investigators [10]. Therefore, this work proposes a new EM-SCA-based IoT forensics model as depicted in the Figure 4.3. The proposed model organises the EM-SCA-based IoT investigation procedure into two main phases at a high-level. The first phase focuses on the identification and the preparation of necessary environment, including the requirements, the IoT devices, the EM data acquisition equipment, etc. The second phase focuses on the data acquisition and analysis to uncover forensic insights. The components of the proposed model are described in the following series of subsections.

4.3.1 Identification of Requirements

At the beginning of an investigation, the first task is to identify the forensic requirements of the investigation. These requirements depend on the exact scenario of the incident being investigated. For instance, in the scenario discussed in Section 4.2, the major requirement is to uncover whether someone intentionally caused the fire in the building, i.e., an arson attack.

The next important task is the identification of IoT devices available to the investigator. When such an IoT device can fulfill the forensic requirements identified previously, it should be counted as a device for EM-SCA. In the scenario discussed in Section 4.2, the question whether someone intentionally caused a fire in the building can be converted to the question whether someone intentionally disabled the IoT fire warning system. This question can be answered by detecting the internal state of the IoT fire warning system using EM-SCA methods.

4.3.2 Planning for Data Acquisition and Analysis

Having identified the forensic requirements and the IoT devices that can fulfill those requirements, the investigator can now plan the data acquisition procedure accordingly. Multiple decisions have to be made during this planning. From the perspective of methods, it is necessary to decide which EM-SCA methods should be applied to acquire the identified forensic insights. From the perspective of hardware settings, the exact frequency of acquiring data, the positioning of the antenna, i.e., both its location and its height over the IoT device, and the duration of EM data acquisition have to be decided.

It is recommended to equip the investigators with a checklist that can ease this decision-making process and release the investigator from having domain expertise on EM-SCA. For example, the recommended EM data acquisition settings for a particular IoT device type, such as frequency of the radiation, sample rate, positioning of the antenna, etc., should be included in such a checklist, which can be directly followed by the investigator. Whenever new EM-SCA methods are developed to inspect the IoT devices, such checklists

should be provided along with the new methods.

4.3.3 Building New Analysis Methods

In certain cases, the investigators may come across requirements that cannot be fulfilled with the existing EM-SCA methods. Similarly, the investigators may come across new IoT devices that are currently not supported by the existing EM-SCA methods. In such situations, it is important for the investigators to properly document such new requirements in order to be addressed in the future. The research community can develop or improve EM-SCA methods based on these documented requirements. For the continuous applicability of EM-SCA in IoT forensics, such regular improvements and research are vital.

4.3.4 Acquiring Electromagnetic Data

With this step, the investigator enters the second phase of the EM-SCA forensic model. The EM data acquisition procedure should be carried out according to the plan prepared previously. If there is a possibility, a portable shielded environment, i.e., a Faraday cage, should be used to prevent the contamination of EM data from external noise. A careful documentation of the hardware settings and the procedure followed is necessary during this stage. The insights gained and the conclusions arrived at the end can be questioned and challenged based on such details, or the lack of them.

4.3.5 Executing Electromagnetic Side-Channel Analysis

Once the EM data have been acquired, the analysis should be conducted using the EM-SCA methods decided at the planning stage. If the outcome of the analysis can be used to conduct subsequent forensic analysis using IoT devices, it would be necessary to perform EM-SCA on-the-spot. In other cases, the data analysis can be conducted later in a forensic laboratory. When applying EM-SCA methods to analyse EM data, it is important to use well-recognised implementations of the methods. Just like any other digital forensic

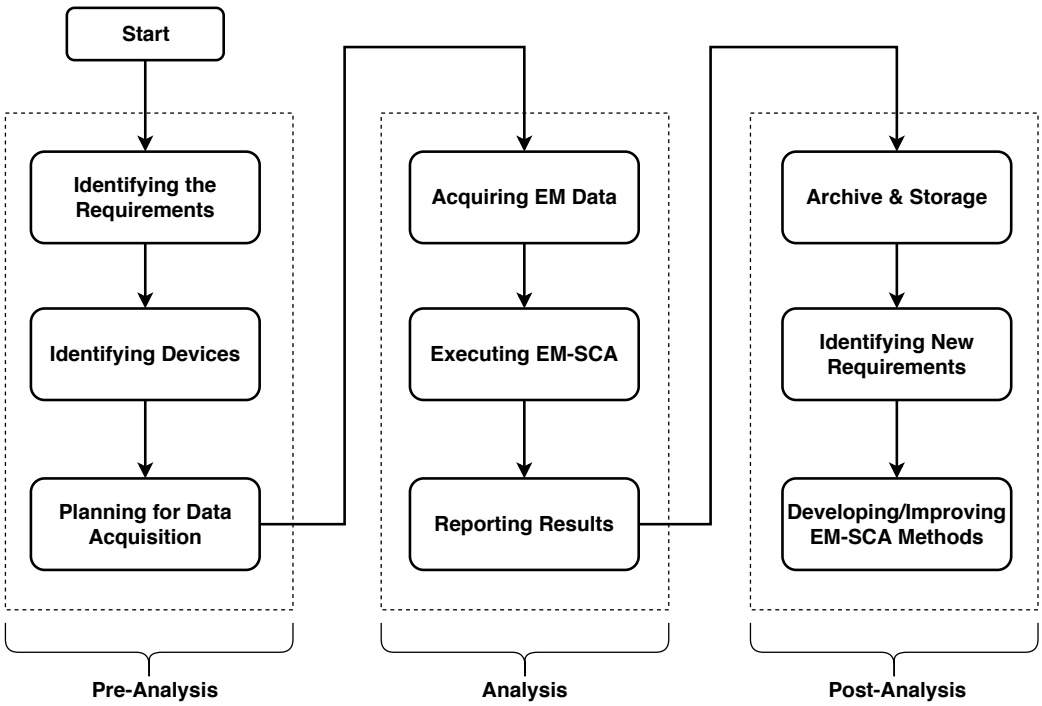

Figure 4.4: A potential investigative workflow that follows the proposed forensic model.

tool used in investigations, the recognition of the EM-SCA tools used in this phase is highly important.

4.3.6 Reporting Results

At the end of the analysis of EM data, the findings should be reported in a format acceptable in the legal context. It is necessary to report the exact hardware settings and procedures followed to acquire EM datasets in detail along with the outcome of each EM-SCA method applied. Some of the outcome of the analysis can be direct court-admissible evidence while many others can be forensic insights useful to the investigators to make further investigative moves.

4.3.7 Overall Workflow

While there are many activities included in the proposed model, a typical investigation can follow a sequential workflow as depicted in Figure 4.4. The process begins with the identification of forensic requirements and the IoT devices. Based on these information, the investigator can plan the data acquisition and analysis procedure in detail. After the planning, the EM data acquisition, the analysis and the reporting of results can be conducted. Finally the post-analysis activities includes the storage of investigative data, the identification of requirements, which could not be met with existing EM-SCA methods and the implementation of those identified new EM-SCA methods for the future use.

4.4 The *EMvidence* Framework

Compared to the conventional digital forensic methodology, the proposed model differs in two important aspects [4]. Firstly, the data acquisition and analysis steps of the proposed model are designed to be executed immediately after a device is found at the triage examination, whereas the conventional forensic data acquisition and analysis mostly occur at a forensic laboratory. Therefore, these steps are intended to be executed by non-expert law-enforcement officers at crime scenes instead of highly trained digital forensic investigators at laboratories. Secondly, the proposed model adds an emphasis to the identification of new forensic requirements and the development of new methods to cater them. Unlike the conventional forensics, this is important in EM-SCA forensics as the IoT devices and their requirements rapidly change.

The real-world application of the proposed EM-SCA-based IoT forensics model depends on tools that facilitate its successful execution. Especially, the data acquisition and the analysis steps in the proposed model raise difficulties to investigators due to the diversity and dynamism of the IoT ecosystem – as pointed out by the third research question of this thesis. In order to meet the requirement, a framework called *EMvidence* was designed. In the design process, in order to keep up with the diversity and dynamism of IoT ecosystem,

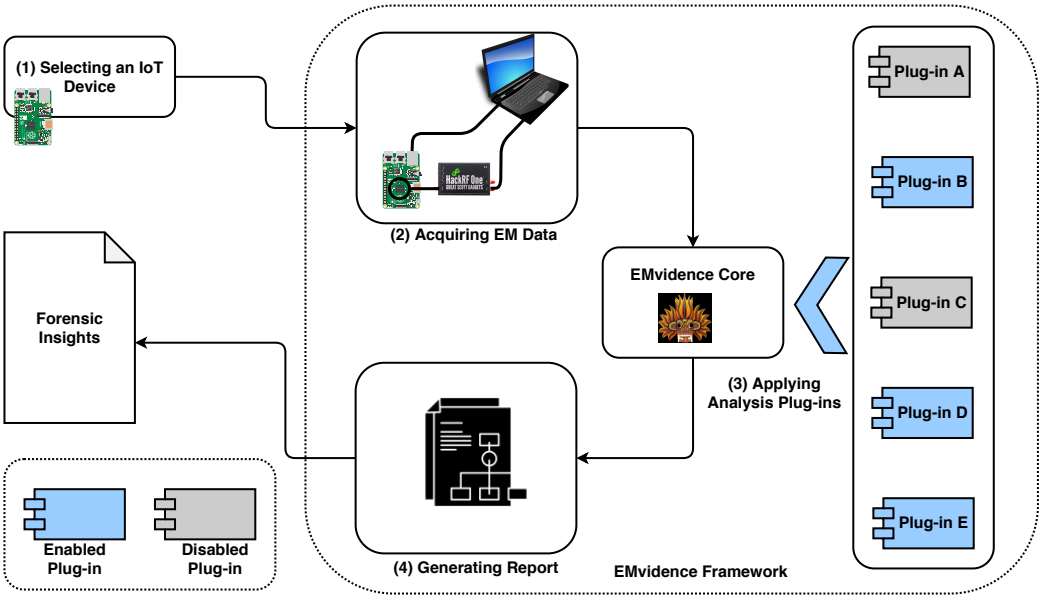

Figure 4.5: Major functional components of the *EMvidence* framework and their involvement in the workflow of analysing an IoT device.

Unix philosophy was adopted [20]. According to the Unix philosophy software systems need to be modular where each module is independent from the rest of the system. A module is specialised and optimised in doing one thing and one thing only in the most efficient way. This enables the easy and efficient maintainability of the entire system – a requirement to change or entirely replace a component does not affect other components of the system.

Regardless of the type of IoT device being analysed and the nature of the forensically-useful insight being unveiled, there are two important functionalities required by the framework that are common to all types of analysis. The first is capturing EM radiation from a DUT with appropriate settings of an SDR hardware. Secondly, the framework need to produce a report based on the analysis of the EM radiation data that is comprehensive enough in forensic contexts. Due to the independence of these two functionalities from the specific EM data analysis procedures, they are included as default functionality of the framework. Meanwhile, each piece of data analysis functionality is distributed into independent plug-ins. Figure 4.5 illustrates the major functional components of the *EMvidence* framework and their involvement in the work-

flow of analysing an IoT device. These framework components are explained in detail in the following subsections.

4.4.1 Data Acquisition Component

One of the default components of the *EMvidence* framework is the facility to acquire EM data from IoT devices. When acquiring EM data from a device, multiple parameters are necessary: the centre frequency of the radiation, the best position on the device to capture the radiation, sample rate, and the duration of the data acquisition. Section 4.6 details the procedure to determine these parameters. Whenever a new IoT device is supported by the framework, these parameters should be determined and embedded into the framework so that the investigators can straightforwardly use those parameter settings when dealing with an IoT device under investigative conditions. The EM data acquired by the *EMvidence* framework are stored in I/Q data format regardless of the specific hardware equipment used to acquire them [49]. By doing so, EM-SCA methods that perform data analysis can access and process the data uniformly.

4.4.2 Report Generation Component

The second default component of the *EMvidence* framework is the generation of reports based on the EM-SCA of acquired EM data. The results of EM-SCA can be in both graphical and textual format. Graphical results include the visualisation of interesting findings from the EM data such as a confusion matrix depicting the EM signal classification results. Textual results includes numerical values such as classification accuracy of a given EM dataset. All these analysis results from a particular EM dataset are combined along with the details of the EM dataset itself into a single report. The report produced by an analysis of an EM dataset can consist of both direct court-admissible evidence and forensic insights that can assist in further stages of the forensic investigation.

4.4.3 EMvidence Core

The core of the *EMvidence* framework is responsible for the management of individual components: the data acquisition component, report generation component and the collection of plug-ins. A plug-in in *EMvidence* is an implementation of a unique EM-SCA method targeted at acquiring a specific type of forensic insight from a specific type of IoT devices. A large collection of plug-ins are necessary to support the diversity of IoT device ecosystem. New plug-ins should be implemented regularly to facilitate the dynamism of IoT ecosystem whenever a new type of IoT device enters the market and draw the attention of forensic community. The *EMvidence* core facilitates the user firstly to acquire EM data using data acquisition component and secondly to select a set of plug-ins, which should be used to analyse the acquired EM dataset. Once selected, the *EMvidence* core individually activates each plug-in to process the data and retrieve the output produced by each of them. Finally, the produced results are directed to the report generation component to compose final analysis report.

As it is evident, the EM-SCA capability of the *EMvidence* framework relies on the methods implemented as individual plug-ins. The functionality of plug-ins are described in detail in Section 4.5 along with the procedure to construct them.

4.4.4 Implementation Details

The *EMvidence* framework was implemented using Python language that provides many advantages. When accessing SDR hardware tools, it is necessary to use a well supported software library. GNU Radio, as a free and open-source library that supports almost all the SDR hardware currently in use, is the ideal choice for this purpose [47]. While the core parts of GNU Radio is implemented in C++ language to maintain high performance, its functionalities are wrapped to support Python language. Therefore, the *EMvidence* implementation benefits from a smooth integration to GNU Radio in the same language. When processing EM data on the *EMvidence*, a multitude of Python

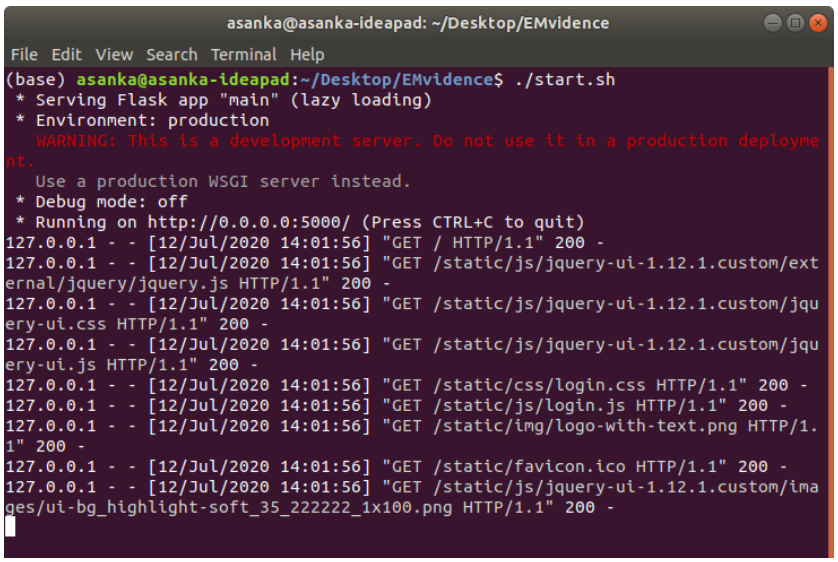

Figure 4.6: The output on the Bash terminal when running *EMvidence*.

libraries are used, such as numpy and scipy [36]. The visualisation of EM data was performed with the help of matplotlib library. The framework was implemented on GNU/Linux operating systems. However, due to the use of Python language and libraries, porting *EMvidence* to work on other operating system platforms can be straightforward.

The GUI components of the *EMvidence* framework are implemented using the Flask library, which is a framework to build web applications. The source code of the *EMvidence* framework and its associated plug-ins are available at a Github repository[2]. Currently, the repository provides a few plug-ins to demonstrate the API facilities. Plug-ins that are aimed at real-world IoT devices will be added to the repository in the future. Furthermore, the framework facilitates future research and development of *EMvidence* plug-ins by third-parties. Figures 4.6 depicts the start of the *EMvidence* GUI by running a shell script, i.e., `start.sh`, on the Bash terminal. The *EMvidence* GUI currently facilitates capturing EM data and analysing them to produce reports. Furthermore, EM traces captured using external means other than the *EMvidence* framework can be uploaded into it as well (see Figure 4.7). Once the data are analysed using one or more modules a comprehensive report is produced, as

[2]https://github.com/asanka-code/EMvidence

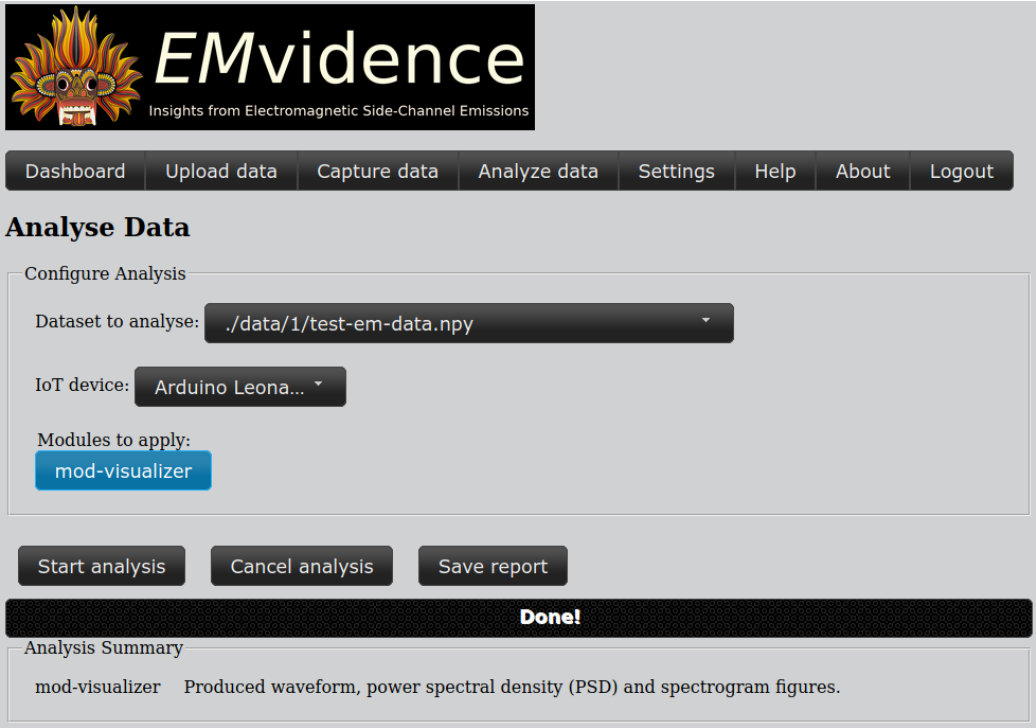

Figure 4.7: Analysing an EM trace using the *EMvidence*.

shown in Figure 4.8, that can be downloaded.

In traditional digital forensics, the data acquired from devices under investigation are handled in a forensically sound manner in order to ensure the court-admissibility of evidence[4, 10]. For example, when a disk image is acquired from a computer, cryptographic hash values are calculated and stored alongside the disk image. The hash values can later be used to verify the integrity of the disk image. Similarly, EM traces acquired from IoT devices has to be stored with a hash verification facility. The *EMvidence* framework supports hash calculation for EM traces acquired in real-time and stores them along with the traces. However, the patterns of the EM signals from an IoT device depends on its current internal states and external noise sources in the vicinity. Therefore, hash values are useful only to maintain the integrity of the originally acquired EM traces during subsequent analysis.

EMvidence - Analysis Report

Report of the electromagnetic side-channel analysis on a target device using EMvidence framework.

Analysis Date/Time: 2020-06-01 17:26:39
Device Under Investigation: Arduino Leonardo
EM Data File: ./data/2/more-data.npy
Hash Value (sha256):
3a79f49866dd2ede44f8455daf5138ad97e492ccbe08fdcec970e1f87f532d11

Analysis Results:

Module ID: 1
Module Name: mod-visualizer
Module Description: This module produces visualizations of EM traces.
Module Output:
Visualizations of the data are illustrated in the following figures.

Figure 4.8: A report generated by *EMvidence* after analysing EM data from a device.

4.5 Plug-ins for EMvidence

As evident from the high-level architecture of *EMvidence*, the capability to analyse and gather a multitude of insights from IoT devices relies on the availability of plug-ins. A plug-in in *EMvidence* is intended to uncover a specific insight from a specific type of IoT devices using the EM data provided by the *EMvidence* core. In order to achieve that goal, it can be developed to use its own EM data processing and analysis methods. Within the scope of this work, EM-SCA methods that use ML techniques are considered and therefore, *EMvidence* plug-ins can use ML-based EM-SCA methods for gathering forensic insights [98, 164, 165].

4.5.1 Plug-in Behaviour

Figure 4.9 illustrates how a plug-in interacts with the core framework when trying to analyse an EM dataset. Once the EM radiation of a DUT is captured,

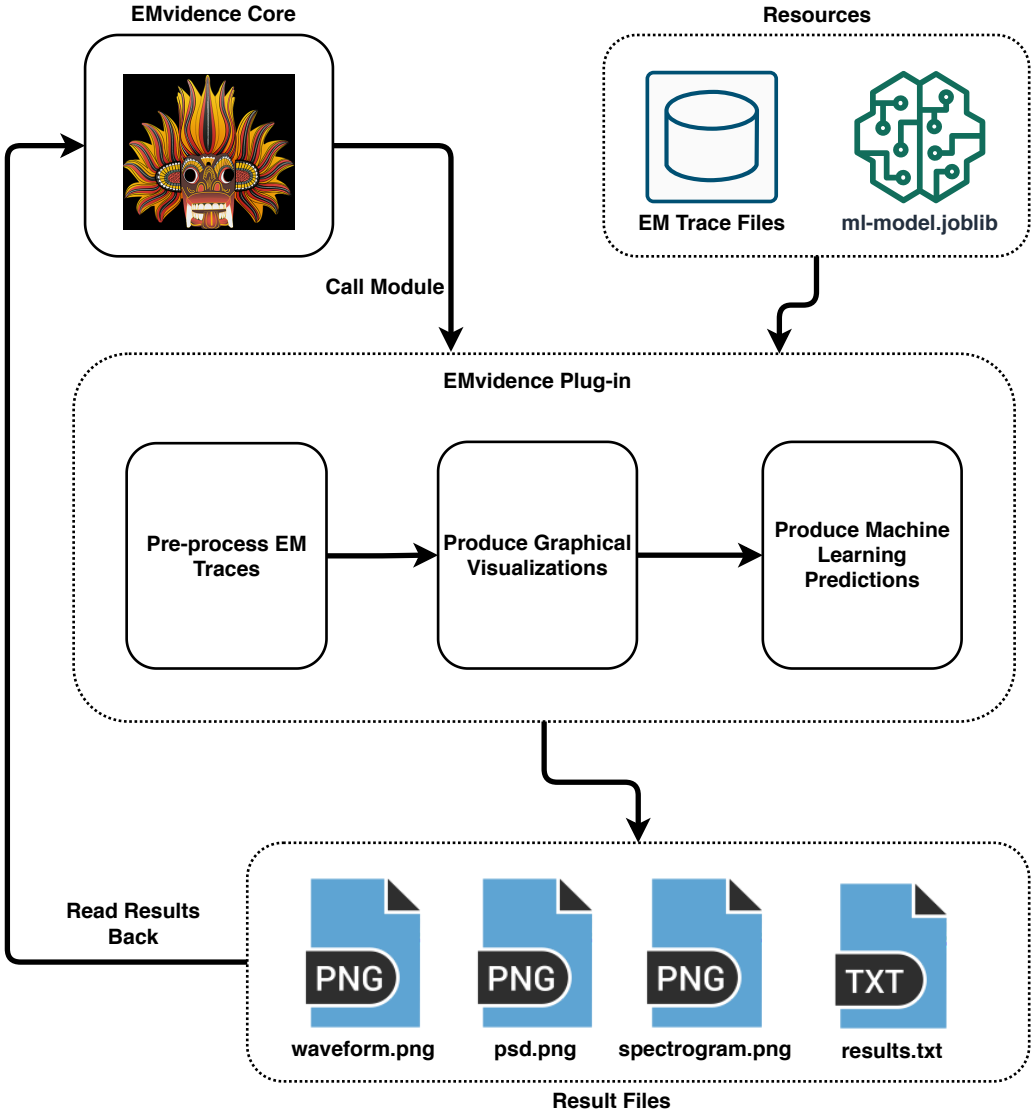

Figure 4.9: A plug-in being called by the *EMvidence* framework's core.

the user of *EMvidence* selects the set of plug-ins that should engage with the data. When a particular module is called, it is provided with access to the EM data that need to be analysed. Upon being called, a plug-in usually preprocess the input EM data files according to its needs. The preprocessed data can be used to produce two types of output from a plug-in. The first is graphical results that can be visualisations of the analysis. The second is textual results that are usually numerical values produced by ML-based analysis. Trained

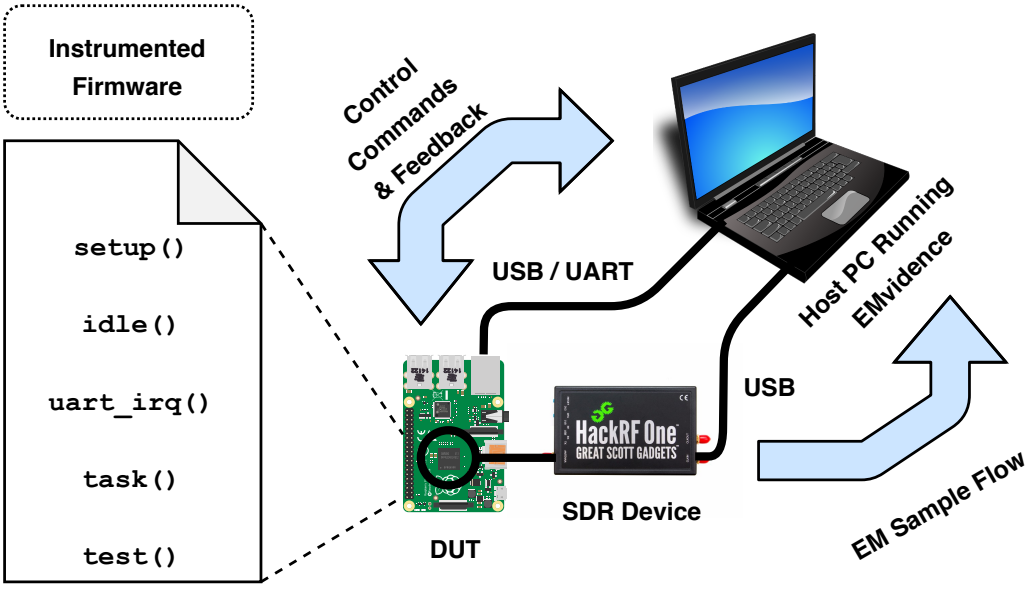

Figure 4.10: Instrumented and controlled EM signal acquisition.

machine learning models that are shipped with the plug-in can be used to produce the latter. The graphical results are individually written into image files in PNG format while textual results are written to a text file with UTF-8 text format. After the plug-in completes its analysis and produce result files, *EMvidence* core reads the produced result files and uses them to compose the final analysis report.

