Side-Channel Attacks on Elliptic Curve Cryptosystems Based on Machine Learning Techniques

By

Seyed Ehsan Saeedi Shahri

A thesis submitted to Macquarie University
for the degree of
Doctor of Philosophy
Department of ENGINEERING
July 2016

MACQUARIE
University
SYDNEY·AUSTRALIA

Typeset in LaTeX 2_ε.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Seyed Ehsan Saeedi Shahri

Dedicated to

My parents, whose unconditional love and encouragement allowed me to start and complete this journey.

Acknowledgements

I would like to express my special appreciation and thanks to my adviser Dr.Yinan Kong; you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research and on my career have been invaluable.

I would also like to thank Professor Michael Heimlich, my co-supervisor, for his excellent support. I am thankful to Selim Hossain, Shahzad Asif and my other group mates for your brilliant comments and suggestions. I am grateful to Dr Keith Imrie for valuable advice and useful comments that improved the quality of this thesis. I wish to acknowledge Macquarie University for awarding me an International Macquarie University Research Excellence Scholarship (iMQRES), and providing financial support to attend national and international conferences during this project.

List of Publications

Publications related to this thesis of which the author is the first-author are as follows.

- E. Saeedi, Y. Kong, and M. S. Hossain, "Feed-Forward Back-Propagation Neural Networks in Side-Channel Information Characterization", *IEICE TRANSACTIONS on Communications.* (Under major revision)

- E. Saeedi, M. S. Hossain, and Y. Kong, "Side Channel Information Characterization based on Cascade Feed-Forward Back-Propagation Neural Network", *Journal of Electronic Testing, May 2016, Springer.* (In press)

- E. Saeedi, Y. Kong, and M. S. Hossain, "Side Channel Attacks and Learning Vector Quantization", *Frontiers of Information Technology and Electronic Engineering, Springer, 2016.*(In press)

- E. Saeedi and Y. Kong, "Side-channel Vulnerabilities of Automobiles", *Transaction on IoT and Cloud Computing 2.2 (2014): pp.1-8.*

- E. Saeedi, M. S. Hossain, and Y. Kong, 'Multi-class SVMs Analysis of Side-Channel Information of Elliptic Curve Cryptosystem'", *2015 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), pp. 1-6, Chicago, IL, USA, July 26-29, 2015.*

- E. Saeedi, M. S. Hossain, and Y. Kong, "Side Channel Analysis of an Elliptic Curve Crypto-system Based on Multi-Class Classification", *The Sixth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Denton, Texas, USA, pp. 1-7, July 13-15, 2015.*

- E. Saeedi and Y. Kong, "Side Channel Information Analysis Based on Machine Learning", *In Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on (pp. 1-7). IEEE.*

- E. Saeedi, and Y. Kong, "Fuzzy Analysis of Side Channel Information", *In Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on (pp. 1-5). IEEE.*

- E. Saeedi, and Y. Kong, "Support Vector Machines in Side Channel Analysis", *the International Symposium on Information Theory and Its Applications. ISITA2014.*

Publications related to the field of this thesis for which the author is one of the co-authors are as follows.

- M. S. Hossain, Y. Kong, E. Saeedi, and N. C. Vayalil, "High-Performance Elliptic Curve Cryptography Processor over NIST Prime Field", *IET Computers and Digital Techniques.*
- M. S. Hossain, Y. Kong and E. Saeedi, "High-Performance FPGA Implementation of Elliptic Curve Cryptography Processor over Binary Field GF (2^{163})", .
- M. S. Hossain, Y. Kong and E. Saeedi, "High-Speed, Area-Efficient, FPGA-Based Elliptic Curve Cryptographic Processor over NIST Binary Fields", *2015 IEEE International Conference on Data Science and Data Intensive Systems (DSDIS), UTS, Sydney, Australia, pp. 175-181, December 11-13, 2015.*

Abstract

This thesis presents a promising approach to Side-channel attacks on the cryptosystems based on elliptic curve cryptography (ECC). This approach is based on machine-learning analysis in characterisation of side-channel information. The original contributions of this thesis is to verify the performance of machine-learning techniques in terms of neural networks (NN), support vector machines (SVM) and principal component analysis (PCA). In this project, PCA is used as a powerful algorithm in the preprocessing stage to decrease the computational complexity of the input dataset, while SVM and NN are utilised as efficient multi-class classifiers to recognise and classify different patterns of side-channel information.

In order to investigate the proposed method, an experiment based on the power consumption and electromagnetic emission of an field-programmable gate array (FPGA) implementation of ECC was conducted. Regarding our experimental results based on an FPGA implementation of ECC, PCA can be used as a strong preprocessing stage to reduce the signal-noise ratio, data-set dimension and algorithm complexity. In addition, after verifying the performance of different techniques and specifications such as kernel functions, neural-network architect and parameters, we inferred that the most efficient machine-learning techniques for side-channel information characterisation are LVQ neural network (with a number of hidden layers between 90 and 100), and SVM with Gaussian RBF kernel function with parameter p value of 5 and 50 for **CS** and **M-SVM**^{2} SVM models respectively with about 80 to 85 % accuracy.