4.5.2 Plug-in Development

When developing a new plug-in for the *EMvidence* framework, two decisions have to be made initially. The first is the type of IoT device that will be considered as the DUT. The second is the precise forensic insight that should be identified by inspecting EM radiation data of the DUT. In order to capture a sufficient EM radiation dataset, a representative DUT is taken and EM data are acquired from it. The EM data acquisition capability of *EMvidence* is used for this purpose.

Depending on the nature of the DUT and the intended insight that should be retrieved, there are two potential approaches to acquire EM data to build a

plug-in. The first approach is observing the device entirely passively. This is useful in situations such as identifying a known version of a device firmware or detecting modifications to firmware. The second approach is capturing data from an instrumented DUT. In situations where it is possible to simulate a particular type of IoT device using a prototyping platform, such a platform can be used to acquire EM data with active manipulations during EM data acquisition. Figure 4.10 depicts such a situation where the DUT is connected to the host computer running *EMvidence*. By sending control commands through the USB/UART port, the DUT firmware can be placed onto different internal behavioural states in order to capture EM data samples for each behaviour of the device.

Once the EM data are acquired, the development of code to process and train an ML model can be done. The *EMvidence* framework provides an API with functions to facilitate this task. The developer of the plug-in can make use of these facilities depending on the forensic insight aimed to acquire from the EM dataset. This phase of the work produces two outputs. The first is software code implementation of data processing functionality. The second is the trained and tested machine learning model. Therefore, a plug-in is essentially a package of the following components.

Machine Learning Model: For *EMvidence* plug-ins that uses ML methods to extract forensic insights from EM data, a trained ML model should be enclosed in the plug-in package. Trained models saved in *joblib* format can be used for this purpose, e.g., *ml-model.joblib*.

Python Scripts: One or more Python scripts that implement certain functions, which will be called by the *EMvidence* core should be included in the plug-in package. *EMvidence* can invoke the plug-in and retrieving the results generated by it using these Python scripts. The code within this script will process an EM trace given to it and use an appropriate method to extract forensic insights, such as using the enclosed machine learning model to make predictions. It will finally produce the graphical and textual results as files.

Configuration File: A configuration file, which will be read by the *EMvidence* framework should be included in the plug-in package. The configuration file contains the details that are necessary to successfully integrate a new

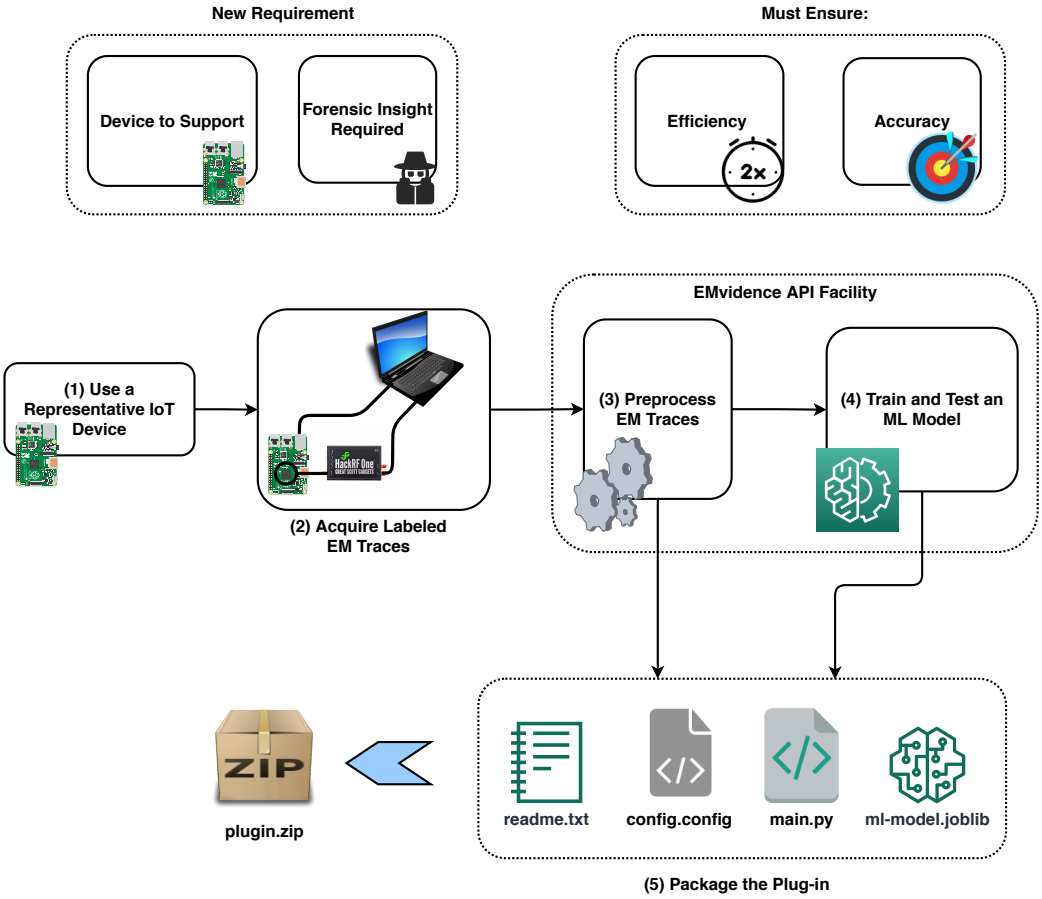

Figure 4.11: The workflow of creating a plug-in for *EMvidence*.

plug-in with the core of the *EMvidence* framework.

Readme File: A text file should be included in the plug-in package that contains any useful information to the end-user of the plug-in. The user can refer to the readme file of a new plug-in before adding it to the framework in order to ensure that the plug-in provides the expected functionality.

Once these files are prepared, they are packaged into a single compressed file. After that, the plug-in can be distributed. A potential user can upload the compressed package into the *EMvidence* framework so that the framework's plug-in management system can extract and integrate it into the existing collection of plug-ins. Figure 4.11 depicts the entire process of creating and packaging a plug-in for the *EMvidence* framework.

4.6 Procedure for Data Acquisition

4.6.1 Representative Internet of Things Devices

When performing experimental evaluations of the proposed methods, it is necessary to use sufficiently representative IoT hardware platforms. IoT devices can be categorised as low-end, middle-end and high-end devices [168]. For the experimental evaluations, two representative IoT device platforms from the low-end and high-end device classes were used; namely a Raspberry Pi 3 B+ [169] and an Arduino Leonardo [170].

The Raspberry Pi 3 B+ device consists of an ARM Cortex-A53 quad-core processor running at 1.4 GHz clock speed. It has a memory capacity of 1 GB. Furthermore, it has WiFi, Bluetooth 4.0, and Ethernet for communication. All of these resources represent the class of a high-end IoT device that is capable of running a Linux-based operating system and comparatively memory- and processing-intensive applications. Therefore, this device can be used to easily emulate various existing IoT devices during experimentation. Meanwhile, the Arduino Leonardo device consists of an 8 bit MCU with an AVR architecture that runs at 16 MHz clock speed. It has 2.5 KB memory that is barely enough to run simpler programs. Therefore, it can be considered as a representative of a lower-end IoT device.

The Raspberry Pi device was running Raspberry Pi OS, which is a variant of the Debian GNU/Linux operating system [171]. When performing experiments to observe EM radiation from the Raspberry Pi, it was connected to a host computer through the Ethernet port and logged into remotely through a Secure Shell (SSH). This enables the control of the Raspberry Pi through the host computer remotely during experimentation. Similarly, Arduino device was connected to the host computer through a USB port. Since Arduino does not contain an operating system, control of the device through USB port was conducted by programming it to take inputs as serial data depending on the requirements of each experiment.

When observing EM radiation from IoT devices, a near-field H-Loop magnetic antenna with a diameter of 15 mm was used in all experiments. The

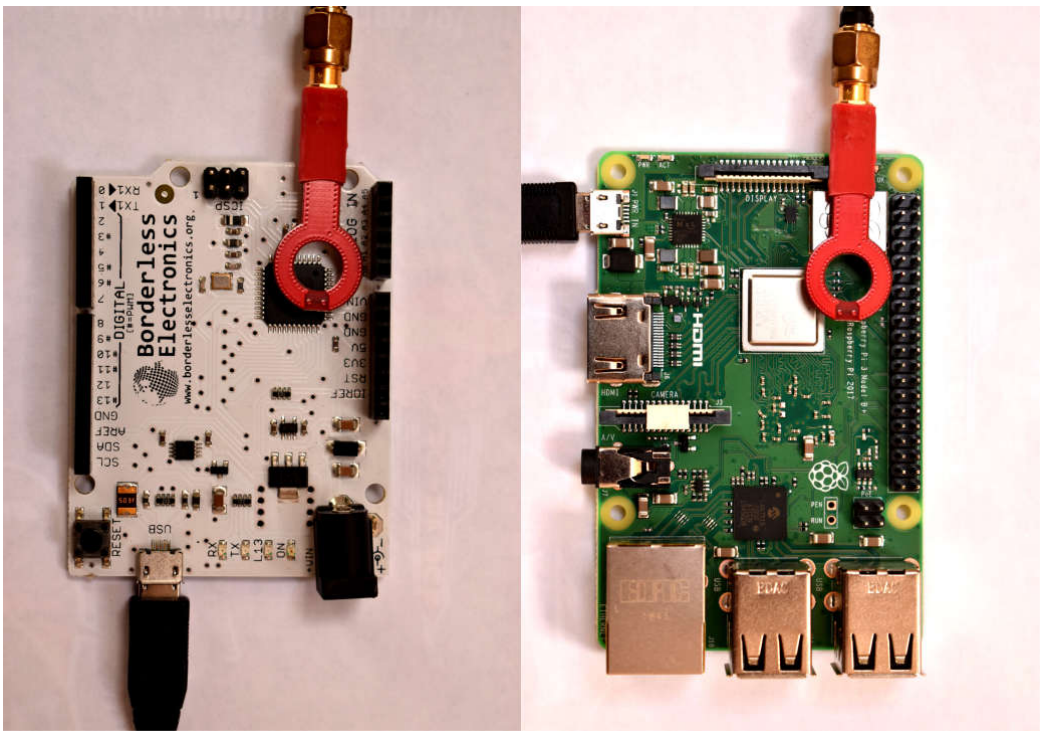

Figure 4.12: Arduino and Raspberry Pi devices with H-loop antennas.

HackRF One SDR device was connected to this H-loop antenna through a semi-rigid RF cable in order to position the antenna in a desired place for a prolonged period of time for experimentation. Both the SDR device and the IoT device being observed were connected to the same host computer to control the experimental setup and to store the captured EM data. As shown in Figure 4.12, the H-loop antenna is placed right on top of the processor chip of the IoT devices.

4.6.2 Determining Data Acquisition Parameters

The acquisition of EM data from an IoT device depends on a set of important parameters: (1) the information-leaking signal frequency, (2) the origin location of the signal, (3) the data acquisition sample rate, and (4) the duration of data acquisition [50]. Before retrieving a particular forensic insights from a particular IoT device in an investigative scenario, these parameters need to

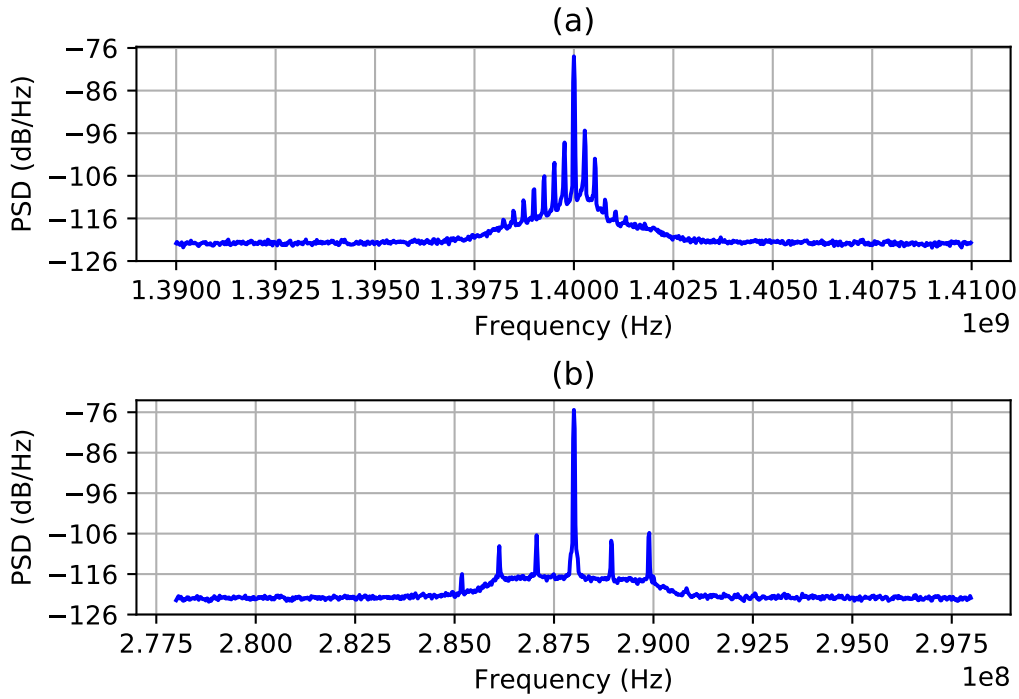

Figure 4.13: Leakage signals of two representative IoT devices – (a) Raspberry Pi 3 B+ at 1.4 GHz and (b) Arduino Leonardo at 288 MHz (18^{th} harmonic).

be predetermined and provided to the forensic investigators as a part of their EM-SCA tool set.

Information-leaking Frequency: The successful observation of EM radiation from a target IoT device requires the precise identification information-leaking frequency of the particular device. A wide variety of frequencies can carry forensic insights. However, currently, there are no systematic methodology to precisely identify them. Under these circumstances, the EM radiation frequency that is always available is the clock frequency of the IoT device's MCU chip [90]. For instance, in the case of Raspberry Pi device, this frequency is 1.4 GHz. In Arduino Leonardo device, the clock frequency is 16 MHz. In some cases, external EM noise from various other sources can exists in the same frequency; causing interference in clearly observing the EM radiation coming from a particular IoT device. When this happens, a higher harmonic frequency of the clock frequency , i.e., a multiple of the original frequency, can

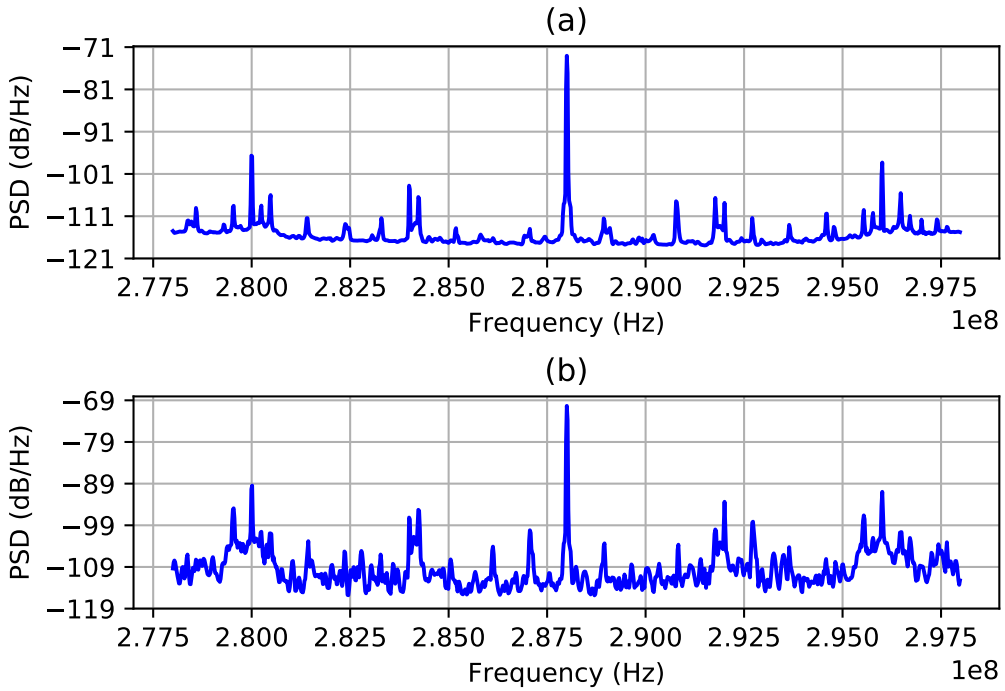

Figure 4.14: PSD of the EM radiation when running Bubble sort algorithm with two different antenna positions on Arduino.

be used as the information-leaking frequency. For instance, Arduino's 16 MHz EM radiation can overlap with shortwave radio transmissions. Therefore, it was empirically identified that the 18^{th} harmonic of the clock frequency, i.e., 288 MHz, is most appropriate to observe EM radiation of the Arduino device (see Figure 4.13).

Source Location of the Signal: The next important decision that is necessary to observe EM radiation is the precise location on the IoT device from where the radiation is coming from. The H-loop antenna has to be placed as close as possible to that location in order to maximise EM signal reception. Figure 4.14 illustrates two observations of the EM signal from the Arduino device by placing the H-loop antenna over two different regions of the processor. As it is evident, the antenna position corresponding to Figure 4.14(b) captures a stronger signal. Therefore, when attempting to identify the most suitable signal reception position, the H-loop antenna should be moved across the target IoT device while keep it as closer to the device surface as possible. The

reception signal should be visualised in real-time, e.g., as an FFT or a spectrogram, to identify the signal variation with the antenna movement. At the end of this empirical assessment, the positioning of the antenna that produced the clearest EM signal observation should be noted and fixed for that particular IoT device type.

Sample Rate of Data Acquisition: As described in Section 2.4, the sample rate and bandwidth of SDR hardware are equal. Therefore, setting the sample rate to a particular value has multiple consequences to the EM signal acquisition process. Firstly, higher sample rates provides more information about the EM signal, while lower sample rates can lose information. Secondly, a higher sample rate increases the bandwidth of the captured signal around its centre frequency. In most cases, multiple information-leaking frequencies occur surrounding the IoT device's clock frequency. The use of a wider bandwidth helps to capture them. Under these circumstances, using as high a sample rate as possible is advantageous. The only limiting factor of the use of fast sample rates is the computational overhead it can cause to further EM data processing stages. The HackRF SDR device used in this work facilitates sample rates up to 20 MHz. Therefore, this highest sample rate was used on the experimental evaluations throughout this thesis.

Duration of Data Acquisition: When observing the EM radiation of an IoT device, the time period of data acquisition has to enclose the internal activity of interest of the IoT device. However, due to the repetitive nature of most IoT firmware, i.e., doing the same activity again and again, it is rarely needed to precisely time the sampling duration to capture a specific internal activity of an IoT device. Therefore, a sufficiently long EM data observation can enclose all the necessary information the IoT device is leaking. Meanwhile, longer sampling duration produces a larger EM trace, causing difficulties to the handling and processing of EM data in later stages. All these concerns point to the need of a carefully set sampling duration, which is not too long and not too short. In the experimental evaluations of this work, EM traces were acquired with varying sampling time periods below 60 s depending on the experimental requirements in each case.

4.7 Experimental Plan

Having presented the design and implementation details of the *EMvidence* framework, the next two chapters explore the specific methods that are necessary to build plug-ins.

4.7.1 Designing Methods to Acquire Forensic Insights

The capability of an *EMvidence* plug-in to extract forensically-useful insights depends on the successful application of ML methods. While there are various ML algorithms that have been published in the literature, the choice of a suitable ML algorithm for a plug-in can vary in different investigative contexts. The inspection of EM radiation and the identification of known patterns that are correlated to specific information of IoT firmware is a classification problem. Chapter 5 develops and evaluates the ML methods that are suitable for such purposes.

4.7.2 Designing Methods to Increase Efficiency

The EM radiation data of a DUT captured using SDR hardware typically consist of a wide bandwidth and sample rate. Therefore, such data are typically highly dimensional. Large dimensionality of data is a challenge to the classification tasks with ML algorithms. Some ML algorithms may fail completely in the face of hugely dimensional data, while some other may demand unreasonable amounts of computational resources to produce predictions within a reasonable time period. This affects the practical usability of plug-ins in *EMvidence* in investigative scenarios. Chapter 6 explores a multitude of methods to increase the efficiency of *EMvidence* plug-ins by addressing the dimensionality problem in EM data.

Chapter 5

Insights from Waves: Machine Learning Methods for EMvidence

5.1 Introduction

The goal of this chapter is to evaluate ML methods for building *EMvidence* plug-ins to acquire potential forensically-useful insights from IoT devices through their EM radiation. IoT devices that are in wide use in day-to-day life comes in all shapes and sizes. While each IoT device can consist of a unique combination of hardware and software, they can broadly be categorised into two classes based on their computing resources as *low-end* and *high-end* IoT devices [168]. The design choice of choosing computing hardware resources to be included in an IoT device depends on multiple reasons. Among them, source of power is an important factor. A device that can be continuously mains powered can have more powerful hardware, and therefore, can be considered as a high-end IoT device. Meanwhile, a device that has to rely on a battery for a prolonged period of time needs to use an energy efficient processor such as MCU or SoC. Hence, such devices can be considered as low-end IoT devices and include health implants, fitness wearable devices, and smart light bulbs. Due to limited on-board storage, these devices generally do not store much data. They tend to either transmit data into an associated smartphone app or hub, or directly to a cloud service. Therefore, even if a low-end

IoT device contains some non-volatile data storage, e.g., an SD card, it is less likely that traditional digital evidence extraction methods would prove fruitful.

Meffert et al. highlighted the need for identifying the running forensic state of IoT devices in an investigation [172]. A forensic state is the state of hardware and software of an IoT device at the time it was seized by law enforcement. For example, an IoT smart lock can have two states, i.e., locked and unlocked, and its state could be a vital information in an investigation. Furthermore, IoT devices that are subject to investigation may have been tampered with intentionally or as a result of malware. Ronen et al. demonstrated that IoT smart bulbs can be infected Over-the-Air (OTA) and be controlled remotely [173]. Such maliciously modified devices are shown to be effective in causing harmful results to humans, such as creating epileptic seizures to vulnerable individuals by adjusting the frequency of LED smart bulbs [174]. Therefore, verifying whether the device is running its default firmware can useful. In case the device has been reprogrammed and protected with encryption, identifying the encryption algorithm is also forensically useful.

The rest of this chapter is organised as follows. Although a multitude of experimental evaluations have been performed, they have a common objective and uses a similar hardware and software setup in most cases. Section 5.2 introduces the experimental procedure where common details across all the experiments are described. However, certain specific details to individual experiments are left to be mentioned under each experiment later. The experiments and the results obtained are laid out in Section 5.3. Finally, Section 5.4 discusses the implications of these results and how they are integrated into the *EMvidence* framework.

5.2 Considerations for Experiments

EM-SCA attacks require a sufficient number of target device EM traces. Furthermore, in order to train ML models using EM traces, it is necessary to have EM trace samples annotated with the specific software activities of the target device they represent. Three hardware components are necessary for the ac-

quisition of EM traces. These are namely; a DUT, a signal capturing device, and a host computer. The signal capturing device is connected to the host computer via USB interface while the DUT may or may not be connected in a similar manner. The host computer runs the *EMvidence* framework and stores the captured EM traces for analysis.

5.2.1 Types of Useful Insights

The number of potential insights from IoT devices that may help to progress forensic investigations can be infinite [15]. Experimentally evaluating all of them is not practically possible. Due to this reason, it is necessary to define a sufficiently inclusive set of insights that may prove the point and provide the confidence for future research. With this goal in mind, the following insights were focused to experimentally evaluate in this chapter.

- **Cryptography-related Events**: According to the requirements of computational resources and the level of security, the choice of cryptographic algorithms employed on an IoT device may vary. The possibility to inspect the storage of a device using classical forensic approach depends on whether the device in question employs data encryption.
- **Detection of the Firmware Version**: The firmware of IoT devices can receive software updates either automatically or by manual actions of the user. These updates bring bug fixes and patches to cover security vulnerabilities. Therefore, IoT devices of the same type may run different versions. In order to decide further moves to inspect an IoT device, the investigators require sufficient knowledge about the firmware version running on it.
- **Malicious Firmware Modification**: A compromised IoT device for a malicious purpose is not identifiable by the visual inspection. Such compromised devices may be relevant to an ongoing investigation, however, may go unnoticed. The inspection of such a device by other means, such as analysing network activities, may even mislead the investiga-

tors. Therefore, the identification of potential tampering to firmware is important.

- **Current Behavioural State**: When an IoT device is discovered during an investigation, there is a high chance for it to be alive and running. The running firmware of an IoT device can be in one of a few states. This is because, most IoT devices are designed to perform a limited set of tasks that will be executed in a state machine-like manner. The identification of the current internal state of a running device at the location where the device was found can help investigators to make informed decisions.

5.2.2 Machine Learning Algorithms

While there are various ML algorithms available for similar purposes, it is not our objective to evaluate which algorithms perform best in gathering forensically-useful insights in which scenarios. Instead, our requirement in this chapter is to test and demonstrate that the intended goal is achievable with ML. Towards this goal, the accuracy of the results achieved through ML algorithms in general is the key focus. The efficiency of training, testing and validation of the results with a particular ML algorithm is a secondary concern. Therefore, we have the freedom of choosing any ML algorithm as far as it facilitates this objective.

Mapping the pattern of EM radiation signals into known internal behaviours of IoT devices is a classification problem from the perspective of ML. This work uses neural network-based classifiers that provide a great flexibility by tuning a multitude of parameters. In its most basic form, a neural network is called MLP [175] where the information flow moves from the input layer towards the output layer across one or more hidden layers. In addition to MLP, this work makes use of LSTM architecture of neural networks as well [176]. An LSTM network consists of feedback connections that allows the network to look at sequences of data points at once instead of individual data points in classical neural networks. Therefore, it is more suitable at identifying patterns that occur in time series data, such as EM traces [177]. In addition to neural networks,

this work also uses SVM in binary classification problems due to the simplicity of such scenarios [178].

5.2.3 Preprocessing Procedure

An EM trace is a vector that represents the amplitude variation of a signal over time. Due to fast sample rates used by signal acquisition hardware, an EM trace with a duration of milliseconds can contain millions of data points. When these data are used directly as input to train and test ML models, the highly dimensional data can negatively affect time and amount of computing resources they demand. As a result, EM traces acquired through the aforementioned hardware setup are not suitable to be directly used to train ML models. Therefore, EM traces need to be preprocessed in order to transform them from a continuous time domain signal into a format that has a manageable feature vector for ML models.

When attempting to classify software activity EM traces, labelled EM traces are needed. For this purpose, EM trace samples are acquired by running each software activity on the target device and collecting EM traces of a predefined length. Originally, each EM trace is in time domain. Time-domain signals are prone to fluctuations caused by external noise. Therefore, each trace is transferred to frequency domain using FFT with an overlapping sliding window [177]. For each EM trace, this results in a collection of FFT vectors representing consecutive time steps. The dimensions of the FFT vectors are still considerably higher to be directly used as the feature vector for ML classifier, e.g., 200,000 dimensions. Therefore, the dimensions of these FFT vectors are further reduced by dividing the elements of each FFT vector into 1,000 equally spaced bins. From each bin, the maximum element is selected as the representative of the bin without losing the generalisation. This results in a 1,000 element long feature vector for each time step of EM traces.

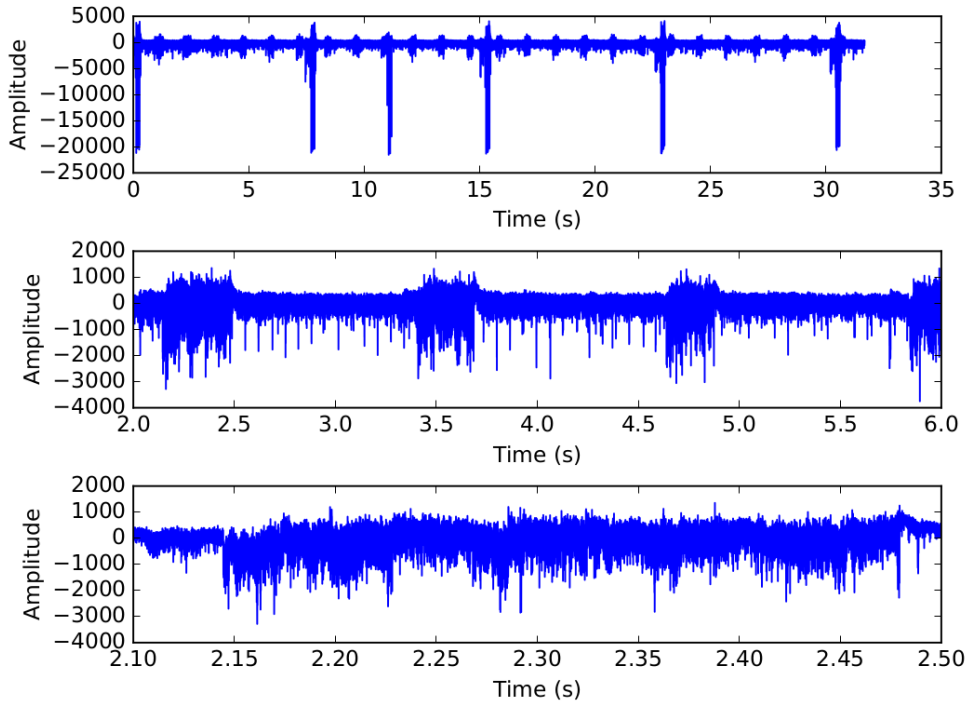

Figure 5.1: Waveform of the AM demodulated signal at the CPU clock frequency of Raspberry Pi. The AM modulated signal represents the AES encryption performed on the device. Sudden higher peaks are an external interference signal coming from an unknown source. (The three sub-figures depict three zoomed-in scales of the same signal.)

5.3 Experimental Evaluation

5.3.1 The Cryptographic Activities of High-end Internet of Things Devices

As previously shown in the literature [80, 81], EM signals coming from the CPU modulate the software behaviour on its amplitude. In order to observe such variations, the following experiment was performed. A Raspberry Pi device was programmed to run a shell script that performed AES encryption operations with a time gap of 1 s. The shell script used OpenSSL commands to invoke the AES-256-CBC algorithm on a large file continuously. The AES op-

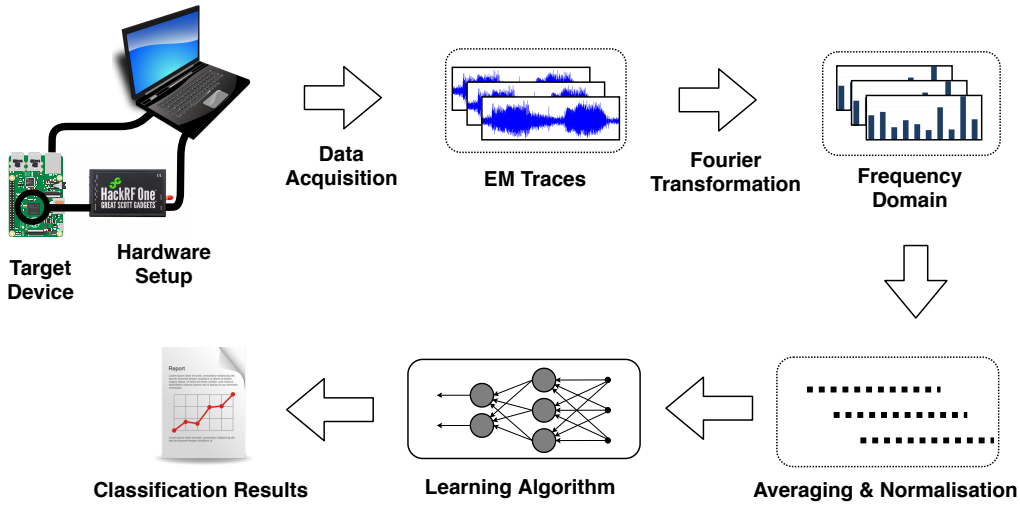

Figure 5.2: The EM trace acquisition and preprocessing stages in order to classify cryptographic activities using an ML model.

erations performed periodically on the Raspberry Pi resulted in observations of amplitude variations in the EM signal, as illustrated in Figure 5.1. The blobs that occur with a 1 s gap in the first sub-figure correspond to the AES encryption operations, while the much higher peaks that occur at irregular intervals are external noise. A selected region of the signal is zoomed-in in the second sub-figure. The third sub-figure illustrates the EM radiation pattern of a single AES encryption operation.

Among the software activities of IoT devices that may have a forensic interest, cryptographic operations are at the forefront [31]. When storing data on-board or transmitting over the network, modern high-end IoT devices tend to rely on cryptographic encryption as a security measure. The following experiment investigates the possibility of using EM radiation of IoT devices in order to automatically detect when they perform data encryption operations. Three major cryptographic algorithms, i.e., AES-128, AES-256, and 3-DES are used, as three classes and a mixture of non-cryptographic operations is used as another class. Figure 5.2 illustrates the procedure of acquiring data, preprocessing data, and finally the classification to classes.

Data Acquisition: The same hardware configuration with a Raspberry Pi as the target device and a HackRF as the EM signal capturing device are used

in this experiment. In order to train a classifier to detect cryptographic operations, a labeled EM trace set is necessary for each classification class. To assist in this task, a UDP communication channel between the Raspberry Pi and the host computer was established through the Ethernet cable. To reduce unnecessary EM noise capture, each time the Raspberry Pi perform a cryptographic operation, it notified the host computer immediately before and after by sending UDP packets. This allows the host computer to identify the time period of the EM data stream coming from the HackRF, that corresponds to the cryptographic operation. Future work will focus on automatically identifying the necessary radiation segments through a sliding window, eliminating this step. Each such identified EM signal segments are saved as an EM trace along with the label of the cryptographic algorithm. Overall, the EM traces collected was about 12 GB.

Data Preprocessing: Due to multiple factors, the acquired EM can have variable lengths in the time-domain and also may not properly enclose the cryptographic operation within its boundary. These reasons include the inherent difference of the time each cryptographic calculation takes to execute, the delays in UDP communication between the Raspberry Pi and the host computer, and the delays in the HackRF data acquisition software to start and stop the EM sampling. Due to the large length and the variability of the lengths, these labeled EM trace samples are still unsuitable to be directly used as training samples for a ML-based classifier. To mitigate these differences in EM traces, each trace is converted into the frequency-domain by using a *Fourier Transformation*. This is achieved by calculating STFT over a specified window of samples over the EM traces [35].

Setting the STFT window size to be too large can cause the resulting frequency-domain vector to be extremely large. Similarly, setting the STFT window to be too small can cause the loss of information. In order to have a balance between the two ends, the STFT window size was empirically set to 0.1 s after testing with different window sizes. Since the sample rate of the HackRF is 20 MHz, the resulting Fourier Transform contained a vector with 200,000 elements; each containing the amplitude of a frequency component of the original EM radiation. In this Fourier Transform vector, it was observed

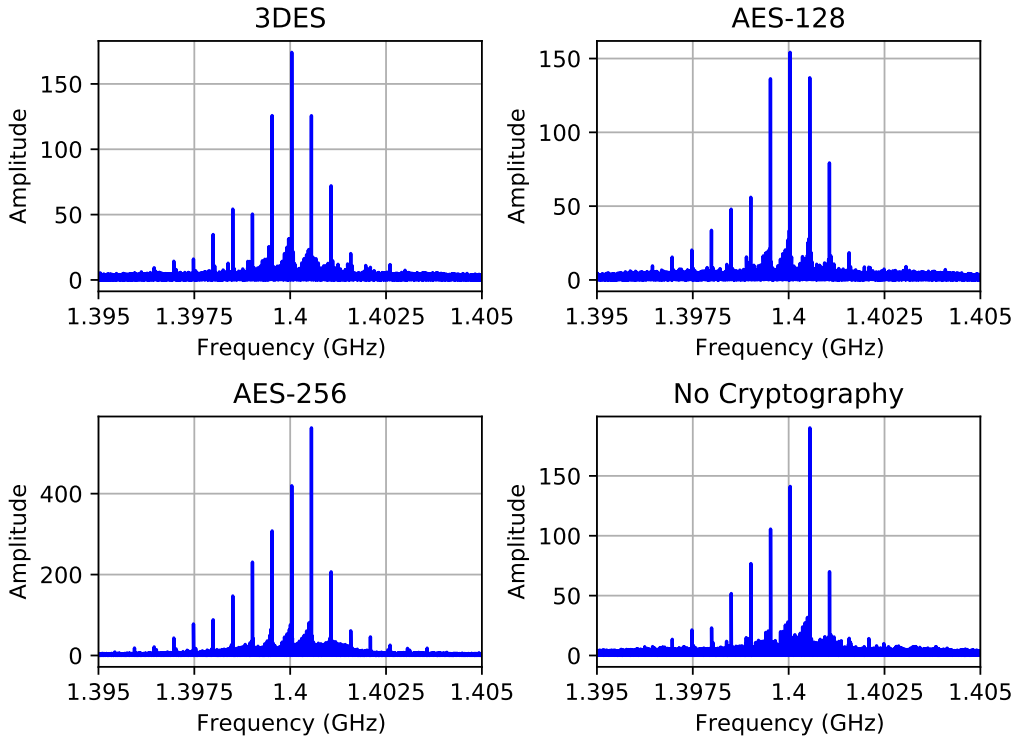

Figure 5.3: Sample Fourier Transform vectors of cryptographic algorithms that run on Raspberry Pi.

that the variation of peaks from software activity was only distinguishable in the middle portion. Therefore, it was decided to use only the frequency components from ¼ to ¾ of the original Fourier Transform through discarding the edges. Figure 5.3 illustrates samples of Fourier Transforms from each class, where it is evident that there are slight variations unique to each activity.

The number of elements in the Fourier Transform was too large to be directly taken as an input vector for modelling. The dimensions can be reduced by breaking the Fourier Transform vector into a limited number of bins. Subsequently, a representative value can be selected for each bin by averaging the values or selecting the maximum valued element in each bin. In this particular experiment, 500 bins were selected where the elements within each bin were averaged to generate feature vector of 500 features. The number of bins and feature vectors were decided through experimentation and evaluation of the produced ML classification models.

Table 5.1: Classification accuracy of cryptographic algorithms.

Activity	Precision	Recall	F1-Score
Other	0.93	0.85	0.89
AES-256	0.78	0.86	0.82
AES-128	0.99	0.92	0.95
3DES	0.81	0.85	0.83

Classification: A neural network was implemented to classify EM traces into the correct class that had four layers; an input layer, two hidden layers, and an output layer. The number of hidden layers and the number of hidden nodes used in each of the hidden layers were decided empirically by evaluating various settings. Accordingly, the first hidden layer was assigned 10 hidden nodes, while the second hidden layer was assigned 5 hidden nodes. The input layer has 500 input nodes for the feature vector and the output layer has 4 nodes for the four classes. For each class, 600 samples were taken for the training process totalling 2,400 training samples for all four classes. The learning rate of the neural network was set to 1^{-20}, which was decided empirically. The classifier code was running on a computer with 64 bit Intel Core i-5 quad-core processor and 16 GB memory, running a Linux operating system. While the EM traces acquisition and preprocessing to generate training samples took several hours, the training and testing phases of the neural network took less than a minute to provide classification results. A 10-fold cross-validation was used to validate the classification results.

Results: The result of the classification is illustrated in Table 5.1. The neural network classifier correctly classified the three cryptographic algorithms and the non-cryptographic scenarios with over 80% accurately. Considering the fact that Raspberry Pi was running a computationally heavy operating system like Linux, which can make use of all four cores of the processor for multi-tasking, the ability to distinguish between these three major encryption algorithm settings hints that it should be possible to detect cryptographic algorithms on much less capable hardware devices. Existing cryptographic key recovery attacks depend on prior knowledge of the cryptographic algorithm being employed. The ability to identify the cryptographic algorithm solely

based on EM observations can increase the likelihood of success for such key recovery attacks.

5.3.2 The Cryptographic Activities of Low-end Internet of Things Devices

Some cryptographic algorithms, e.g., RSA, demand reasonably high computational power making them unsuitable for low-powered computing devices. As a result, ECC has increasingly been deployed on these devices [110]. ECC is a public key cryptography that requires a smaller key length compared to RSA. Numerous different elliptic curves can be used in ECC. Table 5.2 illustrates five major elliptic curves designed to run on low power devices (available in the micro-ecc library). These elliptic curves use different private and public key lengths with bespoke configuration settings.

Due to the differences in settings of each elliptic curve, the running time of ECC operations for each curve can vary. Figure 5.4 illustrates the average time different elliptic curves take to digitally sign a message and to verify the signature of a message (ECDSA). The measurements were calculated by running each ECDSA algorithm on an Arduino device with equally sized messages. For each curve, both signing and signature verifying operations take almost the same amount of average time. However, the average time taken by individual curves are distinguishably different from each other.

In order to investigate the possibility of detecting the presence of ECC cryptography through EM radiation, an experiment was conducted with an LSTM binary classifier. The objective is to train a model to distinguish between ECC

Table 5.2: Private and public key sizes of ECC curves.

Curve	Private Key (bytes)	Public Key (bytes)
secp160r1	21	40
secp192r1	24	48
secp224r1	28	56
secp256r1	32	64
secp256k1	32	64

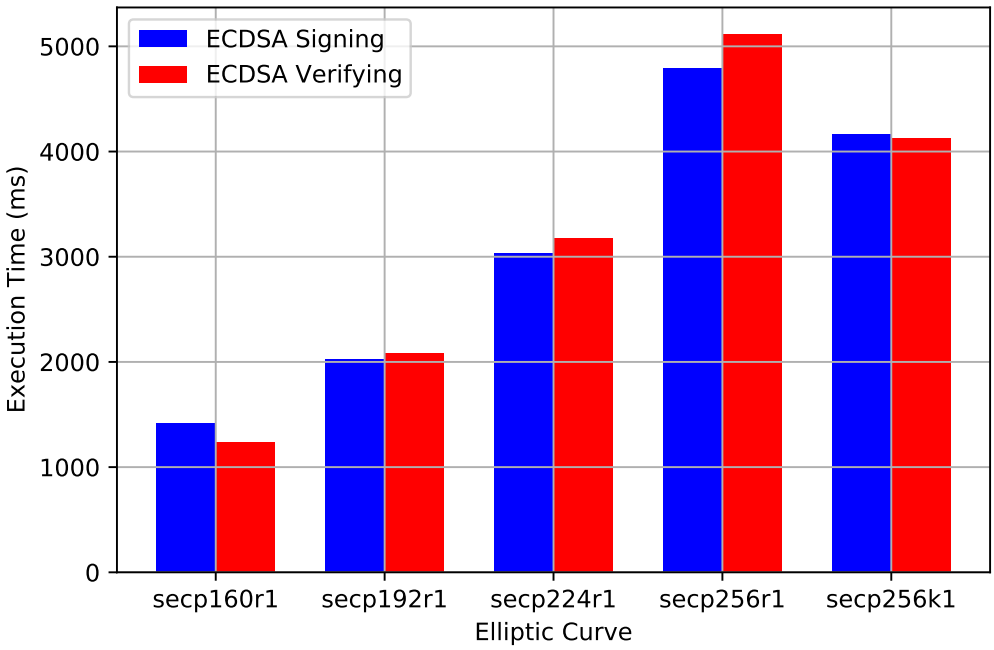

Figure 5.4: The time it takes to digitally sign and verify a message using different ECC curves on an Arduino device.

cryptography operations and other software activities. For this purpose, an Arduino device was programmed to perform ECDSA signing operations that was controlled by sending commands through USB from the host computer. Each time an ECDSA signing operation is performed, a 100 ms trace was captured through a H-loop antenna placed over the Arduino's MCU connected to a HackRF SDR.

For each of the five elliptic curves, 50 EM traces were acquired (totalling 250 traces) for the ECC cryptography class. For non-ECC cryptography operations class, 20 Arduino programs used that have varying complexities. From each Arduino program, 12 traces were acquired for non-ECC cryptography class (totalling 240 traces). A sliding window of 10 ms was used with a 2 ms step size (80% overlap) to collect segments from each trace and subsequently a feature vector for each window segment was calculated using FFT broken down into 1,000 equally spaced bins. Each bin's maximum amplitude frequency component was selected as the representative signal for the bin. This

results in training sequences each having time steps where each time step consists of 1,000 features. All the training samples were normalised to values between 0 and 1. Figure 5.5 illustrates examples of signals acquired from a DUT while running two curves of ECC and running two non-ECC programs.

The LSTM classifier was implemented using Keras library with Python. A single LSTM layer consists of 100 nodes and a fully connected layer with 1 node for binary classification. This last node uses a *sigmoid* activation function, while the model uses *binary crossentropy* as the loss function. The sequence dataset was broken into 75% and 25% sets for training and testing purposes respectively. When using 5 epochs and 64 batch size, the LSTM classifier achieved an impressive 100% accuracy. In order to assess the effect from the sliding window length and EM trace length, further LSTM models were trained and tested varying those parameters. Figure 5.6 illustrates the variation of classification accuracy against both sliding window length and EM trace length. As is evident, the longer the EM traces, the higher the classification accuracy achieved. This is due to the fact that longer EM traces results

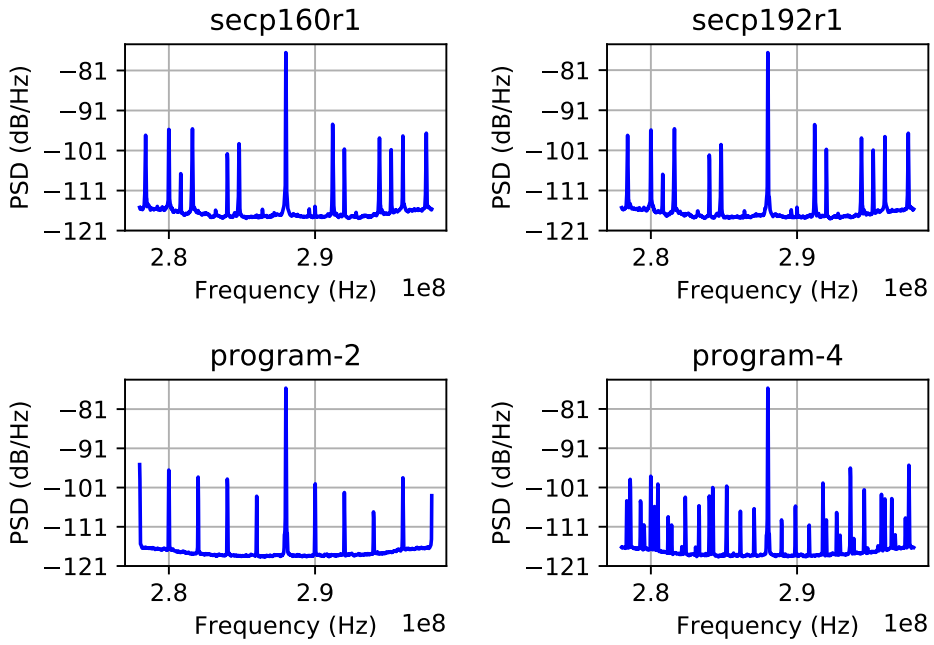

Figure 5.5: Example ECC and non-ECC signals acquired from Arduino device.

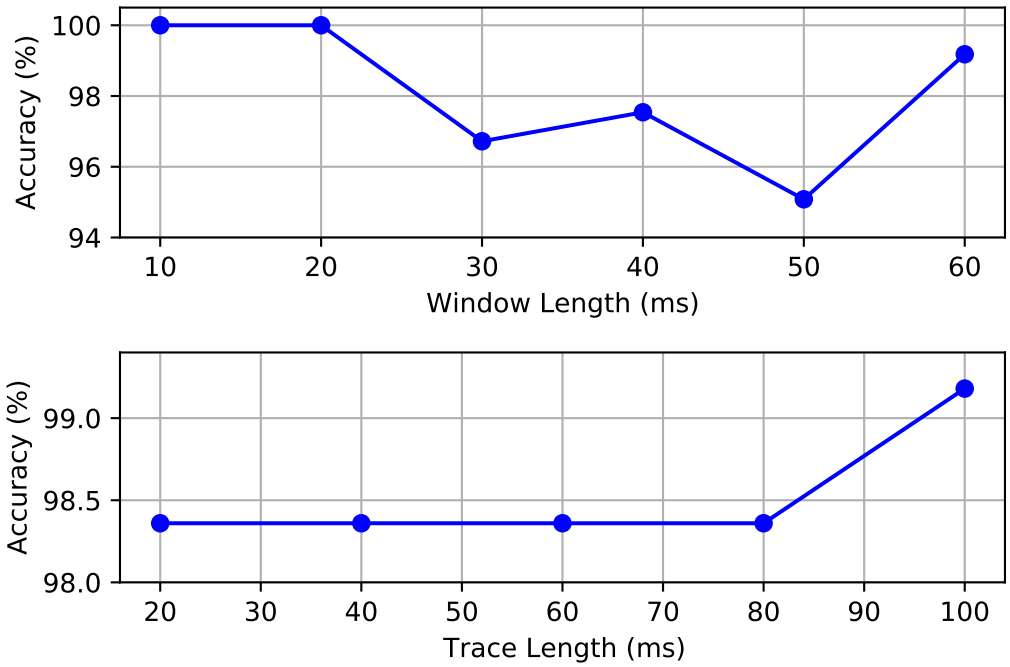

Figure 5.6: Variation of classification accuracy against sliding window length and EM trace length.

in longer sequences with more information for the LSTM model to learn. In contrast, longer sliding window lengths negatively affected the classification accuracy, reducing it to 95% in the worst case. Again, the reason is behind the length of the sequences. Longer windows result in shorter sequences for a fixed length of EM traces.

5.3.3 Firmware Version of Internet of Things Devices

While heavy cryptographic algorithms are employed on resource rich IoT devices, simpler devices are unable to use such computationally heavy algorithms to encrypt data due to the lack of computational resources. Therefore, they are usually programmed to perform a repetitive task continuously. Among the various tasks performed by IoT devices, certain tasks have forensic interest. These include reading data from a specific on-board sensors, such as a microphone, writing data to an on-board storage device, such as an SD card,

```
/* Arduino test program */
void setup(){
}
void loop(){
    for(int i=0, i<20, i++) { delay(10); }
    for(int i=0, i<20, i++) { delay(10); }
    /* further loops */
}
```

Code Snippet 1: An example Arduino program that performs a time complexity $O(n)$ task repetitively inside an infinite loop that is used as a classification target.

and executing a command received remotely through the network. Identifying what operations an IoT device is performing at the moment when it was seized live could prove important. For example, if the device is currently wiping the SD card according to a command received remotely, the investigators need to know it immediately so that they can turn the device off without waiting for any further live analysis.

To explore the possibility of distinguishing different tasks performed by a simple IoT device, the following experiment was carried out. The objective was to train and test a ML model that can classify simple IoT firmware with increasing complexity. It was decided to use an Arduino device for this experiment as its simpler processor matches the resource profile of a lower-end IoT device. In order for classification, ten Arduino programs were selected that repeatedly perform a task inside an infinite loop. Code Snippet 1 illustrates an example Arduino program used as a classification target. As can be seen, the program consists of an infinite loop designed to represent a repetitive task of an IoT device with a time complexity of $O(n)$. Each subtask the device is performing is represented by individual `for` loops with a finite number of iterations. It is assumed that a malicious modification to the device is performed by adding a new subtask to the program or by removing an existing subtask from the program [96].

Data Acquisition: In order to collect EM trace samples for each program, the Arduino was programmed with them separately and allowed to run with a

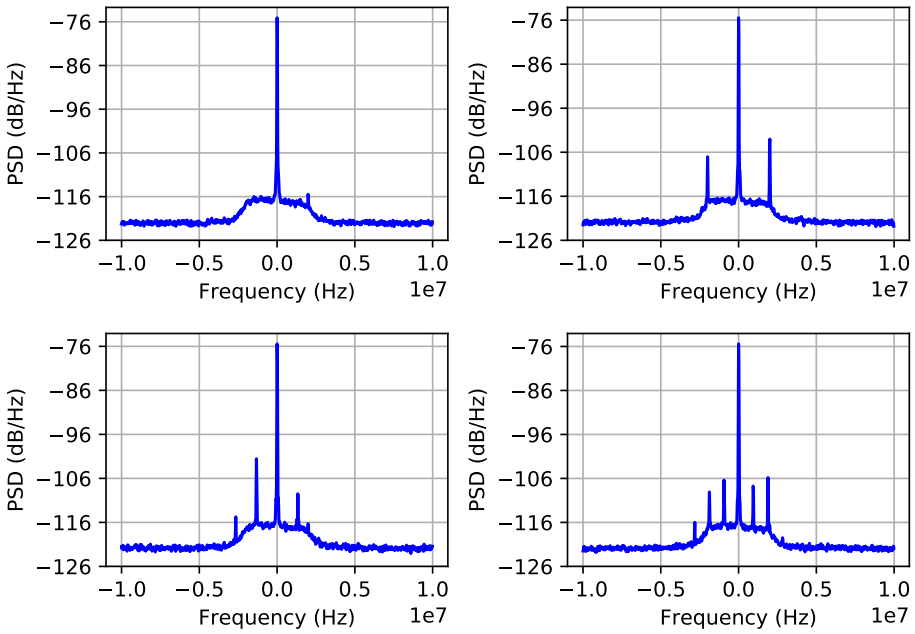

Figure 5.7: Power spectral density (PSD) of the EM radiation from four different Arduino programs that were used for classification.

H-loop antenna placed approximately 1 cm above the MCU of the device. The HackRF was tuned to the information leaking 288 MHz frequency of the target device and sampled data at the rate of 20 MHz. Since the target device was performing a repetitive task, there was no software instrumentation required. Each acquired EM trace was approximately 25 ms long. Since there were ten programs to detect, 600 EM traces were acquired per class, which resulted in 177 GB of data for the overall 6,000 EM traces. Figure 5.7 illustrates the PSD of the EM radiation of four such programs subject to the experiment.