Contents

List of Figures

List of Tables

1

Introduction

Nowadays, tiny electronic devices that have been inserted in a lot of different applications play an important role in our modern life. Some of these components embed a complete computer with its memories, analogue blocks and arithmetic logic unit (ALU), and are used to obtain a secure application: ATM, SIM card (cell phone), ID card, social-security card, identification, signature, and many others. Until the mid 90s, these smart cards were regarded as black boxes and the cryptographic algorithms implemented inside were considered to be the only security needed to ensure the confidentiality of the associated application. Indeed, the cryptosystems often prove to be secure enough, and in most of these systems the security relies only on the algorithm (i.e., the way the secret key is mixed with the messages).

These cryptosystems, even after recent improvements in mathematical cryptography algorithms, are still vulnerable to Side-Channel Attacks (SCA) which have been found to be a powerful class of attack against all implementations of cryptographic algorithms. Side-channel attacks exploit information that is unintentionally leaked during the execution of a cryptographic algorithm on a cryptosystem. In this context, useful information can often be obtained from side channels such as: processing time, power consumption and electromagnetic emanation.

This thesis will propose a powerful and promising method of SCA based on machine-learning techniques in the forms of Neural Networks (NN), Support-Vector Machines (SVM) and Principal Component Analysis (PCA). In order to investigate the performance of different techniques, an experimental investigation was conducted based on an FPGA implementation of elliptic-curve cryptography (ECC).

1.1 Existing Literature

The literature regarding SCA is sparse. An extensive literature search results in only a few tens of hits, and many of them are not directly relevant to the interests of this thesis. The literature which has been deemed to be directly relevant to our topic is discussed below.

Over the past decade there has been a dramatic increase in various applications and implementations of SCA. Since SCAs can generally be performed using relatively cheap equipment, they pose a serious threat to the security of most cryptographic hardware devices. Such devices range from personal computers to small embedded devices such as smart cards and RFIDs (radio-frequency identification devices). In [12] a novel authentication protocol is introduced for security enhancement and eliminating weaknesses of previous implementations. In [13], Li and Lee improved a secure scheme and claimed that it is secure against smart-card-loss attack. Later on, [14] a robust scheme is presented to cope with the defects of the Li-Lee's scheme, while keeping the merits of different password authentication schemes using smart cards. Furthermore, an improved dynamic ID-based authentication scheme was proposed to remedy previous security flaws [15]. [16] presents three principles that are helpful to explain many of the security failures repeated in the past and important for designing more robust schemes in the future.

In addition to the application of SCA, a considerable literature has grown up around the theme of different approaches of side-channel information analysis. It was first shown by Kocher [17], where a timing analysis was used to recover the key from an exponential key processing operation. Coron was the first to report a simple power analysis (SPA) on ECC [18], then SPA on unified formulae introduced by Walter [19] and improved by Stebila and Thriault [20]. This latter attack targets the indistinguishable point operation formulae countermeasure. The more powerful and general form of attack termed differential power analysis (DPA) was introduced in [21]. Later, these differential attack techniques are applied in a differential electromagnetic attack (DEMA) by Quisquater etal. in [22] and [7]. Goubin proposed Refined Power Analysis (RPA) based on the apparition of the particular point during elliptic-curve scalar multiplication (ECSM) [23], and an extension of his work was presented by Akishita and Takagi [24]. Also some successful attacks have been mounted on different cryptosystems such as ECC [7, 25] or RSA [26]. Moreover, several attempts have been made to exploit side-channel information through a profiling-based attack called template attack [27]. In this attack a training device which is fully controllable and accessible is utilised within a training phase to gain additional knowledge for the attack against an identical target device [28, 29]. In [30] machine learning was introduced as a powerful type of profiling-based side-channel attack from an information-theoretical point of view. A number of studies ([30–33]) used support-vector machines (SVMs) as powerful classifiers to classify different patterns of side-channel information. More recent studies have confirmed that neural networks have emerged as a powerful tool to solve classification and pattern recognition problems, and they can be considered as a promising alternative to various conventional classification methods; see [34, 35].

Numerous studies have attempted to address countermeasures against the conventional side-channel attacks; most of the proposed approaches are performed based on implementations of a cryptography algorithm with constant or randomised execution times or execution order (also known as shuffling) to make the occurrence of the leakage unpredictable [18, 36–38]. Brier and Joye proposed a countermeasure based on indistinguishable points operation formulae [39]. Elliptic-curve operations are reviewed so that the operations for computing a doubling and an addition are the same, in another approach formulae to perform a doubling and an addition are rewritten into sequences of identical patterns [40, 41]. In terms of timing attacks on asymmetric cryptosystems, most of them are based on particular implementations of Montgomery multiplication in which a final reduction step is sometimes needed [17]. By adding extra words to the manipulated integers, the final subtraction operation can be avoided [42, 43]. Sato, Schepers and Takagi also introduced another timing attack based on the final reduction of Montgomery multiplication [44].