Data Preprocessing: From the extracted EM traces of each program class, 10 ms long segments were extracted and converted to the frequency domain using FFT. Unlike the aforementioned scenario of classifying between 4 cryptographic classes, this experiment attempts to classify 10 different programs. A 500 element feature vector did not seem to be effective in this case. Therefore, it was empirically decided to create a feature vector of 1,000 fea-

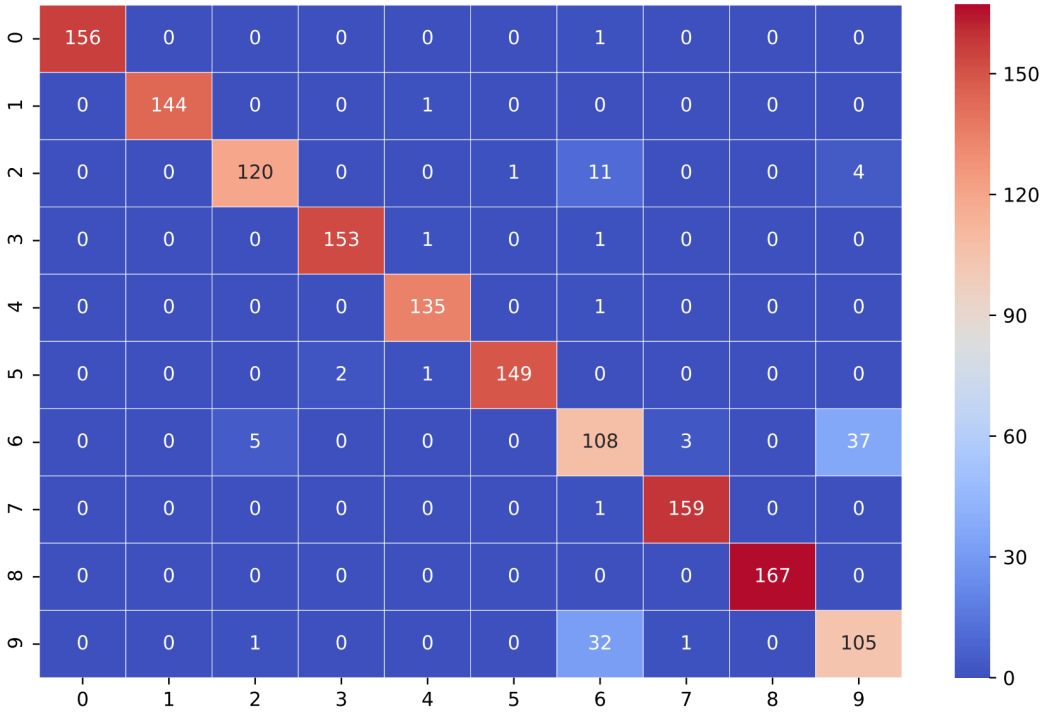

Figure 5.8: Confusion matrix of the neural network classifier to detect ten different Arduino programs, which are labelled from 0 to 9.

tures by breaking a Fourier Transform vector into 1,000 bins. Furthermore, it was noticed that averaging values within a bin smoothed out the most significant frequency component under the noise floor. This most significant frequency ideally would have been selected as the representative element for the bin. Therefore, it was decided to select the maximum value within each bin instead of averaging in order to build the feature vector.

Although the EM trace data were acquired while the target device was running in a noisy environment, there was no noise filtering applied to the EM traces before generating the feature vectors. The choice of the information leaking 18^{th} harmonic of the Arduino clock frequency was made to ensure no strong external noise source in that frequency.

Classification: Similar to the previous experiment, a neural network with two hidden layers was designed, where first hidden layer contained 10 hidden nodes while the second hidden layer contained 3 hidden nodes. The input layer contained 1,000 features and the output layer contains 10 output nodes.

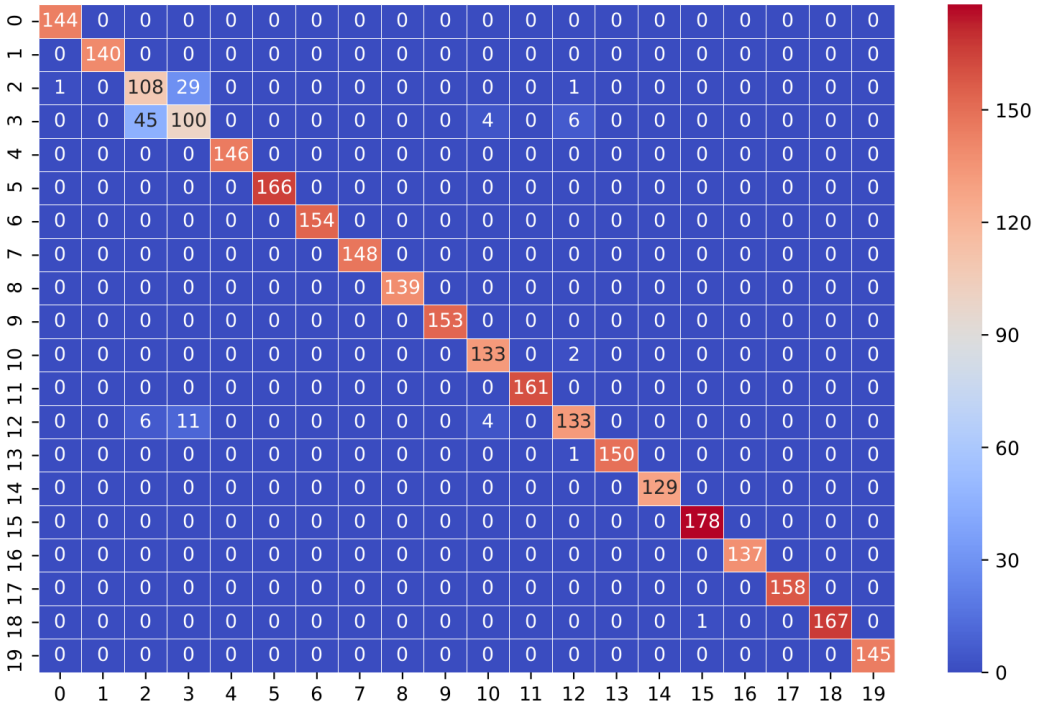

Figure 5.9: Confusion matrix of the neural network classifier to detect twenty different Arduino programs, which are labelled from 0 to 19.

With 600 training samples for each class, a total of 6,000 training samples were fed to the neural network to train and test the model to detect ten Arduino programs running on the target device.

Results: Figure 5.8 illustrates the confusion matrix of the classification results. The programs subject to the experiment are labelled from 0 to 9 in the figure. As can be seen, the majority of the Arduino programs were detected by the classifier accurately. Under a 10-fold cross-validation, the classifier achieved a mean classification accuracy of 90% for an error margin of 11% within a 95% confidence interval. Considering the fact that currently it is nearly impossible to identify the software activities of an IoT device without a significant support from the manufacturer, the achieved accuracy through EM-SCA can potentially be a significant benefit to an investigator to gain insight on the device. Later, this experiment was repeated with 20 different Arduino programs that follow the same source code format. Unlike in the previous case, the MLP classifier was set to have 3 hidden layers with 100, 50, and 30 hidden

nodes respectively. The classifier achieved a 96% accuracy with this setup (see Figure 5.9).

5.3.4 Malicious Modifications to the Firmware of Internet of Things

A very simple IoT device with a 8-bit processor and few kilobytes of memory is only capable of running a simple firmware that can perform a simple and repetitive task. The firmware running on such IoT devices are easier to be replaced by attackers in order to make them run malicious code. A device with a modified firmware can cause malfunctions not intended by the manufacturer. For example, Mirai is a malware that infected certain types of IoT devices through exploiting their unchanged factory default passwords [179]. It enabled the infected IoT devices to take part in distributed DoS attacks without the knowledge of device owner. Therefore, detecting such modifications to the stock firmware of an IoT device is highly necessary. When the EM radiation signature of a target device is already known, any change to the default firmware of the device should cause a detectable change to the EM radiation pattern. Therefore, it is possible to train a ML model to recognise anomalous EM radiation patterns due to firmware changes.

When detecting anomalies using ML models, there are two potential directions; namely outlier detection and novelty detection [180]. In the outlier detection, an unsupervised approach is taken where both legitimate data and anomalous data are provided to the ML model. The model fits into the legitimate data with the assumption that this data are densely packed in the space while anomalies stay comparatively away. In contrast, novelty detection is a semi-supervised approach where only the legitimate data samples are provided to the model to train. Whenever new data samples are provided, the model assesses the likeness of the new data to the data it was trained on in order to determine whether the new data belongs to the same distribution or not.

Since there are infinite possibilities for modification to the default firmware of an IoT device, it is difficult to provide sufficiently representative set of sam-

ples of anomalies for a ML model to learn. Therefore, in this case, the semi-supervised novelty detection by training a model with only the legitimate samples is decided the best technique. In this experiment, a one-class SVM with a non-linear kernel (RBF) provided by the Scikit-Learn library was used for this purpose [61]. When training the model, one of the Arduino programs used in the aforementioned software behaviour detection experiment was used as the legitimate firmware of the device, while a mixture of other programs were used as the modified programs. The model was trained by providing 500 training samples of the legitimate program produced during the previous experiment. For testing, 100 further samples of the legitimate program was provided where the testing error rate was 18%. Finally, when 20 different modified Arduino programs were provided for validation, each of them were detected by the model recording a 100% accuracy on anomalous program detection.

5.3.5 Current Behavioural State of an Internet of Things Device

In order to illustrate the usefulness of detecting forensic state of low-end IoT devices through EM-SCA, the following hypothetical scenario was considered. An IoT device has been deployed in a building as a part of an intruder detection system. The device consists of a sensor that detects movements within a specified space of the premises. The device consists of two actuators, an alarm and a door lock, that it can control independently. Furthermore, the device is connected to a GSM module in order to send and receive SMS. The device firmware is programmed to continuously read the motion sensor to detect intrusions into the premises. Upon detection, it can perform one of three tasks – locking the door, firing an alarm, or sending a text message to the owner. At any time, the device can be disabled by pressing a physical button that puts the device into an idle state. The device does not have any other associated network servers that can log device states. Furthermore, the device does not switch internal states due to any other reason than the specified ones. The five states of the device that we are interested in are namely; (1) idle, (2) read digital sensor (reading motion sensor), (3) control

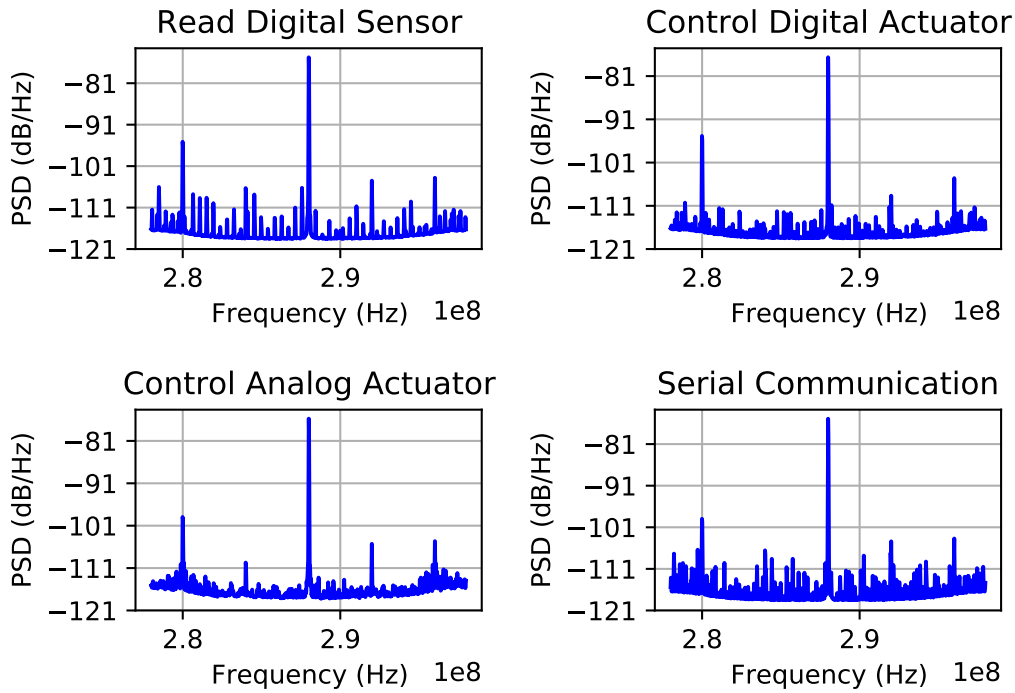

Figure 5.10: The PSD plots of the IoT device's EM signal at different device states.

digital actuator (firing the alarm), (4) control analogue actuator (turning door lock), and (5) serial communication (sending text message).

Suppose that this building is subject to a legal investigation for a crime assumed to be conducted by an intruder. Upon the arrival of the law enforcement officers, one of the investigative questions that arise is whether the intruder detection system functioned as expected or did an insider disable it before the crime occurred. The answer to this question can be found if the current internal state of the device is known. Turning the device off and moving it to a digital forensics laboratory destroys the current internal state of the device. Performing a live EM-SCA on the intrusion detection device and identifying its current software state may be the only viable approach to resolve this problem.

The IoT device was emulated by using an Arduino device. It was programmed to run a software code that puts the device on each of the 5 states chosen by the user. While the device was running on each state, a 30 s long EM trace was captured per state with a sample rate of 20 MHz using HackRF

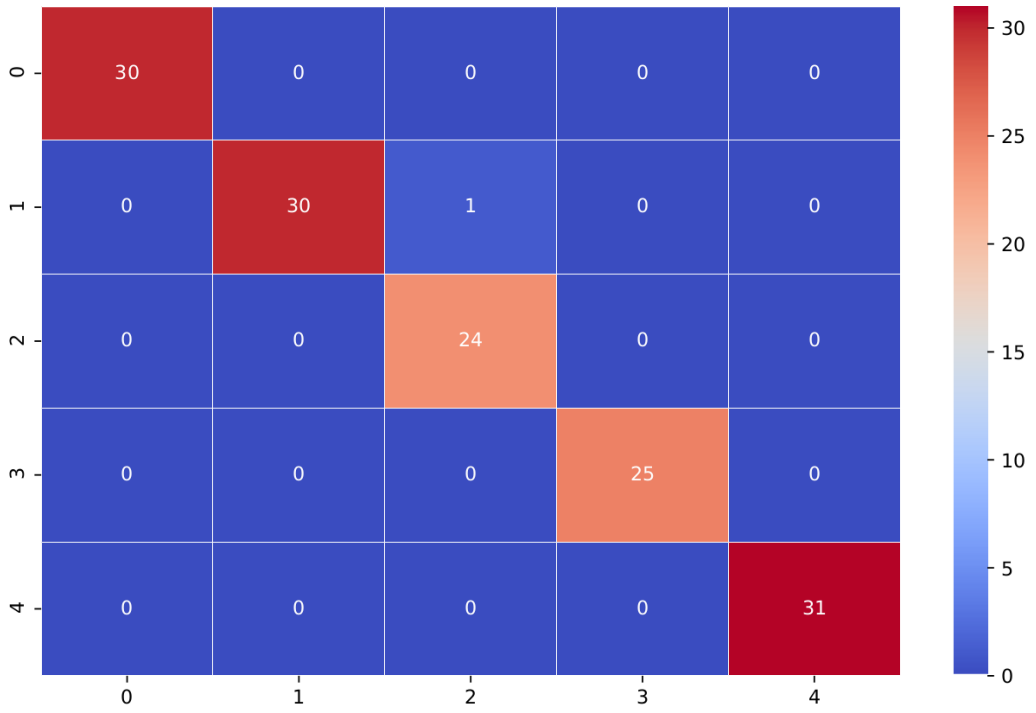

Figure 5.11: Confusion matrix of the IoT device state classifier.

SDR. The SDR was tuned into the 18th harmonic of the Arduino's clock frequency, i.e., 288 Mz. The H-loop antenna of the SDR was placed over the processor of the device during data acquisition. Figure 5.10 illustrates the PSD of the EM signals from IoT device in its different states. A neural network classifier based on MLP architecture was selected to distinguish each state of the IoT device. A non-overlapping sliding window with a width of 250 ms was used to extract EM trace segments; subsequently converted to the frequency domain. These windows were divided into bins and averaged within each bin to produce a vector of 1,000 features that were considered as training and testing samples for the ML classifier.

Figure 5.11 illustrates the confusion matrix of the classification results. The classifier was able to achieve an average F1-score of 99% in distinguishing the 5 IoT device states. This indicates that a pre-trained model to identify internal software states of the IoT device. For example, if it was identified that the device is in the idle state at the time investigators arrived at the scene, it's clear that someone deliberately turned the device into idle state in order to stop it

from triggering the intruder alarm. In that case, fingerprints on the button of the IoT device can potentially help to identify the insider. Once the ML classifier was built, it is integrated into the *EMvidence* framework as a third-party ML model for identifying internal state of the particular type of IoT devices.

5.4 Discussion

As modern digital forensic investigations are increasingly encountering IoT data sources that provide vital information to solve cases, the need for non-intrusive and reliable ways of inspecting IoT devices arises strongly. This chapter highlighted the potential of EM-SCA techniques combined with ML algorithms to tackle this problem. Using two representative IoT devices, a series of experiments were performed to demonstrate that the internal activities of IoT devices can be identified with a significant reliability.

When cryptography-related events are being performed on IoT devices, they can be identified through the EM radiation patterns with a considerable accuracy. When multiple known firmware versions can run on an IoT device, the exact version of the firmware can also be detected. Additionally, if the firmware that is supposed to be running on an IoT device has been maliciously modified, it can be detected with a high reliability. Finally, the internal behaviour of an IoT device can be identified through its EM radiation patterns as well. In the experimental evaluations of this work, a variety of ML algorithms were used to test classification capability of EM radiation patterns, such as MLP, LSTM, and SVM algorithms. The exact choice of an ML algorithm can depend on various factors, such as the number of classes that need to classified, the amount of available datasets for training and testing models, and the limit of the computational resources available to process EM data.

The findings of this chapter can be used to build plug-ins for the *EMvidence* framework by incorporating pretrained ML models to detect interesting internal information of IoT devices in forensic context.

Chapter 6

Curse of Dimensionality: Increasing the Efficiency of EMvidence

6.1 Introduction

The methods introduced and evaluated in Chapter 5 prove that important forensic insights can be acquired from IoT devices through their EM radiation. The EM data for such analysis can be acquired from an investigative scene and processed at a forensic laboratory with high computational resources. However, there can be situations where the acquired EM data should be processed on-the-spot in order to make important decisions based on the EM-SCA outcome. This is where the ability to perform EM-SCA on moderately-resourced computers becomes vital. To achieve that, the efficiency of EM-SCA-based forensic insight-gathering methods need to be increased considerably.

Processing EM data demands high computational resources due to several reasons. Firstly, EM data are typically acquired using extremely fast sample rates. Therefore, EM trace files are large in size and causes storage and processing overhead, e.g., several gigabytes of EM data for an observation of one minute. Secondly, EM data are acquired with high bandwidths to cover

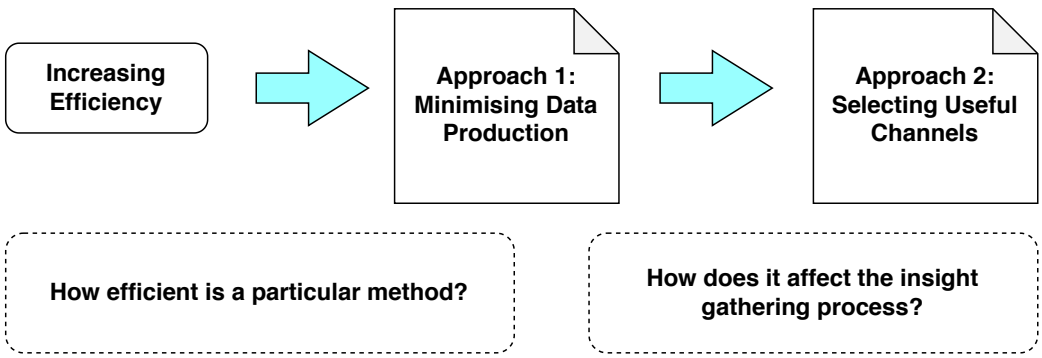

Figure 6.1: The two experimental approaches to increase the efficiency of gathering forensic insights.

as many information-leaking frequencies are possible. EM data acquired with high bandwidths are highly dimensional. For example, EM data acquired with a bandwidth of 20 MHz produces an STFT vector of 20,000 elements even with a window size as small as 1 ms. As a consequence, when capturing and analysing EM data on-site, large data storage space, memory, and processing power are required on the analyst's computer.

This chapter focuses on the challenge of increasing efficiency of gathering forensic insights from IoT devices with the goal of making it possible to perform on moderately-resourced computers. The rest of this chapter is organised as follows. Section 6.2 discusses the potential avenues available to explore towards this objective. Firstly, Section 6.3 focuses on one of the two approaches where the effect of reducing the data production is considered. Secondly, Section 6.4 takes on the possibility of reducing the dimensionality of EM data through intelligent channel selection. Finally, Section 6.5 discusses the implications of the experimental findings of this chapter to the broader objective of leveraging EM-SCA for digital forensics of IoT devices.

6.2 Considerations for Experiments

In order to increase the efficiency of gathering forensic insights, two experimental approaches can be considered (see Figure 6.1). The first approach is the reduction of the amount of EM data produced by SDR hard-

ware and intermediate data produced by preprocessing stages. By doing so, it is expected to minimise the amount of computational and storage resources such large datasets demand. The second approach is the intelligent selection of information-leaking channels from highly dimensional EM data. With reduced dimensionality, the execution time of ML-based forensic insight-gathering methods can be reduced and hence the efficiency of the entire process can be increased.

6.3 Approach 1: Minimising Data Production

The large amount of EM data produced by SDR devices and preprocessing stages is one of the reasons for the inefficiencies in forensic insight gathering using EM-SCA. As the first approach, this section explores the possibility of reducing the EM data production. Towards this goal, three aspects are evaluated experimentally: the processing overhead, the storage overhead and the transmission overhead of EM data.

6.3.1 Electromagnetic Data Processing Overhead

The methods for gathering forensic insights face a processing overhead due to the amount of EM data that are available to be processed. These EM data are typically consumed through a sliding window that goes across each EM trace [177]. It is interesting to evaluate whether fine adjustments to the sliding window can help the forensic insight-gathering methods to relieve from real-time EM data processing overhead. Therefore, the following experiment was conducted to evaluate effect of sliding windows in real-time EM data processing.

A HackRF SDR device was configured to sample EM radiation from an IoT device at 20 MHz sample rate. The produced I/Q data stream was directed to a Python script that extracts small segments of EM data using a sliding window. The size of the sliding window was fixed to 10 ms. The data collection duration was set to 10 s. Now, a set of EM data acquisition trials were conducted each with a unique sliding window step size. The sliding window step

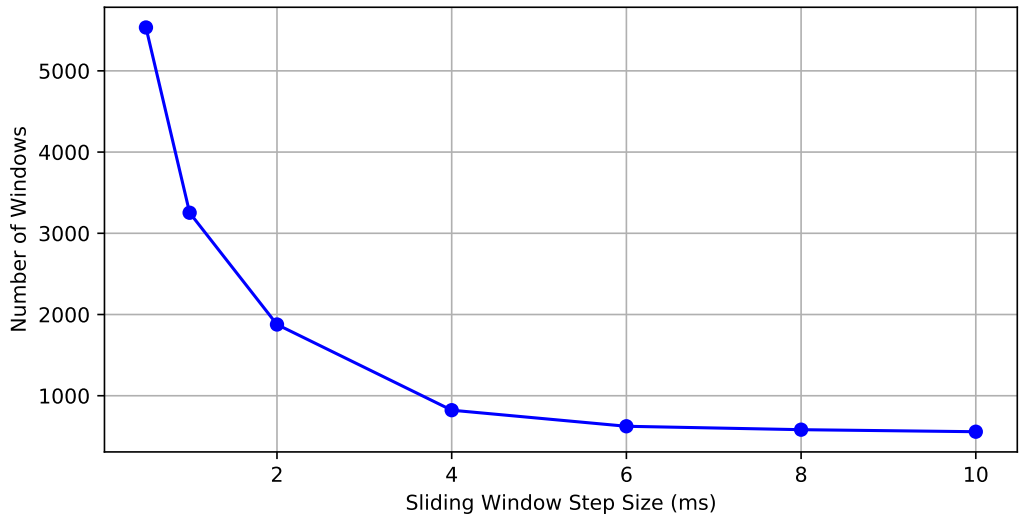

Figure 6.2: The variation of the number of sliding windows produced against the sliding window step size.

sizes considered were 0.5, 1, 2, 4, 6, 8, and 10 ms. The number of sliding windows produced in each trial were plot against the sliding window step size (see Figure 6.2).

Smaller sliding window step sizes produce an extremely large number of windows that should be processed. This can significantly increase the processing overhead for the forensic insight-gathering methods. Therefore, it would be desirable to increase the sliding window step size – making it equal to the window size – to maintain the rate of sliding window production at a low enough level to be manageable. However, on the other hand, an overlapping sliding window is useful for ML-based classifiers to increase the chance of detecting a specific pattern in the signal. In that sense, it is desirable to keep the sliding window step size smaller than the window size itself.

Under these circumstances, there may be an optimal sliding window size and a sliding window step size for each forensic insight-gathering method. However, this optimal size can differ across methods, making it impossible to set a generalised size, which would work for all the forensic insight-gathering methods.

6.3.2 Electromagnetic Data Storage Overhead

When capturing EM data using an SDR device, the use of extremely fast sample rates is necessary to increase the amount of information it captures. When these data are saved into files, the sizes of files are considerably large even for small time windows. For instance, consider a scenario where a HackRF SDR is capturing EM data on 20 MHz sample rate for a period of 1 minute. Each sample generated by the HackRF device through GNURadio library consists of two 32 bit floating-point values representing *Quadrature* and *In-phase* components of the sample in I/Q interleaved stream format. This means, each I/Q sample is 8 bytes long. Therefore, the size of the 1 minute signal capture is approximately 9 GB (8 bytes $\times$ 20 MHz $\times$ 60 s $\approx$ 8.94 GB). In order to apply EM-SCA techniques and machine learning algorithms, thousands of such EM traces are required – making the management of data extremely challenging.

Similar to the sample rate, the use of a large bandwidth is necessary in EM data acquisition. This is because, usually, information-leaking signals occur as multiple side-bands around the CPU clock frequency of the IoT device. A larger data acquisition bandwidth helps to capture these side-bands.

Under these circumstances, it would be desirable to use a lower sample rate while using a large bandwidth during EM data acquisition. Unfortunately, these are contradicting requirements due to the inherent characteristics of SDR devices, i.e., the sample rate and bandwidth are the same in SDRs. One potential solution to achieve this requirement is collecting EM data with the highest possible sample rate/bandwidth and then *down-sampling* the data before saving into EM trace files. However, the question arises whether such down-sampling affects the methods that extract forensic insights using ML classifiers.

In order to evaluate the correlation between sample rate of EM data and the classification accuracy of ML classifiers, the following experiment was conducted. EM traces were captured for 4 different programs running on an Arduino device using a sample rate of 20 MHz. After capturing 600 EM traces per each program, the trace files were *down-sampled* in order to create new sets of EM trace files that have various sample rates; namely 16, 12, 8, 4, 3,

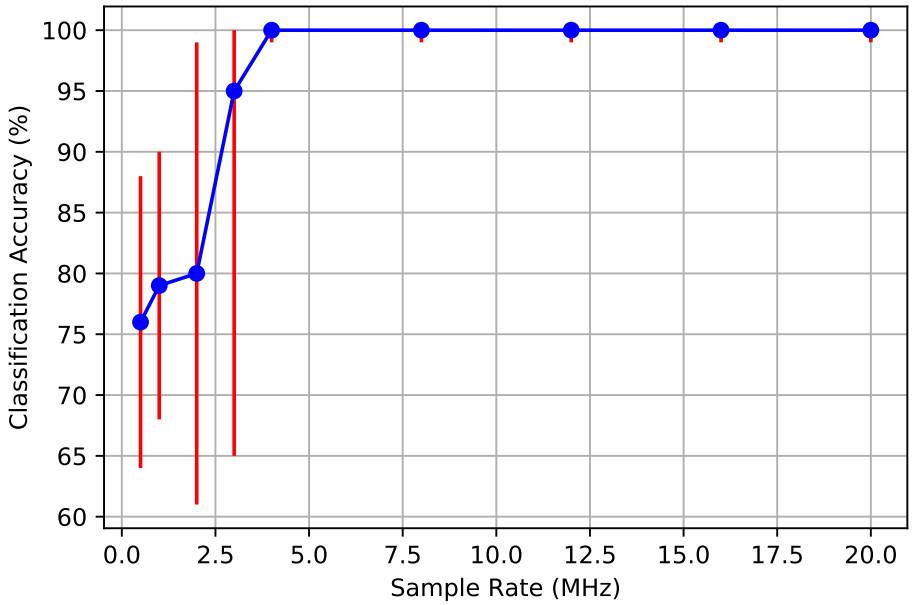

Figure 6.3: The effect of EM trace sample rate to the signal classification accuracy when used with 4 class classifier to identify four different Arduino programs.

2, 1, and 0.5 MHz. Using each dataset (representing its unique sample rate) a neural network-based classifier was trained and tested. After conducting a 10-fold cross-validation for each classifier, the mean *F1-score* was taken along with the 95% confidence interval.

Figure 6.3 illustrates the variation of classification accuracy against the sample rate. It is evident that the classification accuracy is not affected by sample rates as low as 4 MHz. However, when the sample rate goes below 4 MHz, the classification accuracy plummets along with a significant increase in the error margin, depicted in red in Figure 6.3. Considering the maximum sample rate of the HackRF, i.e., 20 MHz, and the lowest possible sample rate that did not adversely affect the classification accuracy in this experiment, i.e., 4 MHz, it is possible to save 80% of the previously required storage space to store the EM data.

This result indicates that it is possible to use a large bandwidth and a lower sample rate without negatively affecting the performance of classification algo-

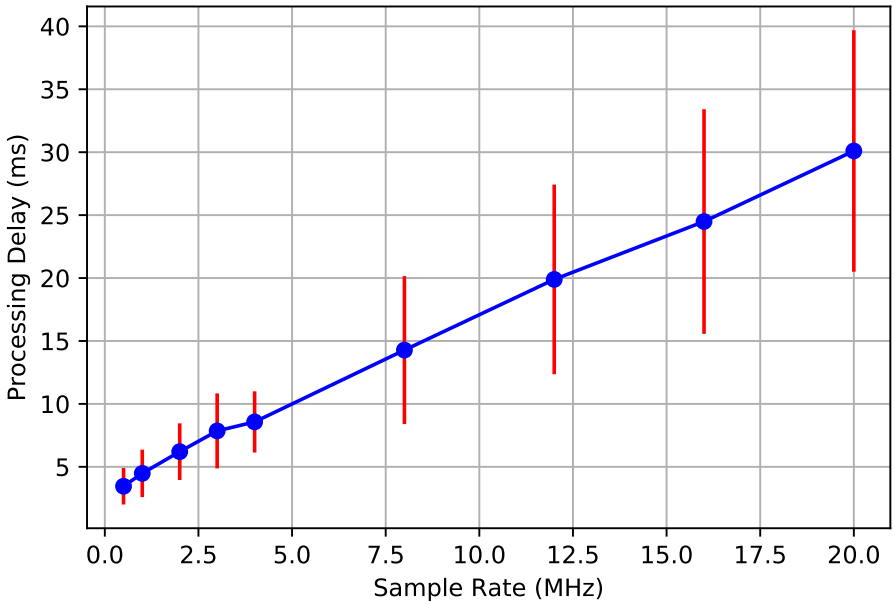

Figure 6.4: The variation of EM data processing overhead against sample rate when used with 4 class classifier to identify four different Arduino programs.

rithms. The lowest possible sample rate has to be decided when constructing an ML model. This could be a significant advantage when capturing EM data in on-site usage scenarios with portable equipment.

6.3.3 Electromagnetic Data Transmission Overhead

The experiment described in Subsection 6.3.2 used EM data captured and saved into I/Q interleaved data files, which were later processed and used to train several ML classifiers. However, when using such ML-assisted EM-SCA methods for the live forensic analysis of IoT devices, real-time preprocessing and transfer of data into ML algorithms is necessary. This is a challenging task since data preprocessing and classification tasks have to be performed within a tight time window in order to keep up with the real-time I/Q data stream. When delivering EM samples in real-time from SDR devices to multiple components of software system, TCP sockets are commonly used [48]. Therefore,

in order to maintain a stable real-time data processing system, the data preprocessing and ML stages must perform faster than the TCP retransmission timeout. On a Linux system, this timeout is typically set to 200 ms and incremented at each timeout up to 15 times [181].

In order to evaluate this aspect, the experiment presented in Subsection 6.3.2 was extended to measure the preprocessing and classification delay of the same classifier against varying sample rates of EM data. Figure 6.4 illustrates the variation of this preprocessing and classification delay of captured data against the sample rate of the SDR device. It is evident that the delay increases linearly with the sample rate. Even at the highest sample rate of the HackRF SDR, i.e., 20 MHz, the processing delay does not exceed 40 ms, which is well below the TCP retransmission timeout. Although the classification delay of ML algorithms may differ depending on the type of the algorithm and its configuration, this result provides a confidence that fast sample rates do not cause any considerable burden to the I/Q data transmission across TCP sockets to multiple ML models for the parallel acquisition of forensic insights.

6.4 Approach 2: Selecting Useful Channels

The experiments performed in the first approach of this chapter have shown that the reduction of sample rate is a viable option for classification scenarios with a smaller number of classes. However, with increasing number of classes, the decrease of sample rate may cause the acquired dataset to lose vital information that are necessary to perform accurate predictions. Therefore, it is desirable to keep the sample rate at the highest possible level and use alternatives to increase the efficiency of ML. The second approach considered in this section is selecting useful channels from the already acquired data with high sample rates.

While there is a large number of frequency channels that can potentially carry information about the activity of a device's MCU, typically a small subset of them are useful [85]. There can be channels that carry redundant information while some others may not leak any information at all. Therefore, the

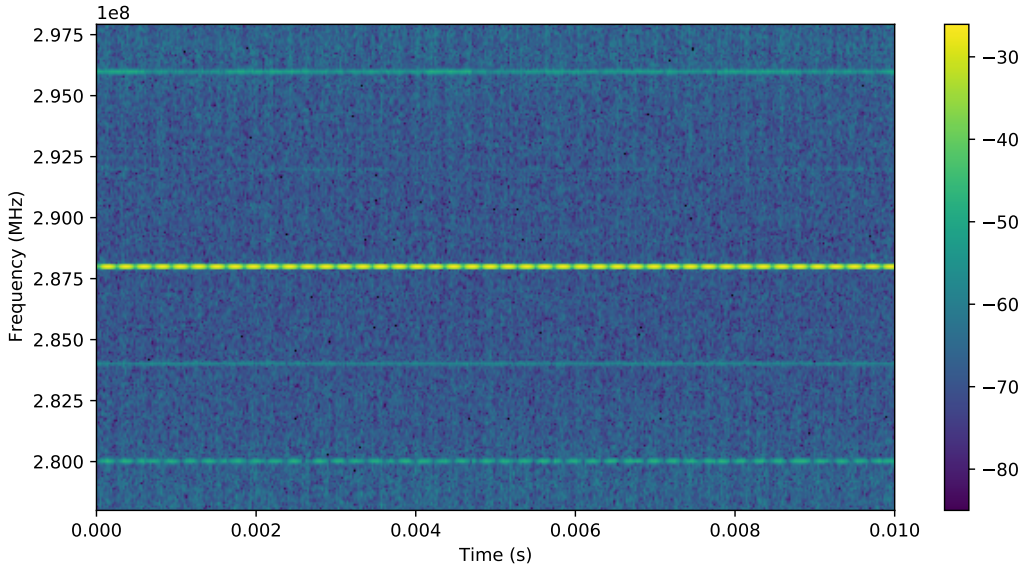

Figure 6.5: Spectrogram of the observed EM signal from DUT

identification of frequency channels that are useful out of the large number of available channels can improve the efficiency of performing EM-SCA. In order to address this problem, this work presents a systematic methodology to identify information-leaking frequency channels from high dimensional EM data, essentially reducing dimensionality.

When listening to the EM radiation of the CPU of a computing device, the system clock frequency is typically considered as the key focus [90]. However, it is not possible to predetermine the exact frequency channel that leaks information. In most practical scenarios, multiple frequencies closer to the CPU clock frequency can leak information [58]. Due to this uncertainty of the information-leaking frequency, it is necessary to observe EM radiation over a wider bandwidth around the CPU clock frequency. For example, consider a scenario of listening to the CPU of an IoT device running at 288 MHz. Signals are sampled using an SDR device by tuning it to 288 MHz and setting the bandwidth to 20 MHz. Consequently, the captured EM traces include signals from 278 MHz to 298 MHz, as shown in Figure 6.5. Among this wide band of signals, some will contain useful information while a majority are unlikely to contain anything useful whatsoever. When listening to EM radiation of the

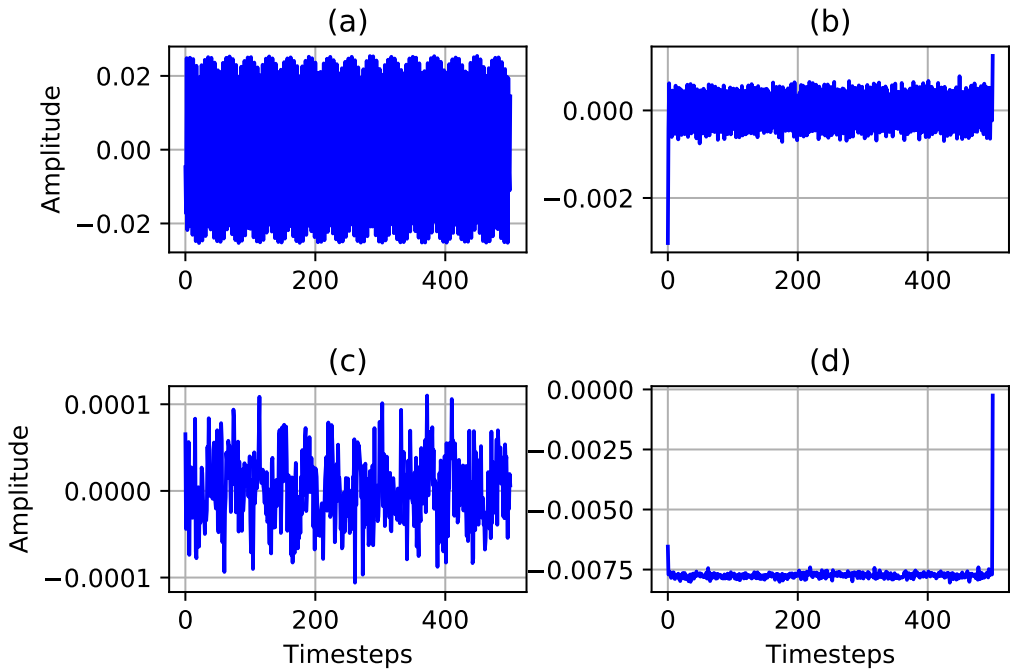

Figure 6.6: Waveform of some randomly selected channels of the EM dataset.

CPU of a device, it is important to have an SDR device that supports the frequency in question. In the case of HackRF SDR that is used in this work, it is possible to tune to any frequency between 1 MHz to 6 GHz, which is sufficient in most practical scenarios [41].

The identification of information-leaking channels from a wide band of channels is a challenge that an attacker needs to overcome in order to efficiently perform EM-SCA. One potential approach is plotting randomly selected channels and visually identifying the channels that have apparent changes over time. Figure 6.6 illustrates some of such randomly selected channels of EM data acquired from an IoT device. However, due to the availability of a large number of channels to inspect, this is an arduous task and not realistically feasible across the entire frequency range. Another potential method of reducing the number of channels to inspect is breaking the frequency domain into equally-sized bins and then averaging the signals within each bin. The weakness of this method is the possibility of having multiple information-leaking channels in the same bin and getting them averaged. This likely results

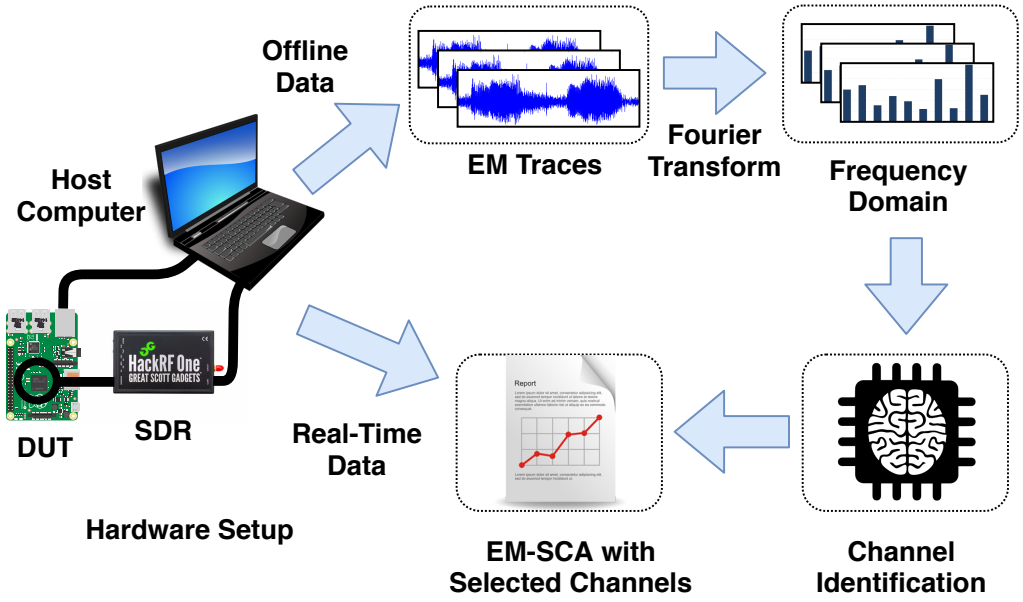

Figure 6.7: The workflow to generate EM traces, identify channels, and finally perform EM-SCA with selected channels.

in a loss of valuable information-leaking channels.

This work experimentally proposes to reduce the number of EM channels from a large bandwidth through multiple channel selection techniques. Once the information-leaking channels are identified for a particular device type, real-time EM signals can be captured from IoT devices and only the identified useful subset of channels can be used for gathering forensic insights (see Figure 6.7).

6.4.1 Procedure of Experiments

For testing channel selection methods, 10 programs running on an Arduino device, named as class 0 to class 9, were used to create an EM dataset. Code Snippet 2 illustrates an example program used to produce the dataset. The programs differ from each other from the number of `for` loops, however, keeping the time complexity at $O(n)$ on each. The choice of programs was made with the goal of exploring the possibility of detecting even minor variations in the code [96]. Therefore, the difference between two consecutive programs is

```
/* Arduino test program */
void setup(){
}
void loop(){
    for(int i=0, i<20, i++) { delay(10); }
    for(int j=0, j<20, j++) { delay(10); }
    /* further loops */
}
```

Code Snippet 2: An example Arduino program used to produce EM data for the experiments.

maintained at a minimum. Since the scope of this work limits to IoT devices, the number of possible programs that can occur on a device is limited. Therefore, this experiment assumes that the programs that are being searched for are known in the first place.

Although the Arduino's system clock runs at 16 MHz, a higher harmonic observed at 288 MHz was used when acquiring data in order to avoid external noise sources. Initially, EM traces were acquired while the device is running the 10 different programs repetitively. Each EM trace file contains a 500 ms long observation with a sample rate of 20 MHz. Collected EM traces are converted to the frequency domain through STFT function with a window size of 1 ms. The resulting dataset contains 20,000 individual frequency channels representing each of the 10 programs of the IoT device across time.

This dataset is fed into a series of channel selection methods to identify a limited set of information-leaking channels (see Figure 6.8). In the experiments using the channel selection methods, 100 channels out of 20,000 (that represents 0.5% of the total channels) were selected. The effectiveness of the channel selection methods is evaluated by attempting to classify the 10 Arduino programs using the selected channels with the help of trained ML models. For this purpose, Random Forests (RF) [182] classification algorithm was used. RF models are fast and very accurate in prediction. Furthermore, they can process datasets with noise and `NaN` (Not a Number) values. RF has two main parameters: the number of created trees, i.e., estimators, and the

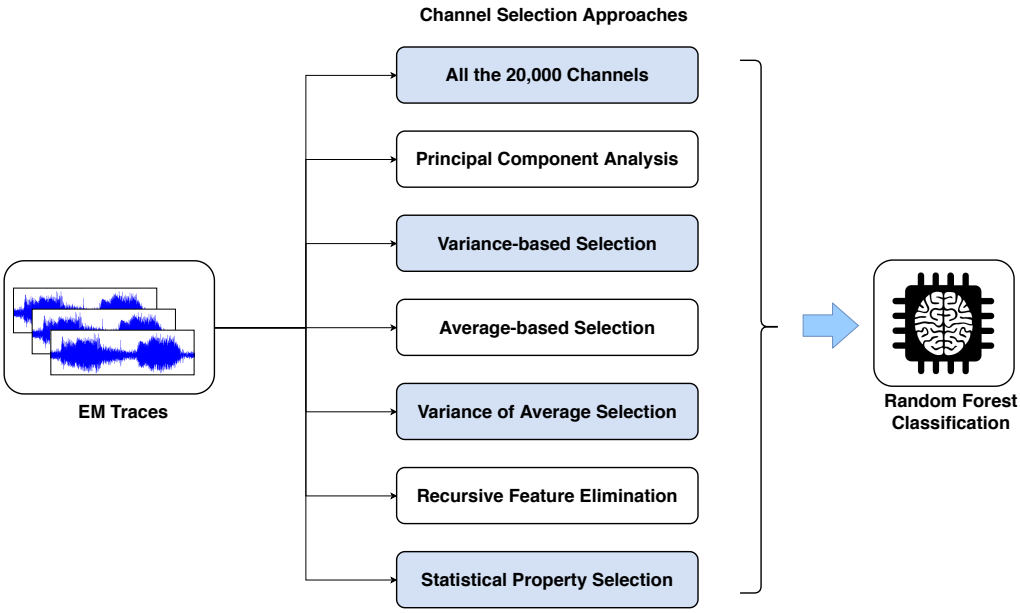

Figure 6.8: The series of methods explored for channel selection.

depth of those trees. The final prediction is the majority vote among all the created trees. In these experiments, all the RF methods use 500 estimators and the trees have a maximum depth of 50 levels. The accuracy of the tested models was the average after applying cross-validation with five partitions and ten repetitions.

Whenever the average and variance were calculated in channel selection methods, the outliers values were removed beforehand. Outliers were considered as those points that had a Z-score[1] absolute value higher than 3. A close inspection of the values of the 20,000 channels revealed that almost all of their values are very close to zero. Experimentation demonstrated two things: firstly, most channels have small variance and secondly, the average of all the sample points of each channel is, in general, very low. With these first hand insights, various experiments conducted and their results are outlined in the following subsections.

[1]Z-scores are used in statistics to measure an observation's deviation from the group's mean value [183]. They are also known as Altman Z scores due to their developer, Edward Altman. According to the normal distribution table, 99% of the values will have an absolute value of less than 3.

6.4.2 Using 20,000 Channels

In this first experiment, a supervised ML model was created using all the available channels. The reason for this is to have a baseline for performance comparison with other channel selection methods. For a channel selection method to be qualified as successful, the ML models built using the channels it selected have to achieve a classification accuracy as good as or better than the baseline accuracy. Due to the fact that the number of channels is very high, 5,000 trees were used in this experiment as opposed to the 500 trees used in the rest of the experiments.

The results of the experiments are shown in Table 6.1 and the confusion matrix of the results is shown in Figure 6.9. The average accuracy of the experiments conducted is 0.9315, and the time to predict 2,004 samples was 7.7435 s. The results demonstrate that the accuracy for classes (0, 1, 2, 3, 4, 7 and 9) were 100% correct, for classes (5 and 8) are acceptable, but the accuracy for class 6 is very low. The algorithm confuses class 6 with classes 5 and 8 quite often. If class 6 was not considered from the EM dataset when building ML model, the overall accuracy it achieves is 0.9804.

Table 6.1: Average accuracy per class using the entire 20,000 channels.

Class	Accuracy
0	1
1	1
2	1
3	1
4	1
5	0.9043
6	0.5129
7	1
8	0.9199
9	1

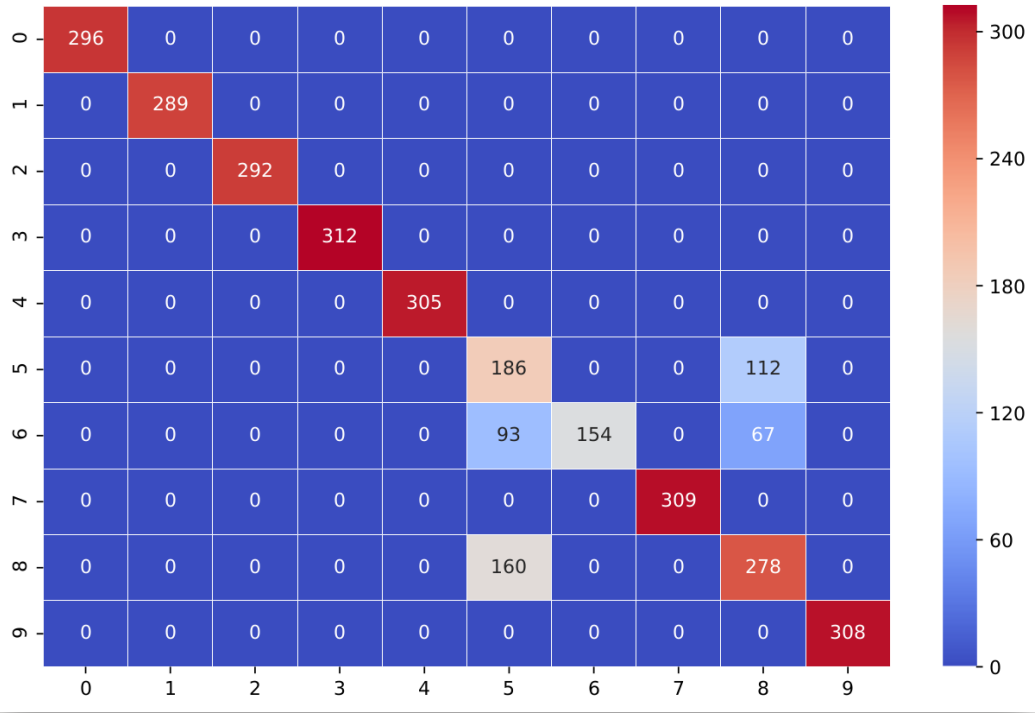

Figure 6.9: The confusion matrix of classifying programs using all the channels.

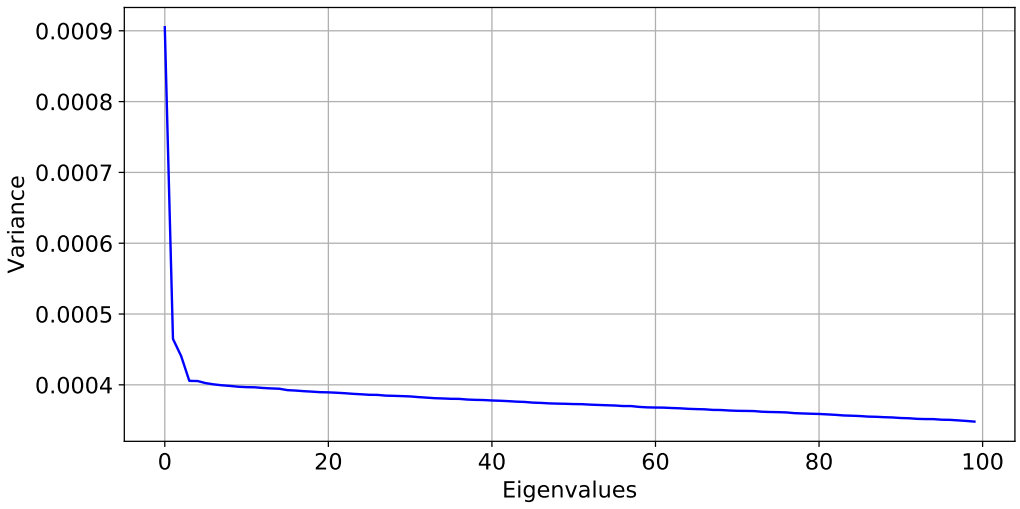

Figure 6.10: Variance (y-axis) of the top 100 eigenvalues (x-values) when applying principal component analysis.

6.4.3 Principal Component Analysis

In this second experiment, Principal Component Analysis (PCA) [184] was applied, which uses a linear combination of weighted variables to drastically reduce the number of features. The new features are linear combinations of the previous ones and are called *eigenvectors* (also called *principal components*). Each of the eigenvectors has an eigenvalue and they are ordered according to this value in such a manner that the first components have more information than the last ones (that can be rejected). Ideally, PCA will reduce the feature space without losing significant information, which allows the creation of models in less time and with a higher accuracy. In Figure 6.10, the top 100 eigenvalues are presented that were used to create a predictive model.

Table 6.2: Average accuracy per class for the best 100 PCA components.

Class	Accuracy
0	0.3438
1	0.2823
2	0.3882
3	0.1457
4	0.1373
5	0.0935
6	0.1768
7	0.2459
8	0.1375
9	0.1555

After running the experiments, the average accuracy of PCA was found to be 0.1870, which is very poor. This is likely due to multiple reasons. The original set of features is 20,000, which is very a high number for PCA. Furthermore, most of those original features are very low constant values. In addition to that, the number of variables used is also considerably higher for PCA. The results by class are depicted in Table 6.2. The low performance of PCA was found to be not favourable and, therefore, was discarded.

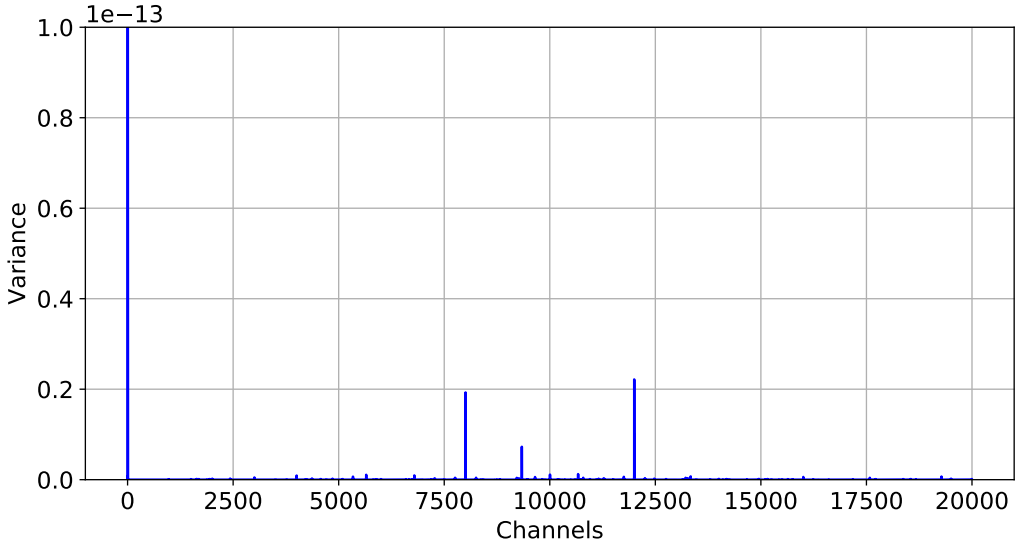

Figure 6.11: The variance (y-axis) of the 20,000 channels (x-axis). The limit of y-axis is set to 10^{-13} in order to visualise lower values. However, the variance of 5^{th} and 6^{th} channels are 0.000282 and 0.000222 respectively.

6.4.4 Channel Selection Based on the Variance

In this experiment, the variance for each channel was calculated and subsequently, the highest 100 were selected. Prior to calculating the variance, the outliers were removed using Z-score as described in Subsection 6.4.1. The spectrum of all the channels after calculating the variance is depicted in Figure 6.11. It is evident that the variance is very low for most channels. In order to increase the visibility of the lower peaks, the limit of the y-axis is set to 10^{-13} in this figure. However, the variance of 5^{th} and 6^{th} channels are 0.000282 and 0.000222 respectively that goes beyond the limit of the y-axis.

During the experiments, the variance threshold was set to select at least the top 100 channels with the highest variance. Accordingly, 103 channels were selected by setting the threshold to 1.0632×10^{-8}. The selected channels were saved in a matrix along with their class and an RF classifier was trained to predict the classes. The average accuracy of the experiments is 0.5431. This is substantially higher than the performance of the PCA experiment, i.e., 0.1870, but still quite far from acceptable performance. The time for building each RF model with 100 features was 1 m and 37 s. The performance per

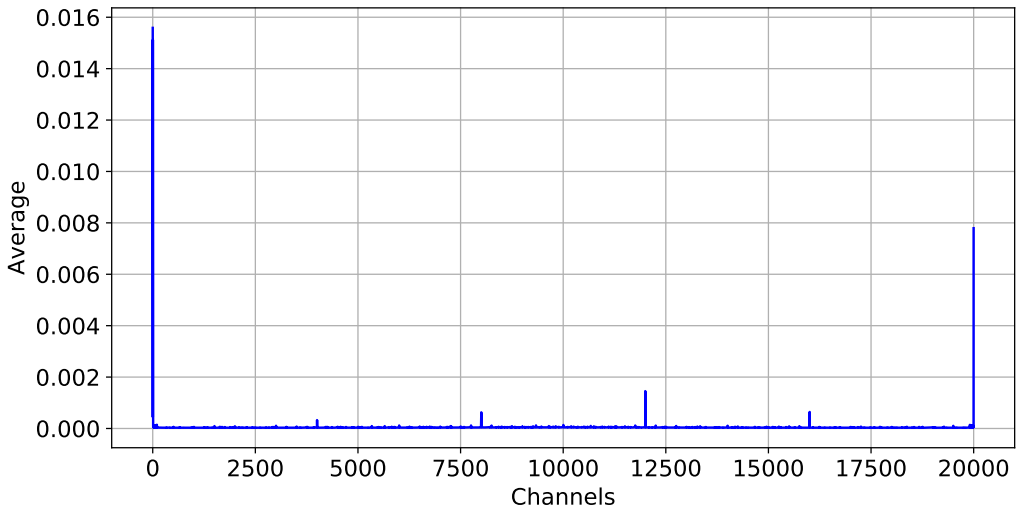

Figure 6.12: Average (y-axis) for each of the 20,000 channels (x-axis).

class is shown in Table 6.3.

Table 6.3: Average accuracy per class of the highest 103 channels ordered by variance.

Class	Accuracy
0	0.6473
1	0.5965
2	0.5391
3	0.5882
4	0.8027
5	0.2684
6	0.3845
7	0.4830
8	0.6272
9	0.5058

6.4.5 Channel Selection based on the Average

After observing that the results of applying channel selection based on the variance were not sufficient, a selection based on the average was explored; again removing the outlier values as described in Subsection 6.4.1. The rea-

soning behind this choice is that the channels that are active tend to have high values while non-active channels stays closer to zero, most of the times. Figure 6.12 depicts different average values for the channels. After calculating the average, a threshold of 6.9936×10^{-5} was set to select the highest 100 channels. The accuracy of the RF classifier was 0.5423, which was very similar to the channel selection made on the basis of variance. The performance per class is shown in Table 6.4.

Table 6.4: Average accuracy per class of the highest 100 channels ordered by average.

Class	Accuracy
0	0.6555
1	0.6067
2	0.5197
3	0.5770
4	0.8025
5	0.2694
6	0.3696
7	0.4939
8	0.6288
9	0.5196

6.4.6 Applying Average per Class and Variance between the Classes

In this experiment, the previous two approaches were combined to select channels using both the variance and the average. As the first step, the average value of each channel per activity was calculated, removing the outlier values. The original matrix has 20,000 rows (where each row represents a channel) and 10,020 columns (where each column has the timestamp values of each channel for a given class). The resulting matrix had the same number of rows (channels) but only 10 columns (one per class). The second step was to calculate the variance between the average of the 10 classes for all the channels, and the result is shown in Figure 6.13.

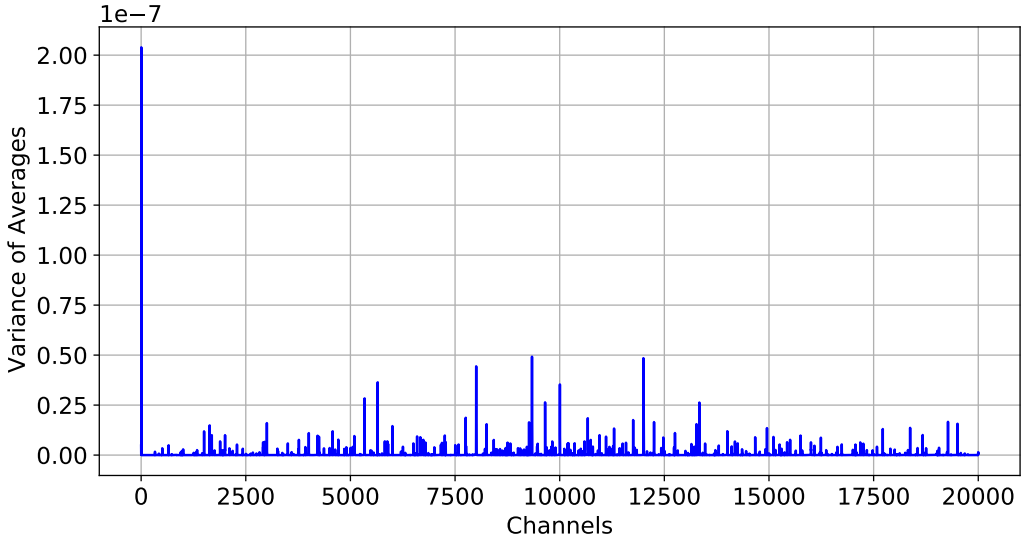

Figure 6.13: Variance between the average of each of the classes for all the channels.

It is evident that the resulting spectrum has much more diversity than the the previous experiments where only the variance or only the average were used. A variance threshold of 0.000033 was used to select the highest 100 channels from this result. Subsequently, the selected channels were used to predict the corresponding classes using an RF classifier. As can be seen in Table 6.5, the results of the experiments are much better than in previous experiments. For this experiment, three different numbers of channels were used. The average accuracy with the 10 higher channels is 0.5753, with the 100 higher channels is 0.9047 and with the 500 higher channels is 0.9395.

6.4.7 Applying Recursive Feature Elimination

The Recursive Feature Elimination (RFE) is a wrapper method, i.e., implements supervised models during its execution, for selecting features proposed by Guyon et al. [185]. Initially, RFE creates a model using all the possible attributes of the dataset. Then, each attribute is ranked according to its importance. Using these ranks, RFE rejects the weakest attributes and creates a new model, whose performance is again evaluated. This process is repeated until it reaches the minimum number of required features. In order to have

Table 6.5: Average accuracy per class after calculating the variance between the average per activity.

Class	10 Channels	100 Channels	500 Channels
0	0.8220	0.9995	1
1	0.7364	1	1
2	0.8019	1	1
3	0.9090	1	1
4	0.5613	0.9990	1
5	0.3531	0.7749	0.9351
6	0.3592	0.5135	0.5179
7	0.4028	1	1
8	0.3828	0.7603	0.9422
9	0.4247	1	1
Average	**0.5753**	**0.9047**	**0.9395**

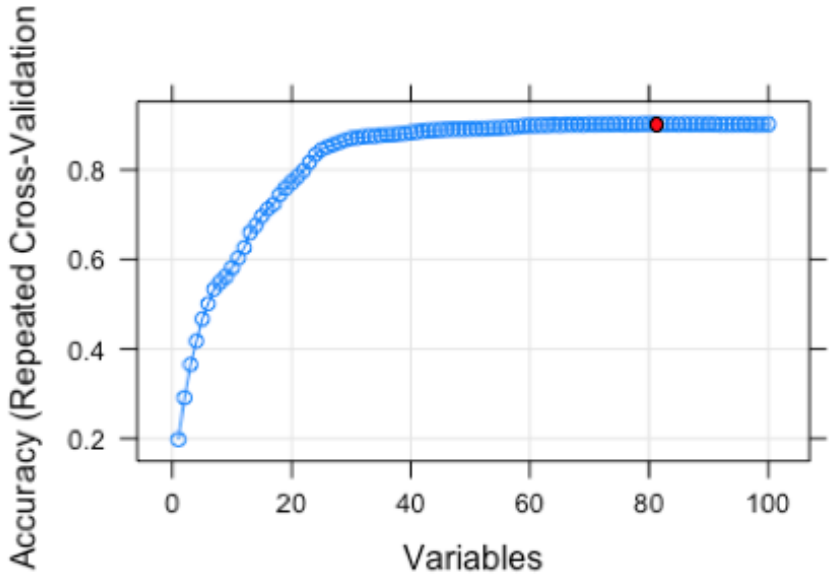

Figure 6.14: The optimal number of features (model with highest performance) for the RFE algorithm is 81 (marked with a red dot).

more reliable results, RFE applies cross-validation. As shown in Figure 6.14, RFE gives a list of feature sets along with the corresponding model's performance. The optimal subset of features is selected from the results of the model

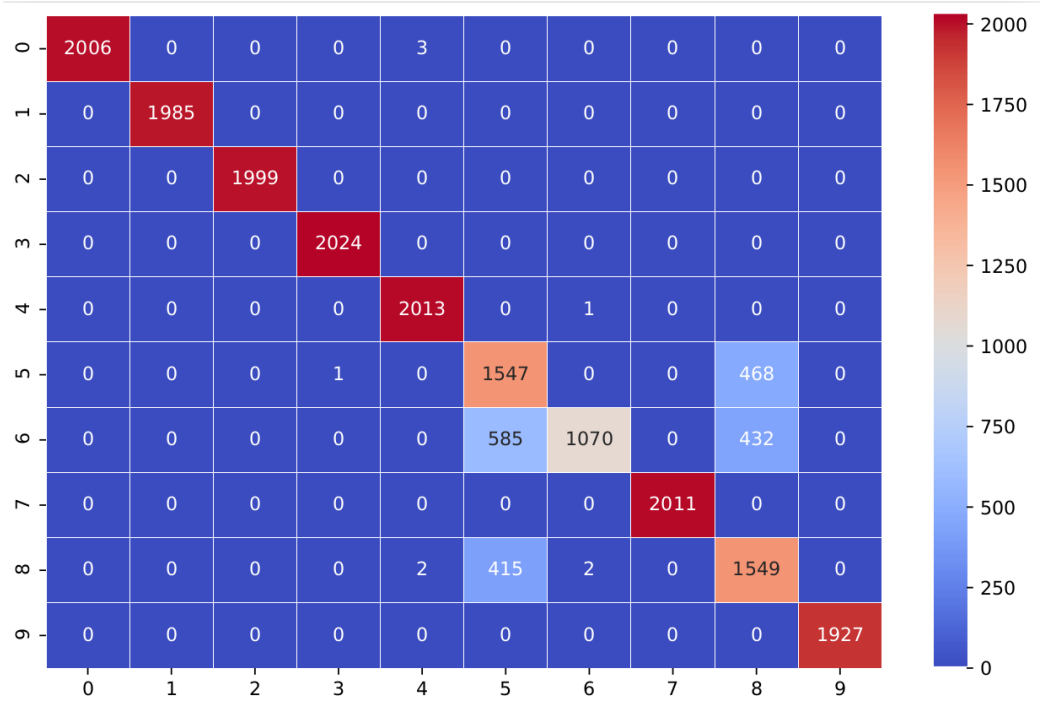

Figure 6.15: The confusion matrix for the 10 Activities and 81 features, i.e., the optimal number for RFE.

that demonstrates the highest performance according to the selected metric. Various metrics, such as the accuracy, the Receiver Operating Characteristic (ROC), and the F1-Score, can be used for this purpose. In this experiment, the accuracy was used as the metric for the RFE method. The time it took to complete the channel selection with RFE was approximately 16 hours.

The optimal number of channels selected by RFE was 81 channels. When those channels were used to train and test an RF model, it achieved an accuracy of 0.9047. However, if the class 6 is removed, the performance improves to 0.9503. The instances of class 6 in the dataset presents very similar values to those of classes 5 and 8. Due to this reason, the RF model finds it difficult to distinguish between them. The fact that such a high accuracy was achieved even when the activities in the dataset are so similar encourages optimism about being able to accurately differentiate real-world IoT firmware activities in the future. Figure 6.15 illustrates the confusion matrix of the results. Table 6.6 illustrates the classification accuracy for the individual classes.

Table 6.6: Average accuracy per class of the selected 81 channels by RFE.

Class	Accuracy
0	0.9986
1	1
2	1
3	1
4	0.9994
5	0.768
6	0.5129
7	1
8	0.7868
9	1

6.4.8 Using a Time Window of 50 Timestamps

The application of a time window is a common practice when dealing with time-series data in various problem domains [96, 177, 186, 187]. Compared to a single data sample, a collection of data samples across a window can contain more information, which can help to make better predictions. Based on this reasoning, the following feature selection method was designed. Firstly, a sliding window was used to extract segments of each channel of the EM dataset. For each window of a channel, the following collection of statistical metrics were calculated from both the time and the frequency domains:

- **Time Domain:** mean, standard deviation, root mean square, maximal amplitude, minimal amplitude, median, number of zero-crossing, skewness, kurtosis, first-quartile, third-quartile, and autocorrelation.
- **Frequency Domain:** mean frequency, median frequency, entropy, energy, principal frequency, and spectral centroid.

Table 6.7 illustrates the classification accuracy of the RF model. It is trained using the data generated by a sliding window of 50 timestamps. The average accuracy of the experiments were 0.8000. The low accuracy was probably due to the fact that the model needs more samples to be trained. The application of a time window reduces the number of samples available for training an ML model.

Table 6.7: Result of the experiments when applying a time window of 50 samples.

Class	Accuracy
0	0.8857
1	0.9800
2	0.9500
3	0.9175
4	1.0000
5	0.5867
6	0.4255
7	0.8583
8	0.5467
9	0.9750

6.4.9 Summary of the Channel Selection Methods

Figure 6.16 summarises the results obtained with the channel selection methods. Channel selection with variance and average did not result in a sufficient classification accuracy for the considered EM dataset of Arduino programs. However, the average per class and variance between classes proved to be effective with an accuracy of 93.95% by reducing the channel space to 500. Furthermore, when applying the RFE method, the 10 classes were identified with a 90.47% classification accuracy.

The performance of the classification is seriously undermined by the class 6 that achieved 51.35% classification accuracy. If this class is not considered, the accuracy of the models after applying the average together with the variance will be 98.60% for 500 features and 94.81% for 100 features. It is necessary to further study the factors of a software program that cause significantly distinguishable radiation pattern in unique channels. The difficulties of distinguishing certain software programs, such as class 6, is only explainable with a better understanding on such factors.

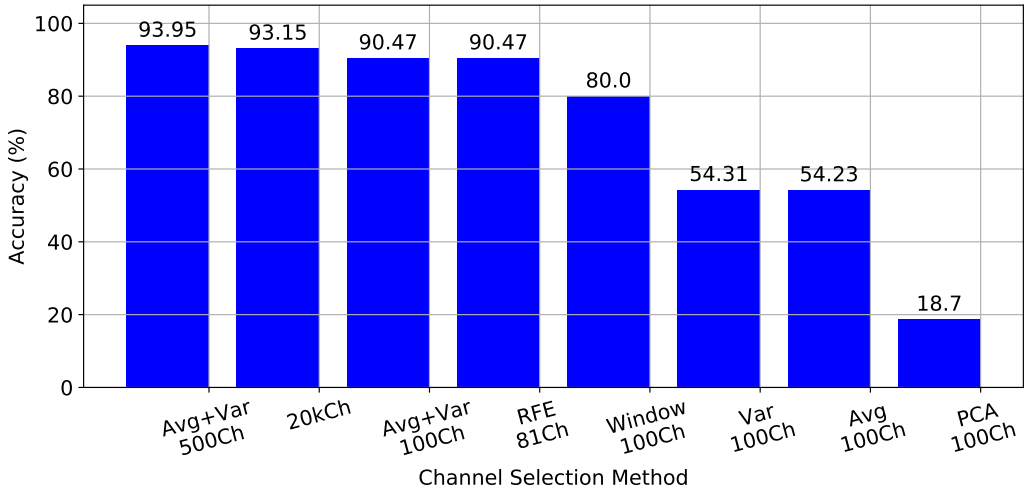

Figure 6.16: Summary of the experimental results with different techniques.

6.5 Discussion

This chapter focused on increasing the efficiency of forensic insight-gathering methods from the EM radiation of IoT devices. Towards this objective, two approaches were considered separately. The fist approach is evaluating potential methods for minimising EM data production. The second approach is the design and evaluation of potential methods to intelligently select information-leaking channels from high-dimensional EM data, effectively reducing the dimensionality.

In the first approach, an experimental study was conducted to identify if the overhead to EM data transmission, processing, and storage can be minimised. Manipulations to real-time sliding window and sample rate during data acquisition were considered as potential contributors to those three overheads. The experiments revealed that the use of a faster real-time sliding window step size can help to minimise processing overhead, however with the risk of having less windows to make predictions. Similarly, the reduction of sample rate reduces the the amount of EM data to be stored. However, it is important to identify the minimum sample rate that does not harm the ML-based classification. The reduction of sample rate below a certain value can negatively affect the forensic insight-gathering process.

The second approach, which considered intelligent channel selection methods, indicates that high dimensional EM side-channel data can be reduced drastically to a manageable set of dimensions that are sufficient to accurately identify software activities running on an IoT device. The evaluation of this approach used an EM dataset representing 10 different programs running on an IoT device. Starting with a dataset that consists of 20,000 dimensions, a channel selection method successfully identified 500 information-leaking channels. A dimensionality reduction of this scale can considerably improve the storage, processing and ML-based prediction using large EM datasets.

The experimental findings of this chapter opens up an opportunity to build plug-ins for the *EMvidence* framework, which can be executed very efficiently. Due to the need of performing EM-SCA-based forensic inspection during the triage examination phase of investigations, the capability to run the *EMvidence* framework on a moderately-resourced computer, such as a laptop, comes in handy.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

With the ever-increasing applications of IoT systems in domestic and industrial environments, digital forensic investigations increasingly require the extraction of digital evidence from them. Most forensically-useful information in IoT devices are currently extracted by intrusive inspections of hardware that makes them less forensically sound [15]. The work presented in this thesis explored the potential of leveraging EM-SCA methods as an alternative approach to forensically inspect IoT devices. Towards this objective, three research questions were defined and studied experimentally.

Firstly, the question was raised whether it is possible to extract forensic insights from IoT devices through EM-SCA. Addressing this question, it was demonstrated that four types of forensic insights can be acquired from EM radiation, i.e., cryptography-related events, firmware version, firmware modification, and device behavioural state. Previously, this type of information of IoT devices could only be acquired by intrusive means, such as chip-off forensics [13]. In contrast, the ML-assisted EM-SCA methods presented in this work reduce the risk of tampering or permanently damaging the IoT evidence sources during investigations.

The second question raised the concern whether it is possible to use such EM-SCA methods on moderately-resourced computers during triage exami-

nation phase of an investigation. The experimental evaluation indicates that there are two potential approaches to increase the efficiency of EM-SCA forensic insight-gathering methods: the careful reduction of EM data sample rate and the intelligent selection of information-leaking channels. The growing amount of disparate data from different types of IoT devices has already been causing inefficiencies in forensic investigations [7]. Similarly, the overhead of the dimensionality and the size of EM data is a problem that obstructs the use of EM-SCA methods in real-world IoT forensic investigations. The two experimentally-evaluated approaches reduce this challenge to a manageable extent.

Finally, the third research question focused on the discovery of a methodology to use EM-SCA methods in the highly diverse and dynamic IoT ecosystem. To address this question, this thesis presented the design of a novel IoT forensic model. The proposed model facilitates the application of a large collection of EM-SCA methods in IoT investigations and enables the implementation and the seamless integration of new EM-SCA methods into the methodology. A proof-of-concept of the model was implemented as an open-source framework called *EMvidence*. In contrast to the existing forensic models [4], the proposed model enables the acquisition of forensic insights from IoT devices during the triage examination phase in a non-intrusive fashion.

The experimental evaluations of this work were conducted on a specific hardware setup, consisting of two representative IoT devices and an SDR data acquisition equipment. Therefore, the exploration of detecting four types of forensic insights was performed by emulating real-world IoT device scenarios on the two representative hardware platforms. Similarly, the experimentation on the efficiency of EM-SCA methods was performed by using an EM dataset produced by the same hardware setup. Under these circumstances, the successful real-world application of the experimental findings requires the implementation of *EMvidence* plug-ins targeting specific real-world IoT devices. Consequently, the experimentation on a diverse set of IoT platforms is a necessary future step on the journey towards strengthening the reliability and generalisability of the findings of this work.

7.1.1 Implications of This Work

The findings of this work have the following immediate implications in the domain of digital forensics.

7.1.1 Future of Digital Forensics

This work is the first attempt to bring EM-SCA into the digital forensics domain. While IoT forensics has been identified as an important area for the progression of digital investigations, current work being published in the domain are mostly limited and focuses on specific IoT devices. The rapid changes in the IoT arena is difficult to be coped with using classical digital forensic approach. The introduction of EM-SCA into the domain can rapidly change the way investigators and researchers are looking at IoT in the future.

7.1.2 Legal Acceptability

Forensic investigations require a substantial amount of reliability for digital evidence to be admissible in a court of law without reasonable doubt. The capabilities demonstrated with EM-SCA combined with the ML approach described in this work need to be time-tested before being used as a reliable and court-admissible evidence source. However, at the current stage, EM-SCA techniques can provide helpful directions for an investigator in order to uncover court-admissible evidence using classical forensic methods.

7.1.3 Platform for New Research

The entrance of newcomers into the research domain of EM-SCA for digital forensics is currently obstructed by multiple barriers. Although the components are available separately, such as SDR hardware, DSP and ML libraries for programming languages, it involves a steep learning curve before getting a working hardware and software setup ready for experimentation. The proposed forensic model and its implementation, the *EMvidence* framework, fill this gap and provides a starting point for newcomers to build and test new research in the domain.

7.2 Future Work

Based on the findings of this work, several future directions can be identified. They are described in the following subsections in no particular order.

7.2.1 Evaluation of Commonly-used Internet of Things

The experimental demonstrations provided in this thesis used Arduino and Raspberry Pi devices as representative IoT devices. The MCUs of the two devices represent the higher and lower ends of IoT device hardware capabilities. Therefore, experimental demonstration on these devices is sufficiently representative of the IoT ecosystem. However, it is necessary to evaluate *EMvidence* with the most commonly encountered IoT devices in real-world digital forensic scenarios. A certain set of devices are frequently drawing the attention of digital forensic community due to their wide involvement in real-world digital forensic investigations. One of such IoT device types is AI voice assistants – among them, the most popular is Amazon Echo [14]. The sheer diversity of the AI voice assistant devices themselves is making forensic analysis a challenging task. Similarly, smart wearable devices such as fitness trackers, and healthcare implants such as pacemakers have a high relevance in modern IoT forensic arena [15]. Therefore, an important future work is to implement a set of *EMvidence* plug-ins to cover such most commonly encountered IoT devices in digital forensic investigations.

7.2.2 Interoperability between Evidence Sources

The digital forensics domain consists of multiple subdomains including file system forensics, network forensics, cloud forensics, mobile forensics, and IoT forensics. There are specialised tools used by investigators to acquire evidence in each subdomain. As different tools are developed by different companies or open-source communities, the formats used by them to produce analysis results greatly differ from one another. As a result, conclusions drawn in investigations by combining findings from different tools and aspects can

cause ambiguities. This challenge of interoperability has been long identified as a problem in the digital forensics community [188].

In recent years, a significant effort has been put to build a standardised specification to represent information produced by various digital forensic tools in different contexts. The results of this effort by the community is Cyber-investigation Analysis Standard Expression (CASE) [189]. Various types of investigation-related information can be represented in a common language using the CASE standard. Recognising the necessity of it, digital forensic tools currently in use are slowly building support for the CASE standard. Under these circumstances, it is necessary to identify potential methods to make EM-SCA-based IoT analysis results compatible with the CASE standard.

7.2.3 Management of Electromagnetic Data

The EM data produced by SDR hardware are stored in raw I/Q interleaved sample format files. Depending on the sample rate and the duration of signal observation, a single such EM data file can reach sizes over gigabytes. In an investigative scenario, multiple such EM data files can be generated. Furthermore, the investigator's computer running an instance of the *EMvidence* framework can contain EM data related to multiple investigations. The management of these EM datasets and their associated metadata is a challenge, which can get worse with time. Therefore, it is necessary to develop metadata storage formats suitable for digital forensic investigation scenarios by augmenting existing standards, such as HDF5 [190], VITA-49 [191], SigMF [192], and AFF4 [193].

7.2.4 Hardware Independence of Machine Learning Models

ML models trained to build *EMvidence* plug-ins currently rely on specific settings of the SDR device. For example, parameters such as sampling rate, bandwidth, level of signal amplification, and the exact positioning of the H-loop antenna over DUT during signal acquisition can play an important role in the final signal classification by training an ML model. Therefore, an ML model

trained and tested on a particular SDR hardware setting may not currently work on new data produced on different settings. Further research is necessary to identify methods that generalise ML models between different data acquisition settings.

7.2.5 Cryptographic Key Retrieval in Forensic Context

The experimental evaluations of this work have demonstrated that is it possible to identify cryptography-related settings of an IoT device through analysing its EM radiation. These cryptography-related events on a device can include encryption/decryption of data stored on the device and communicated through network interfaces. After identifying such cryptography-related settings of a device, the next natural step that should be attempted is retrieving cryptographic keys used by the device. While a large body of literature covers various cryptographic key retrieval attacks using EM-SCA methods, performing such attacks on forensic investigative scenarios with IoT devices found on-the-spot has not yet been formally explored. Building *EMvidence* plug-ins that are capable of performing cryptographic key retrieval attacks is an interesting avenue for future research. When doing so, the potential integration of already existing open-source toolchains, such as ChipWhisperer [138], with *EMvidence* is worthy of further exploration.

Bibliography

[1] M. Scanlon, J. Farina, and M.-T. Kechadi, “Network Investigation Methodology for BitTorrent Sync: A Peer-to-Peer Based File Synchronisation Service,” *Computers & Security*, vol. 54, pp. 27 – 43, 10 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016740481500067X

[2] S. Soltani and S. A. H. Seno, “A Survey on Digital Evidence Collection and Analysis,” in *7th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, 2017, pp. 247–253.

[3] E. Casey, *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet*. Academic Press, 2011.

[4] X. Du, N.-A. Le-Khac, and M. Scanlon, “Evaluation of Digital Forensic Process Models with Respect to Digital Forensics as a Service,” in *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS 2017)*. Dublin, Ireland: ACPI, 06 2017, pp. 573–581.

[5] B. L. R. Stojkoska and K. V. Trivodaliev, “A Review of Internet of Things for Smart Home: Challenges and Solutions,” *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[6] M. Chernyshev, S. Zeadally, Z. Baig, and A. Woodward, “Internet of Things Forensics: The Need, Process Models, and Open Issues,” *IT Professional*, vol. 20, no. 3, pp. 40–49, 2018.

[7] D. Quick and K.-K. R. Choo, “IoT Device Forensics and Data Reduction,” *IEEE Access*, vol. 6, pp. 47 566–47 574, 2018.

[8] I. Yaqoob, I. A. T. Hashem, A. Ahmed, S. A. Kazmi, and C. S. Hong, "Internet of Things Forensics: Recent Advances, Taxonomy, Requirements, and Open Challenges," *Future Generation Computer Systems*, vol. 92, pp. 265–275, 2019.

[9] R. K. Lomotey, J. C. Pry, and C. Chai, "Traceability and Visual Analytics for the Internet of Things (IoT) Architecture," *World Wide Web*, vol. 21, no. 1, pp. 7–32, 2018.

[10] D. Lillis, B. Becker, T. O'Sullivan, and M. Scanlon, "Current Challenges and Future Research Areas for Digital Forensic Investigation," in *The 11th ADFSL Conference on Digital Forensics, Security and Law (CDFSL 2016)*. Daytona Beach, FL, USA: ADFSL, 05 2016, pp. 9–20.

[11] R. Torrance and D. James, "The State-of-the-Art in IC Reverse Engineering," in *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 363–381.

[12] F. Courbon, S. Skorobogatov, and C. Woods, "Reverse Engineering Flash EEPROM Memories using Scanning Electron Microscopy," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2016, pp. 57–72.

[13] S. Watson and A. Dehghantanha, "Digital Forensics: The Missing Piece of the Internet of Things Promise," *Computer Fraud & Security*, vol. 2016, no. 6, pp. 5–8, 2016.

[14] S. Li, K.-K. R. Choo, Q. Sun, W. J. Buchanan, and J. Cao, "IoT Forensics: Amazon Echo as a Use Case," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6487–6497, 2019.

[15] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches and Open Issues," *IEEE Communications Surveys & Tutorials*, 2020.

[16] R. Poussier, V. Grosso, and F.-X. Standaert, "Comparing Approaches to Rank Estimation for Side-Channel Security Evaluations," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2015, pp. 125–142.

[17] E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons," *Integration, the VLSI journal*, vol. 40, no. 1, pp. 52–60, 2007.

[18] S. Wakabayashi, S. Maruyama, T. Mori, S. Goto, M. Kinugawa, and Y.-i. Hayashi, "Poster: Is Active Electromagnetic Side-channel Attack Practical?" in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2587–2589.

[19] M. Randolph and W. Diehl, "Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman," *Cryptography*, vol. 4, no. 2, p. 15, 2020.

[20] "Unix Philosophy," wikipedia.org, Accessed: 2020-07-26. [Online]. Available: https://en.wikipedia.org/wiki/Unix_philosophy

[21] P. Suresh, J. V. Daniel, V. Parthasarathy, and R. Aswathy, "A State of the Art Review on the Internet of Things (IoT) History, Technology and Fields of Deployment," in *2014 International Conference on Science Engineering and Management Research (ICSEMR)*. IEEE, 2014, pp. 1–8.

[22] A. Tewari and B. Gupta, "Security, Privacy and Trust of Different Layers in Internet-of-Things (IoTs) Framework," *Future Generation Computer Systems*, vol. 108, pp. 909–920, 2020.

[23] K. A. Townsend, J. W. Haslett, T. K.-K. Tsang, M. N. El-Gamal, and K. Iniewski, "Recent Advances and Future Trends in Low Power Wireless Systems for Medical Applications," in *Fifth International Workshop on System-on-Chip for Real-Time Applications (IWSOC'05)*. IEEE, 2005, pp. 476–481.

[24] A. Rayes and S. Salam, "Internet of Things from Hype to Reality," *The Road to Digitization; River Publisher Series in Communications; Springer: Basel, Switzerland*, vol. 49, 2017.

[25] S. S. I. Samuel, "A Review of Connectivity Challenges in IoT Smart Home," in *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*. IEEE, 2016, pp. 1–4.

[26] H. Kharrufa, H. A. Al-Kashoash, and A. H. Kemp, "RPL-based Routing Protocols in IoT Applications: A Review," *IEEE Sensors Journal*, vol. 19, no. 15, pp. 5952–5967, 2019.

[27] OpenText Security, "EnCase Forensic," guidancesoftware.com, Accessed: 2020-08-13. [Online]. Available: https://www.guidancesoftware.com/encase-forensic

[28] B. Carrier, "The Sleuth Kit," sleuthkit.org, Accessed: 2020-08-13. [Online]. Available: https://sleuthkit.org

[29] M. S. Ahmad, N. E. Musa, R. Nadarajah, R. Hassan, and N. E. Othman, "Comparison between Android and iOS Operating System in Terms of Security," in *8th International Conference on Information Technology in Asia (CITA)*. IEEE, 2013, pp. 1–4.

[30] A. MacDermott, T. Baker, and Q. Shi, "IoT Forensics: Challenges For the IoA Era," in *New Technologies, Mobility and Security (NTMS), 2018 9th IFIP International Conference on*. IEEE, 2018, pp. 1–5.

[31] E. Casey and G. J. Stellatos, "The Impact of Full Disk Encryption on Digital Forensics," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 3, pp. 93–98, 2008.

[32] A. W. Fritzke, "Obfuscating Against Side-Channel Power Analysis Using Hiding Techniques for AES," Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2012.

[33] E. A. Vincze, "Challenges in Digital Forensics," *Police Practice and Research*, vol. 17, no. 2, pp. 183–194, 2016.

[34] X. Lin, *Introductory Computer Forensics: A Hands-on Practical Approach*. Springer, 2018.

[35] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997, ch. 28.

[36] A. B. Downey, *Think DSP: Digital Signal Processing in Python*. O'Reilly Media, Inc., 2016.

[37] R. Tessier and W. Burleson, "Reconfigurable Computing for Digital Signal Processing: A Survey," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 28, no. 1-2, pp. 7–27, 2001.

[38] A. M. Wyglinski, R. Getz, T. Collins, and D. Pu, *Software-Defined Radio for Engineers*. Artech House, 2018.

[39] W. H. Tuttlebee, *Software Defined Radio: Enabling Technologies*. John Wiley & Sons, 2003.

[40] S. Cass, "A $40 Software-defined Radio," *IEEE Spectrum*, vol. 50, no. 7, pp. 22–23, 2013.

[41] M. Ossmann, "HackRF," greatscottgadgets.com, Accessed: 2020-08-29. [Online]. Available: https://greatscottgadgets.com/hackrf/

[42] T. A. Milligan, *Modern Antenna Design*. John Wiley & Sons, 2005.

[43] RF Explorer, "RF Explorer Near Field Antenna Kit Datasheet," RF Explorer, Spain, Tech. Rep., 2017. [Online]. Available: http://j3.rf-explorer.com

[44] P. Juyal, S. Adibelli, N. Sehatbakhsh, and A. Zajic, "A Directive Antenna based on Conducting Disks for Detecting Unintentional EM Emissions at Large Distances," *IEEE Transactions on Antennas and Propagation*, vol. 66, no. 12, pp. 6751–6761, 2018.

[45] A. Csete, "GQRX SDR Open Source Software Defined Radio," gqrx.dk, Accessed: 2020-06-29. [Online]. Available: https://gqrx.dk/

[46] "SDR# Software," airspy.com, Accessed: 2020-06-29. [Online]. Available: https://airspy.com/download/

[47] E. Blossom, "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux Journal*, vol. 2004, no. 122, p. 4, 2004.

[48] "Tutorial: GNU Radio Companion," gnuradio.org, Accessed: 2020-06-29. [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorial_GRC

[49] National Instruments, "White Paper: What is I/Q Data?" ni.com, Accessed: 2020-06-29. [Online]. Available: http://www.ni.com/tutorial/4805/en/

[50] C. D. O'Connell, "Exploiting Quasiperiodic Electromagnetic Radiation using Software-defined Radio," Ph.D. dissertation, University of Cambridge, 2019.

[51] J. C. Maxwell, "A Dynamical Theory of the Electromagnetic Field," *Philosophical transactions of the Royal Society of London*, vol. 155, pp. 459–512, 1865.

[52] M. Jabbar and M. A. Rahman, "Radio Frequency Interference of Electric Motors and Associated Controls," *IEEE Transactions on Industry Applications*, vol. 27, no. 1, pp. 27–31, 1991.

[53] R. Getz and B. Moeckel, "Understanding and Eliminating EMI in Microcontroller Applications," National Semiconductor, Tech. Rep., 1996. [Online]. Available: https://www.ti.com/lit/an/snoa382/snoa382.pdf

[54] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology (CRYPTO '99)*. Springer, 1999, pp. 789–789.

[55] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to Differential Power Analysis," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 5–27, 2011.

[56] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 16–29.

[57] P. Robyns, P. Quax, and W. Lamotte, "Improving CEMA using Correlation Optimization," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 1–24, 2019.

[58] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers," in *Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS)*, ser. CCS '18. ACM, October 2018.

[59] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[60] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[61] 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*, vol. 12, pp. 2825–2830, 2011.

[62] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An Imperative Style, High-performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8026–8037.

[63] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A System for Large-scale Machine Learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[64] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, "A Practical Implementation of the Timing Attack," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 1998, pp. 167–182.

[65] D. Brumley and D. Boneh, "Remote Timing Attacks are Practical," *Computer Networks*, vol. 48, no. 5, pp. 701–716, 2005.

[66] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Annual International Cryptology Conference*. Springer, 1996, pp. 104–113.

[67] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-tenant Side-Channel Attacks in PaaS Clouds," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 990–1003.

[68] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level Cache Side-Channel Attacks are Practical," in *IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 605–622.

[69] S. J. O'Malley and K.-K. R. Choo, "Bridging the Air Gap: Inaudible Data Exfiltration by Insiders," in *20th Americas Conference on Information Systems (AMCIS)*. Association for Information Systems, 2014.

[70] D. Genkin, A. Shamir, and E. Tromer, "RSA Key Extraction via Low-bandwidth Acoustic Cryptanalysis," in *International Cryptology Conference*. Springer, 2014, pp. 444–461.

[71] W. Van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?" *Computers & Security*, vol. 4, no. 4, pp. 269–286, 1985.

[72] M. G. Kuhn and R. J. Anderson, "Soft Tempest: Hidden Data Transmission using Electromagnetic Emanations," in *International Workshop on Information Hiding*. Springer, 1998, pp. 124–142.

[73] Z. Hongxin, H. Yuewang, W. Jianxin, L. Yinghua, and Z. Jinling, "Recognition of Electromagnetic Leakage Information from Computer Radiation with SVM," *Computers & Security*, vol. 28, no. 1-2, pp. 72–76, 2009.

[74] F. Elibol, U. Sarac, and I. Erer, "Realistic Eavesdropping Attacks on Computer Displays with Low-cost and Mobile Receiver System," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. IEEE, 2012, pp. 1767–1771.

[75] A. Zankl, H. Seuschek, G. Irazoqui, and B. Gulmezoglu, "Side-Channel Attacks in the Internet of Things: Threats and Challenges," in *Solutions for Cyber-Physical Systems Ubiquity*. IGI Global, 2018, pp. 325–357.

[76] D. Agrawal, J. R. Rao, and P. Rohatgi, "Multi-Channel Attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2003, pp. 2–16.

[77] H. W. Ott, *Electromagnetic Compatibility Engineering*. John Wiley & Sons, 2011.

[78] H. Wolfe, "Setting up an Electronic Evidence Forensics Laboratory," *Computers & Security*, vol. 22, no. 8, pp. 670–672, 2003.

[79] M. Ettus and M. Braun, "The Universal Software Radio Peripheral (USRP) Family of Low-cost SDR," *Opportunistic Spectrum Sharing and White Space Access: The Practical Reality*, pp. 3–23, 2015.

[80] R. Callan, A. Zajić, and M. Prvulovic, "A Practical Methodology for Measuring the Side-Channel Signal Available to the Attacker for Instruction-level Events," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 242–254.

[81] A. Zajic and M. Prvulovic, "Experimental Demonstration of Electromagnetic Information Leakage from Modern Processor-memory Systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 56, no. 4, pp. 885–893, 2014.

[82] R. Callan, A. Zajić, and M. Prvulovic, "FASE: Finding Amplitude-modulated Side-Channel Emanations," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3. ACM, 2015, pp. 592–603.

[83] M. Prvulovic, A. Zajić, R. L. Callan, and C. J. Wang, "A Method for Finding Frequency-Modulated and Amplitude-Modulated Electromagnetic Emanations in Computer Systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 59, no. 1, pp. 34–42, 2017.

[84] B. B. Yilmaz, R. L. Callan, M. Prvulovic, and A. Zajić, "Capacity of the EM Covert/Side-Channel Created by the Execution of Instructions in a Processor," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 605–620, 2018.

[85] G. Laput, C. Yang, R. Xiao, A. Sample, and C. Harrison, "EM-Sense: Touch Recognition of Uninstrumented, Electrical and Electromechanical Objects," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 2015, pp. 157–166.

[86] A. Bianchi and I. Oakley, "Wearable Authentication: Trends and Opportunities," *IT - Information Technology*, vol. 58, no. 5, pp. 255–262, 2016.

[87] C. Yang and A. P. Sample, "EM-ID: Tag-less Identification of Electrical Devices via Electromagnetic Emissions," in *IEEE International Conference on RFID (RFID)*. IEEE, 2016, pp. 1–8.

[88] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "Exploiting the Analog Properties of Digital Circuits for Malicious Hardware," *Communications of the ACM*, vol. 60, no. 9, pp. 83–91, 2017.

[89] M. M. Ahmed, D. Hely, N. Barbot, R. Siragusa, E. Perret, M. Bernier, and F. Garet, "Radiated Electromagnetic Emission for Integrated Circuit Authentication," *IEEE Microwave and Wireless Components Letters*, 2017.

[90] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso, "Zero-overhead Profiling via EM Emanations," in *Proceedings of the 25th In-*

ternational Symposium on Software Testing and Analysis. ACM, 2016, pp. 401–412.

[91] R. L. Callan, "Analyzing Software using Unintentional Electromagnetic Emanations from Computing Devices," Ph.D. dissertation, Georgia Institute of Technology, 2016.

[92] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1095–1108.

[93] T. Espitau, P.-A. Fouque, B. Gérard, and M. Tibouchi, "Side-Channel Attacks on BLISS Lattice-Based Signatures: Exploiting Branch Tracing against strongSwan and Electromagnetic Emanations in Microcontrollers," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1857–1874.

[94] B. D. Stone and S. J. Stone, "Radio Frequency-based Reverse Engineering of Microcontroller Program Execution," in *2015 National Aerospace and Electronics Conference (NAECON)*. IEEE, 2015, pp. 159–164.

[95] S. J. Stone, M. A. Temple, and R. O. Baldwin, "Detecting Anomalous Programmable Logic Controller Behavior using RF-based Hilbert Transform Features and a Correlation-based Verification Process," *International Journal of Critical Infrastructure Protection*, vol. 9, pp. 41–51, 2015.

[96] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "EDDIE: EM-based Detection of Deviations in Program Execution," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 333–346.

[97] C. L. Giles, G. M. Kuhn, and R. J. Williams, "Dynamic Recurrent Neural Networks: Theory and Applications," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 153–156, 1994.

[98] X. Wang, Q. Zhou, J. Harer, G. Brown, S. Qiu, Z. Dou, J. Wang, A. Hinton, C. A. Gonzalez, and P. Chin, "Deep Learning-based Classification and Anomaly Detection of Side-Channel Signals," in *Cyber Sensing 2018*, vol. 10630. International Society for Optics and Photonics, 2018, p. 1063006.

[99] D. R. Reising, "Exploitation of RF-DNA for Device Classification and Verification using GRLVQI Processing," Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2012.

[100] C. K. Dubendorfer, "Using RF-DNA Fingerprints to Discriminate ZigBee Devices in an Operational Environment," Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2013.

[101] B. Danev, D. Zanetti, and S. Capkun, "On Physical-layer Identification of Wireless Devices," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, p. 6, 2012.

[102] M. W. Lukacs, A. J. Zeqolari, P. J. Collins, and M. A. Temple, "RF-DNA Fingerprinting for Antenna Classification," *IEEE Antennas and Wireless Propagation Letters*, vol. 14, pp. 1455–1458, 2015.

[103] R. D. Deppensmith and S. J. Stone, "Optimized Fingerprint Generation using Unintentional Emission Radio-frequency Distinct Native Attributes (RF-DNA)," in *Aerospace and Electronics Conference, NAECON 2014-IEEE National*. IEEE, 2014, pp. 327–330.

[104] T. J. Bihl, K. W. Bauer, and M. A. Temple, "Feature Selection for RF Fingerprinting with Multiple Discriminant Analysis and using ZigBee Device Emissions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1862–1874, 2016.

[105] B. Stone and S. Stone, "Comparison of Radio Frequency Based Techniques for Device Discrimination and Operation Identification," in *11th International Conference on Cyber Warfare and Security: ICCWS2016*. Academic Conferences and Publishing Limited, 2016, p. 475.

[106] N. Sohaib ul Hassan, "EM Side Channel Analysis on Complex SoC architectures," *MSc thesis, Tampere University of Technology*, 2016.

[107] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing Keys from PCs using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2015, pp. 207–228.

[108] L. Goubin, "A Refined Power-analysis Attack on Elliptic Curve Cryptosystems," in *International Workshop on Public Key Cryptography*. Springer, 2003, pp. 199–211.

[109] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "ECDH Key Extraction via Low-bandwidth Electromagnetic Attacks on PCs," in *Cryptographers' Track at the RSA Conference*. Springer, 2016, pp. 219–235.

[110] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, "ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1626–1638.

[111] A. Bogdanov, I. Kizhvatov, K. Manzoor, E. Tischhauser, and M. Witteman, "Fast and Memory-efficient Key Recovery in Side-Channel Attacks," in *International Conference on Selected Areas in Cryptography*. Springer, 2015, pp. 310–327.

[112] J.-J. Quisquater and D. Samyde, "Electromagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards," *Smart Card Programming and Security*, pp. 200–210, 2001.

[113] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic Analysis: Concrete Results," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2001, pp. 251–261.

[114] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of Power Analysis Attacks on Smartcards." *Smartcard*, vol. 99, pp. 151–161, 1999.

[115] M. F. Witteman, J. G. van Woudenberg, and F. Menarini, "Defeating RSA Multiply-Always and Message Blinding Countermeasures," in *Cryptographers' Track at the RSA Conference (CT-RSA)*, vol. 6558. Springer, 2011, pp. 77–88.

[116] E. Sanfelix, C. Mune, and J. de Haas, "Unboxing the White-box Practical Attacks Against Obfuscated Ciphers," *Black Hat Europe*, vol. 2015, 2015.

[117] G. Camurati, A. Francillon, and F.-X. Standaert, "Understanding Screaming Channels: From a Detailed Analysis to Improved Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 358–401, 2020.

[118] U. Calari and M. C. Lampkin, "RFID reader," Apr. 15 1997, US Patent 5,621,199.

[119] M. Hutter, S. Mangard, and M. Feldhofer, "Power and EM Attacks on Passive 13.56 MHz RFID Devices," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol. 7. Springer, 2007, pp. 320–333.

[120] T. Kasper, D. Oswald, and C. Paar, "EM Side-Channel Attacks on Commercial Contactless Smartcards using Low-cost Equipment," *Information Security Applications*, pp. 79–93, 2009.

[121] T. Souvignet and J. Frinken, "Differential Power Analysis as a Digital Forensic Tool," *Forensic Science International*, vol. 230, no. 1-3, pp. 127–136, 2013.

[122] R. Xu, L. Zhu, A. Wang, X. Du, K.-K. R. Choo, G. Zhang, and K. Gai, "Side-Channel Attack on a Protected RFID Card," *IEEE Access*, pp. 1–1, 2018.

[123] C. Kim, M. Schläffer, and S. Moon, "Differential Side Channel Analysis Attacks on FPGA Implementations of ARIA," *ETRI journal*, vol. 30, no. 2, pp. 315–325, 2008.

[124] S. Chari, J. R. Rao, and P. Rohatgi, "Template Attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2002, pp. 13–28.

[125] H. Saputra, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, R. Brooks, S. Kim, and W. Zhang, "Masking the Energy Behavior of DES Encryption," in *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*. IEEE Computer Society, 2003, p. 10084.

[126] T. Kim, S. Lee, D. Choi, and H. Yoon, "Protecting Secret Keys in Networked Devices with Table Encoding against Power Analysis Attacks," *Journal of High Speed Networks*, vol. 22, no. 4, pp. 293–307, 2016.

[127] M. Witteman and M. Oostdijk, "Secure Application Programming in the Presence of Side Channel Attacks," in *RSA Conference*, vol. 2008, 2008.

[128] Y. Ishai, A. Sahai, and D. Wagner, "Private Circuits: Securing Hardware against Probing Attacks," in *Annual International Cryptology Conference*. Springer, 2003, pp. 463–481.

[129] F.-X. Standaert, T. Malkin, and M. Yung, "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks," in *Eurocrypt*, vol. 5479. Springer, 2009, pp. 443–461.

[130] N. Golmie, O. Rebala, and N. Chevrollier, "Bluetooth Adaptive Frequency Hopping and Scheduling," in *Military Communications Conference (MILCOM'03)*, vol. 2. IEEE, 2003, pp. 1138–1142.

[131] V. Iyer, F. Hermans, and T. Voigt, "Detecting and Avoiding Multiple Sources of Interference in the 2.4 GHz Spectrum," in *12th European Conference on Wireless Sensor Networks (EWSN)*. Springer, 2015, pp. 35–51.

[132] M. R. Barron, "Creating Consumer Confidence or Confusion? The Role of Product Certification in the Market Today," *Marquette Intellectual Property Law Review*, vol. 11, no. 2, p. 413, 2007.

[133] Y.-i. Hayashi, "State-of-the-art Research on Electromagnetic Information Security," *Radio Science*, vol. 51, no. 7, pp. 1213–1219, 2016.

[134] "ISO/IEC 17825:2016 Testing Methods for the Mitigation of Non-invasive Attack Classes against Cryptographic Modules," iso.org, Accessed: 2018-01-21. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:17825:ed-1:v1:en

[135] "ISO/IEC TS 30104:2015 Physical Security Attacks, Mitigation Techniques and Security Requirements," iso.org, Accessed: 2018-01-21. [Online]. Available: https://www.iso.org/standard/56890.html

[136] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi *et al.*, "Test Vector Leakage Assessment (TVLA) Methodology in Practice," in *International Cryptographic Module Conference*, vol. 1001, 2013, p. 13.

[137] M. Marinov, "TempestSDR Remote Video Eavesdropping using a Software-defined Radio Platform," github.com, 2018, Accessed: 2018-02-01. [Online]. Available: https://github.com/martinmarinov/TempestSDR

[138] "ChipWhisperer Embedded Hardware Security Toolchain," newae.com, 2019, Accessed: 2020-07-29. [Online]. Available: https://www.newae.com/chipwhisperer

[139] C. O'Flynn and Z. D. Chen, "ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research," in *International Workshop*

on Constructive Side-Channel Analysis and Secure Design. Springer, 2014, pp. 243–260.

[140] Riscure, "Inspector SCA: Side-Channel Analysis Tool for Embedded Systems," riscure.com, 2019, Accessed: 2019-10-01. [Online]. Available: https://www.riscure.com/security-tools/inspector-sca/

[141] C. Ramsay and J. Lohuis, "White Paper: TEMPEST Attacks against AES: Covertly Stealing Keys for 200 Euros," Fox-IT, Netherlands, Tech. Rep., 2017. [Online]. Available: https://resources.fox-it.com/rs/170-CAK-271/images/Tempest_attacks_against_AES.pdf

[142] A. B. Blanco, J. de Fuentes, L. G. Manzano, L. H. Encinas, A. M. Munoz, J. R. Oliva, and I. S. Garcıa, "A Framework for Acquiring and Analyzing Traces from Cryptographic Devices," in *13th EAI International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2017.

[143] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith, "Rethinking SSL Development in an Appified World," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013, pp. 49–60.

[144] J. Zdziarski, *iPhone Forensics: Recovering Evidence, Personal Data, and Corporate Assets*. O'Reilly Media, Inc., 2008.

[145] A. Hoog, *Android Forensics: Investigation, Analysis and Mobile Security for Google Android*. Elsevier, 2011.

[146] B. Hay, M. Bishop, and K. Nance, "Live Analysis: Progress and Challenges," *IEEE Security & Privacy*, vol. 7, no. 2, 2009.

[147] C. J. Yang and A. P. Sample, "EM-Comm: Touch-based Communication via Modulated Electromagnetic Emissions," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, p. 118, 2017.

[148] M. Guri, A. Kachlon, O. Hasson, G. Kedma, Y. Mirsky, and Y. Elovici, "GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies," in *USENIX Security Symposium*, 2015, pp. 849–864.

[149] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "Digital Image Steganography: Survey and Analysis of Current Methods," *Signal Processing*, vol. 90, no. 3, pp. 727–752, 2010.

[150] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*. Springer Science & Business Media, 1997, vol. 408.

[151] D. Lillis, F. Breitinger, and M. Scanlon, "Hierarchical Bloom Filter Trees for Approximate Matching," *Journal of Digital Forensics, Security and Law*, vol. 13, no. 1, 01 2018.

[152] V. Corey, C. Peterman, S. Shearin, M. S. Greenberg, and J. Van Bokkelen, "Network Forensics Analysis," *IEEE Internet Computing*, vol. 6, no. 6, pp. 60–66, 2002.

[153] S. K. Goudos, I. T. Rekanos, and J. N. Sahalos, "EMI Reduction and ICs Optimal Arrangement Inside High-Speed Networking Equipment Using Particle Swarm Optimization," *IEEE Transactions on Electromagnetic Compatibility*, vol. 50, no. 3, pp. 586–596, 2008.

[154] M. Schulz, P. Klapper, M. Hollick, E. Tews, and S. Katzenbeisser, "Trust the Wire, They Always Told Me!: On Practical Non-destructive Wire-tap Attacks Against Ethernet," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016, pp. 43–48.

[155] W. Entriken, "System Bus Radio," github.com, Accessed: 2018-01-26. [Online]. Available: https://github.com/fulldecent/system-bus-radio

[156] W. Liu, K. Huang, X. Zhou, and S. Durrani, "Full-Duplex Backscatter Interference Networks Based on Time-Hopping Spread Spectrum," *IEEE*

Transactions on Wireless Communications, vol. 16, no. 7, pp. 4361–4377, July 2017.

[157] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient Backscatter: Wireless Communication out of Thin Air," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 39–50, 2013.

[158] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall, "Wi-Fi Backscatter: Internet Connectivity for RF-powered Devices," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 607–618.

[159] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti, "BackFi: High Throughput WiFi Backscatter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 283–296, 2015.

[160] P. Zhang, D. Bharadia, K. Joshi, and S. Katti, "Hitchhike: Practical Backscatter using Commodity WiFi," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems*. ACM, 2016, pp. 259–271.

[161] J. Z. Kolter and M. A. Maloof, "Learning to Detect Malicious Executables in the Wild," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2004, pp. 470–478.

[162] P. Saboia, T. Carvalho, and A. Rocha, "Eye Specular Highlights Telltales for Digital Forensics: A Machine Learning Approach," in *18th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2011, pp. 1937–1940.

[163] S. Mukkamala and A. H. Sung, "Identifying Significant Features for Network Forensic Analysis using Artificial Intelligent Techniques," *International Journal of Digital Evidence*, vol. 1, no. 4, pp. 1–17, 2003.

[164] L. Lerman, G. Bontempi, and O. Markowitch, "Side Channel Attack: An Approach Based on Machine Learning," in *Proceedings of 2nd International Workshop on Constructive Side-Channel Analysis and Security Design (COSADE)*. Schindler and Huss, 2011, pp. 29–41.

[165] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking Cryptographic Implementations using Deep Learning Techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26.

[166] J. Williams, "ACPO Good Practice Guide for Digital Evidence," Association of Chief Police Officers, United Kingdom, Tech. Rep., 2012. [Online]. Available: https://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf

[167] Á. MacDermott, S. Lea, F. Iqbal, I. Idowu, and B. Shah, "Forensic Analysis of Wearable Devices: Fitbit, Garmin and HETP Watches," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2019, pp. 1–6.

[168] M. O. Ojo, S. Giordano, G. Procissi, and I. N. Seitanidis, "A Review of Low-end, Middle-end, and High-end IoT Devices," *IEEE Access*, vol. 6, pp. 70 528–70 554, 2018.

[169] "Raspberry Pi 3 Model B+," raspberrypi.org, 2020, Accessed: 2020-07-11. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[170] "Arduino Leonardo," arduino.cc, 2020, Accessed: 2020-07-11. [Online]. Available: https://www.arduino.cc/en/Main/Arduino_BoardLeonardo

[171] "Raspberry Pi OS," raspberrypi.org, 2020, Accessed: 2020-07-11. [Online]. Available: https://www.raspberrypi.org/downloads/raspberry-pi-os/

[172] C. Meffert, D. Clark, I. Baggili, and F. Breitinger, "Forensic State Acquisition from Internet of Things (FSAIoT): A General Framework and

Practical Approach for IoT Forensics through IoT Device State Acquisition," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM, 2017, p. 56.

[173] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 195–212.

[174] E. Ronen and A. Shamir, "Extended Functionality Attacks on IoT Devices: The Case of Smart Lights," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 3–12.

[175] B. Cheng and D. M. Titterington, "Neural Networks: A Review from a Statistical Perspective," *Statistical Science*, pp. 2–30, 1994.

[176] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[177] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to Time Series Predictable through Time-window Approaches," in *Neural Nets WIRN Vietri-01*. Springer, 2002, pp. 193–200.

[178] A. Ben-Hur and J. Weston, "A User's Guide to Support Vector Machines," in *Data mining techniques for the life sciences*. Springer, 2010, pp. 223–239.

[179] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the Mirai Botnet," in *USENIX Security Symposium*, 2017, pp. 1092–1110.

[180] "Novelty and Outlier Detection," scikit-learn.org, 2020, Accessed: 2020-08-20. [Online]. Available: https://scikit-learn.org/stable/modules/outlier_detection.html

[181] "TCP Protocol - Ubuntu Manual Pages," ubuntu.com, Accessed: 2020-10-14. [Online]. Available: http://manpages.ubuntu.com/manpages/cosmic/man7/tcp.7.html

[182] A. Liaw, M. Wiener *et al.*, "Classification and Regression by Random Forest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[183] E. I. Altman *et al.*, "Predicting Financial Distress of Companies: Revisiting the Z-score and ZETA Models," *Stern School of Business, New York University*, pp. 9–12, 2000.

[184] I. T. Jolliffe and J. Cadima, "Principal Component Analysis: A Review and Recent Developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[185] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene Selection for Cancer Classification using Support Vector Machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[186] N. I. Sapankevych and R. Sankar, "Time Series Prediction using Support Vector Machines: A Survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, 2009.

[187] H. Ponce, L. Miralles-Pechuán, and M. Martínez-Villaseñor, "A Flexible Approach for Human Activity Recognition using Artificial Hydrocarbon Networks," *Sensors*, vol. 16, no. 11, p. 1715, 2016.

[188] E. Casey, "Standardization of Forming and Expressing Preliminary Evaluative Opinions on Digital Evidence," *Forensic Science International: Digital Investigation*, p. 200888, 2020.

[189] "Cyber-investigation Analysis Standard Expression (CASE)," caseontology.org, 2020, Accessed: 2020-07-29. [Online]. Available: https://caseontology.org/

[190] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An Overview of the HDF5 Technology Suite and its Applications," in *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*. ACM, 2011, pp. 36–47.

[191] T. Cooklev, R. Normoyle, and D. Clendenen, "The VITA-49 Analog RF-digital Interface," *IEEE Circuits and Systems Magazine*, vol. 12, no. 4, pp. 21–32, 2012.

[192] B. Hilburn, N. West, T. O'Shea, and T. Roy, "SigMF: The Signal Metadata Format," in *Proceedings of the GNU Radio Conference*, vol. 3, no. 1, 2018.

[193] M. Cohen, S. Garfinkel, and B. Schatz, "Extending the Advanced Forensic Format to Accommodate Multiple Data Sources, Logical Evidence, Arbitrary Information and Forensic Workflow," *Digital Investigation*, vol. 6, pp. S57–S68, 2009.

Appendix A

List of Abbreviations

Acronyms

ADC Analog-to-Digital Conversion.

AES Advanced Encryption Standard.

AI Artificial Intelligence.

AM Amplitude Modulation.

API Application Programming Interface.

BLE Bluetooth Low Energy.

CASE Cyber-investigation Analysis Standard Expression.

CCTV Closed-circuit Television.

CEMA Correlation Electromagnetic Analysis.

CPA Correlation Power Analysis.

CPU Central Processing Unit.

CRT Cathode-ray Tube.

DEMA Differential Electromagnetic Analysis.

DES Data Encryption Standard.

DoS Denial of Service.

DPA Differential Power Analaysis.

DRM Digital Rights Management.

DSP Digital Signal Processing.

DUT Device under Test.

ECC Elliptic Curve-based Cryptography.

ECDH Elliptic Curve-based Diffie Hellman.

ECDSA Elliptic Curve-based Digital Signature Algorithm.

EM Electromagnetic.

EM-SCA Electromagnetic Side-Channel Analysis.

EMC Electromagnetic Compatibility.

EU European Union.

FASE Finding Amplitude-modulated Side-channel Emanations.

FCC Federal Communications Commission.

FDA Food and Drug Administration.

FFT Fast Fourier Transform.

FPGA Field-programmable Gate Arrays.

GPIO General Purpose Input/Output.

GRC GNU Radio Companion.

GSM Global System for Mobile Communications.

GUI Graphical User Interface.

I/O Input/Output.

I/Q In-phase/Quadrature-phase.

IC Integrated Circuit.

IEC International Electrotechnical Commission.

IoT Internet of Things.

IP Internet Protocol.

LED Light-emitting Diode.

LSTM Long Short-term Memory.

MAC Medium Access Control.

MCU Microcontroller Unit.

MDL Multiple Discriminant Analysis.

ML Machine Learning.

ML Maximum Likelihood.

MLP Multi-layer Perceptron.

OTA Over-the-Air.

PCA Principal Component Analysis.

PCB Printed Circuit Board.

PLC Programmable Logic Controllers.

PSD Power Spectral Density.

RADAR Radio Azimuth Direction and Ranging.

RAM Random Access Memory.

RF Random Forests.

RF Radio Frequency.

RF-DNA Radio Frequency Distinct Native Attributes.

RFE Recursive Feature Elimination.

RFID Radio Frequency Identification.

RNN Recurrent Neural Networks.

ROC Receiver Operating Characteristic.

RSA Rivest-Shamir-Adleman.

SAVAT Signal AVailability for an ATtacker.

SD Secure Digital.

SDR Software-defined Radio.

SEM Scanning Electron Microscopes.

SEMA Simple Electromagnetic Analysis.

SMA SubMiniature version A.

SMS Short Message Service.

SNR Signal-to-Noise Ratio.

SoC System-on-Chip.

SPA Simple Power Analysis.

SSH Secure Shell.

SSL Secure Socket Layer.

STFT Short-term Fourier Transform.

SVM support vector machines.

TCP Transmission Control Protocol.

TTL Time-to-Live.

TV Television.

TVLA Test Vector Leakage Assessment.

UART Universal Asynchronous Receiver/Transmitter.

UDP User Datagram Protocol.

USB Universal Serial Bus.

USRP Universal Software Radio Peripheral.

VLSI Very Large-scale Integration.

VM Virtual Machines.

WBC White Box Cryptography.

XSS Cross-site Scripting.

Appendix B

EMvidence User Documentation

About

EMvidence is a tool that can be used to gather insights from electromagnetic (EM) side-channel radiation of IoT devices. Users can capture EM traces of a device-under-test (DUT) through *EMvidence* using a software defined radio (SDR) hardware. Additionally, users can upload EM traces that are captured through other means into *EMvidence* as well. An EM trace can be analysed to gather insights of the DUT by enabling various *EMvidence* plug-ins. Some *EMvidence* plug-ins are provided by the developer while users have the freedom to build third-party plug-ins according to their needs.

Installation

System Requirements:

- The computer must have at least 8 GB of RAM and sufficient disk space in order to store and process EM traces.
- The Ubuntu operating system is most preferred and the instructions assume so. If you are using something else, please check how to install the required packages on your preferred system.

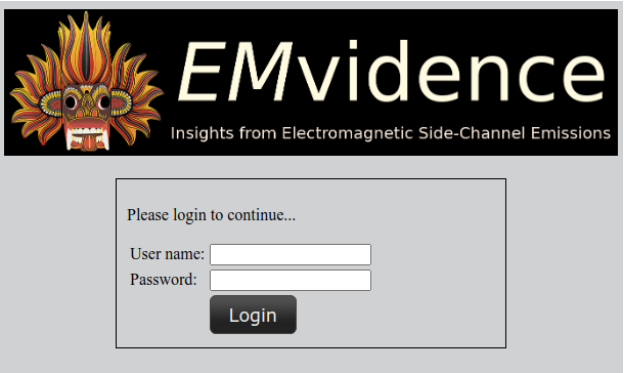

Figure B.1: Login window of the *EMvidence* framework.

- An Internet connection to download and install required packages.

Installing Required Packages:

- Install Miniconda for Python 3.7, which is required to install other important packages.
- Now, open a terminal and run the following commands to install required packages through conda.

```
conda install -c ryanvolz gnuradio
conda install -c conda-forge weasyprint
```

Downloading and Running:

Run the following commands on a terminal.

```
git clone https://github.com/asanka-code/EMvidence.git
cd EMvidence/EMvidence/
./start.sh
```

Now, open a web browser and goto the URL `http://0.0.0.0:5000`. You can login by using `emvidence` as the username and password (see Figure B.1).

Usage of *EMvidence*

Dashboard

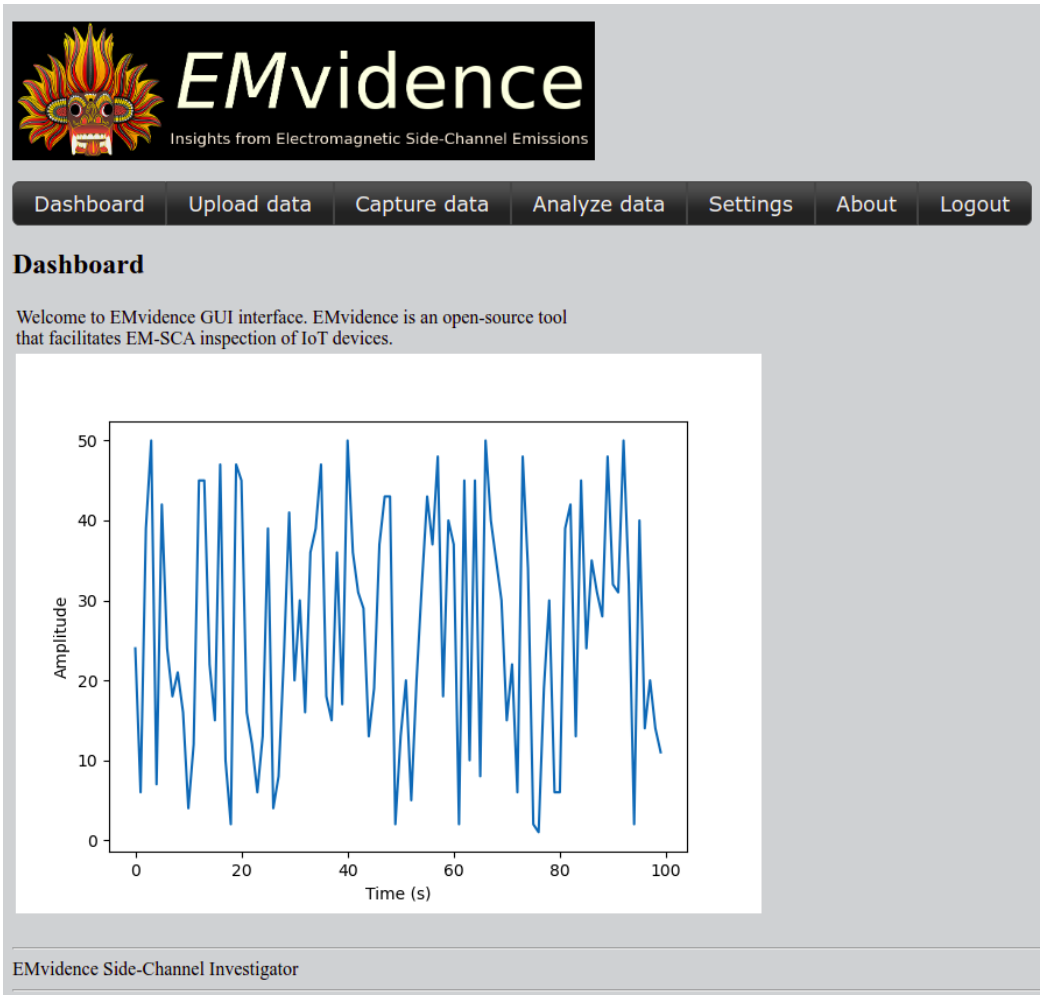

Figure B.2: Dashboard interface of the *EMvidence* framework.

Once logged in, the user is sent to the dashboard of the *EMvidence* GUI interface (see Figure B.2). The user can navigate between different options such as uploading separately acquired EM data, capturing new EM data, analysing EM data and also adjusting the default settings of the system through the menu bar. In future versions of *EMvidence*, the dashboard interface will be improved to provide a summary of the currently stored investigative EM data, important

system settings and also the details of SDR hardware currently connected to the system.

Uploading Data

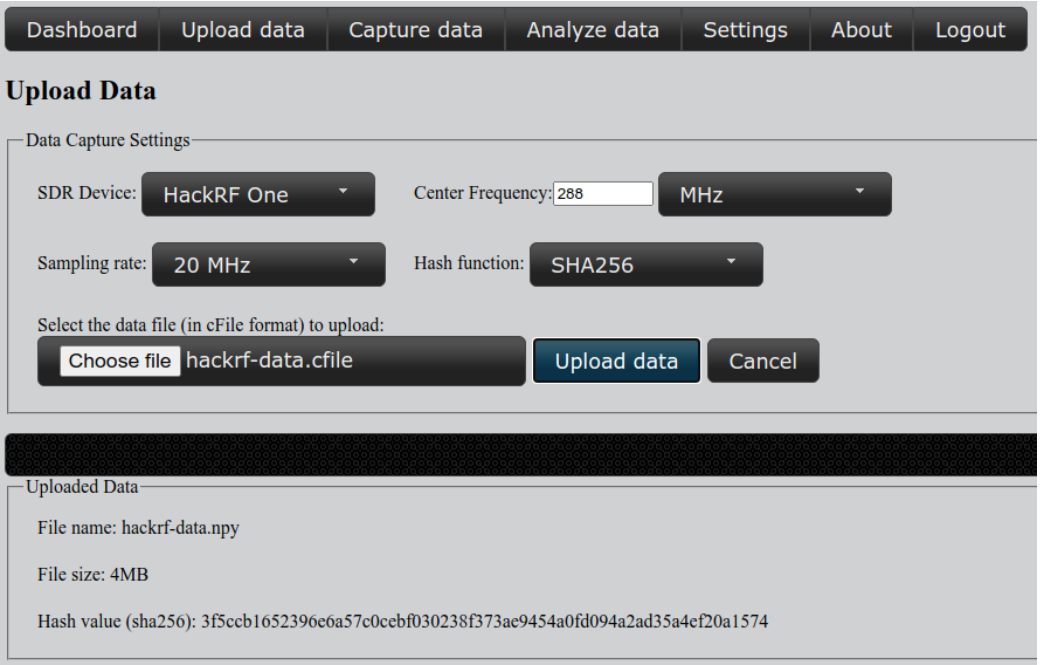

Figure B.3: The interface for uploading EM data into *EMvidence* framework.

If there exists any EM data previously acquired from a different software and/or hardware setup, the *EMvidence* framework provides the facility to upload such data into the *EMvidence* for analysis. This can be done by clicking on the *Upload data* menu and going to the data uploading window (see Figure B.3).

In order to upload an EM data file to *EMvidence*, the data should be in I/Q format stored with the file extension `.cfile`. Once the upload is complete, *EMvidence* automatically convert the data into `.npy` format for easier and efficient processing. When uploading EM data, multiple parameters related to the data must be selected in the menu so that the data can be properly interpreted later. These parameters are namely, the SDR device used to capture the data, centre frequency of data, and sample rate. Furthermore, a *hash* function, e.g.,

MD5, SHA1, SHA256, should be selected to calculate and store a hash value along with the EM data file.

Capturing Data

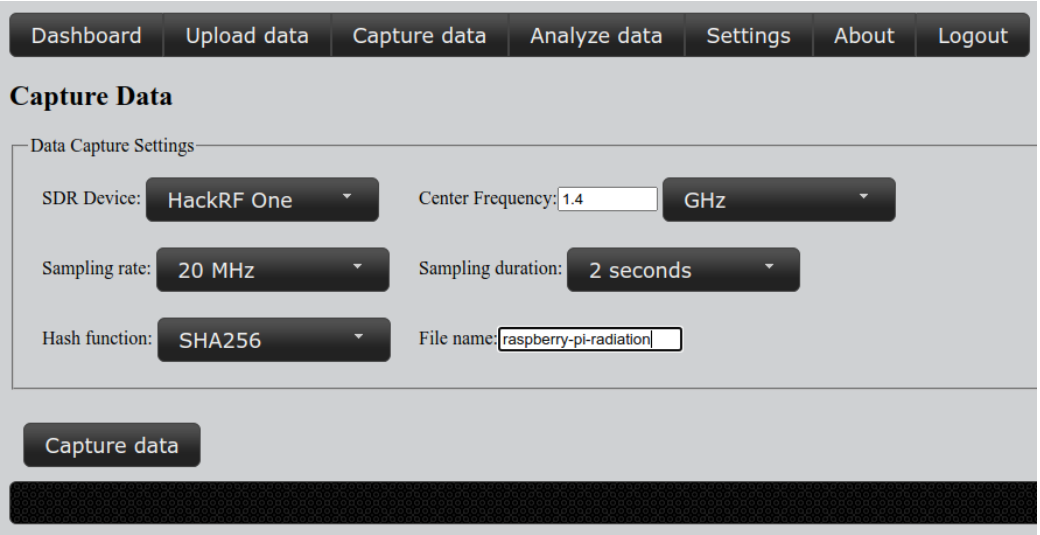

Figure B.4: The interface for capturing EM data using the *EMvidence* framework.

Capturing EM data from IoT devices can be performed on *EMvidence* by going through the *Capture data* interface (see Figure B.4). First of all, a suitable SDR hardware should be connected to the computer where the *EMvidence* framework is running. *EMvidence* currently supports HackRF and RTL-SDR devices, and will support many more devices in the future. When the SDR device is ready and its antenna is placed closer to the IoT device being investigated, the user can adjust the data capture settings in the *EMvidence* interface. These capture settings include the SDR device, centre frequency, sample rate, sample duration, hash function, and a file name to store the data.

Analysing Data

Finally EM data can be analysed on *EMvidence* by using a wide variety of plug-ins. In the *Analyse data* window, *EMvidence* provides the facility to select

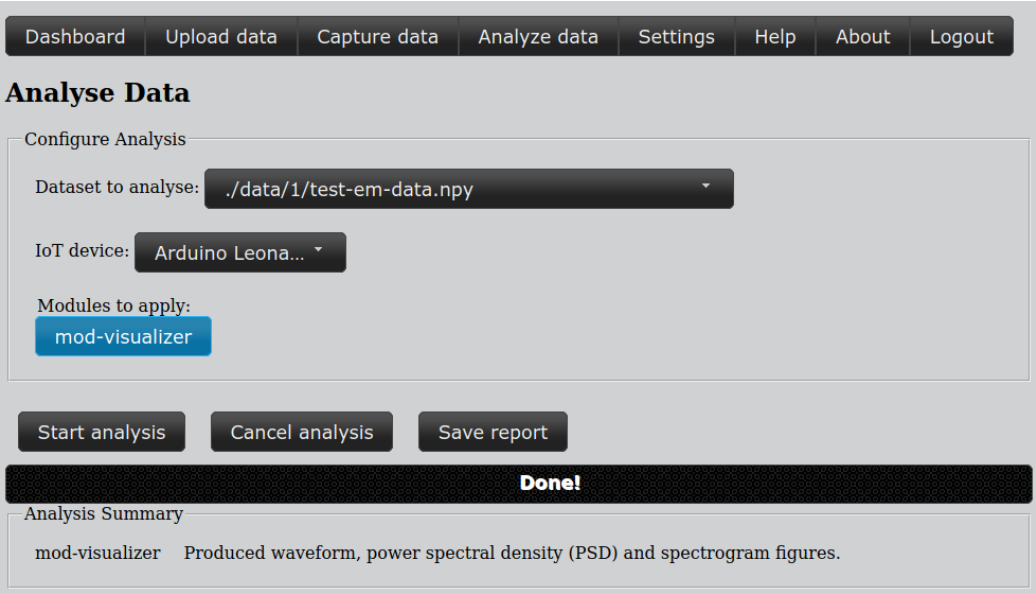

Figure B.5: The interface for analysing EM data using plug-ins on the *EMvidence* framework.

a dataset acquired by or uploaded to the system previously (see Figure B.5). Depending on the specific IoT device being analysed, the user can select one or more plug-ins for the analysis from this window. After selecting the plug-ins, *Start analysis* button should be clicked to start the analysis. Once the analysis is compete, the report can be opened as a PDF file, which contains the details of the EM dataset and the output of each plug-in.

Settings

The settings window of *EMvidence* provides multiple facilities to make adjustments to the whole system (see Figure B.6). Most importantly, whenever a new plug-in is developed or obtained from a third-party, it can be integrated into *EMvidence* by uploading the plug-in as a `.zip` file from the settings window. Similarly, adding and removing new IoT device types can be done through the same settings interface. Finally, whenever an EM dataset needs to be removed from the system, it can be selected and deleted from this window.

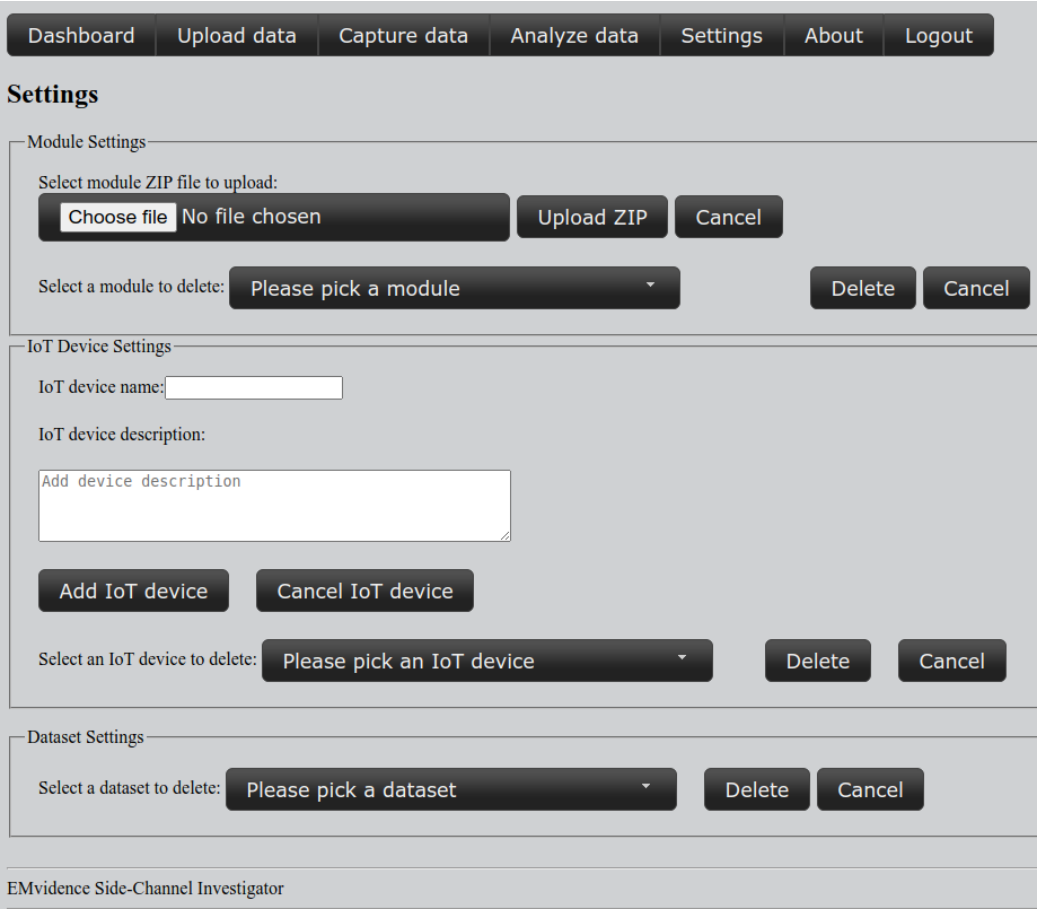

Figure B.6: Settings interface of the *EMvidence* framework.

Plug-in Development

Plug-ins that are shipped by default with *EMvidence* are located at the `backyard` directory[1] in the source code directory. In order to assist with research and development of new plug-ins for *EMvidence*, the skeleton code for a plug-in is provided in the same directory.

Please refer to the *EMvidence* framework's source code repository in Github[2] for the latest updates.

[1] `https://github.com/asanka-code/EMvidence/tree/master/backyard`
[2] `https://github.com/asanka-code/EMvidence`