While several countermeasures against conventional attacks have been proposed, cryptosystems are still vulnerable to SCA because some inherent leakage during single executions in a cryptography algorithm cannot be prevented in many cases; for example location-based leakage [45], address-bit leakage [46], or operation-dependent leakage [47]. Furthermore, most of the countermeasures have a negative effect, sometimes significant, on the performance of cryptosystems [48] or the cost of implementation [36].

1.2 Motivation for this Research

The security of the embedded systems plays an important roles in designing electronic devices. It is well known fact that the U.S government has spent considerable resources in the classified TEMPEST program in order to prevent the leakage of sensitive information through electromagnetic radiation since the 1950s. In that time, the world of cryptographers and related industrial companies which design and fabricate cryptographic tokens, such as smart cards, was shocked when some side-channel analysers demonstrated that they were able to extract the keys of several widely used smart cards by power consumption analysis. As a result, the exciting research area of Side-Channel Attacks (SCA) was born. SCA makes one important point clear: a real-world security system always consists of various layers. If cryptographers, software developers and hardware designers fail to cooperate and do not mutually check each other's work, such a security system is very likely to display some inherent vulnerability. Hence the interaction of different trained and skilled people plays a profound role during the design process of a security system.

Even after proposing several countermeasures against conventional attacks, some inherent leakages during single executions in a cryptography algorithm cannot be prevented in many cases, for example, location-based leakage [45], address bit leakage [46], or operation-dependent leakage [47]. In addition to the drawbacks of countermeasures, most of them have a negative effect, sometimes significant, on the performance of cryptosystems [48] or the cost of implementation [36]. Recent literature has shown

that the safety of cryptosystems has always been a challenge, since both attacks and countermeasures interact strongly, as countermeasures get broken by improved attacks and new countermeasures are developed to thwart ever more advanced attacks. In addition, recent work brought out approaches that produce comparable results, but are still not optimal. This indicates a need to verify the various approaches of SCA and highlight the cryptosystem vulnerability.

1.3 Research Objective

The main issues of SCA are accuracy of finding key-bit values, algorithm processing-time, memory consumption and cost of implementation for real-time attacks. This thesis seeks to remedy these problems by using machine-learning techniques. The aim of this thesis is to propose a promising method of SCA by investigating the performance of different multi-class classifiers and pattern-recognition algorithms in analysis of the side-channel information of an ECC-cryptosystem.

1.4 Research Contribution

The main contribution of this thesis is to improve side channel attacks (SCA) and investigate the potential physical vulnerabilities of cryptosystems. For this purpose, a powerful method of information analysis based on machine-learning techniques are utilised. In this thesis, machine-learning techniques are used in the forms of Neural Networks (NN), Support Vector Machines (SVM) and Principal Component Analysis (PCA). PCA is used to address a common problem of SCA; i.e. handling the huge amount of data collected in physical measurements, while SVM and NN are applied as a promising multi-class classifiers. To verify the performance of the proposed method, an experiment was conducted on FPGA implementation of ECC, in which the specifications of the proposed method, such as different models of classification, network architectures, kernel functions and parameters, which can significantly affect the performance of classifiers, are verified and set for an efficient and reliable side channel analysis system.

1.5 Research Outcome

Regarding our experimental results based on an FPGA implementation of elliptic curve cryptography (ECC), PCA can be used as a strong preprocessing stage to reduce the signal-noise ratio, dataset dimension and algorithm complexity. In addition, the most efficient machine-learning techniques for side-channel information characterisation with about 80 to 85% accuracy are LVQ neural networks (with a number of hidden layers between 90 and 100), and SVM with Gaussian RBF kernel function with a parameter p value of 5 and 50 for CS and M-SVM^2 SVM models respectively.

1.6 Thesis Outline

The thesis is organised as follows:

- Chapter 2: "Overview of Side-Channel Cryptanalysis" presents the necessary background and preliminary studies of the most common approach of cryptosystem implementation, physical side-channel measurements, side channel analysis and new approaches to improve and expand the conventional algorithms.
- Chapter 3: "Elliptic-Curve Cryptosystem Implementation" devoted to the implementation and considerations of an ECC cryptosystem based on an FPGA.
- Chapter 4: "Preprocessing Stage" presents Principal Component Analysis (PCA) as a preprocessing stage that addresses the difficulty, in side-channel information analysis, in how to handle the huge input dataset.
- Chapter 5: "Support-Vector Machine as a Side-Channel Classifier" dedicate to support-vector-machine (SVM) as a powerful and robust algorithm for characterisation of side-channel information. Moreover, different models of classification, kernel functions and parameters, which can significantly affect the performance of classifiers, are investigated.
- Chapter 6: "Neural Networks as Side-Channel-Information Classifiers" discusses the characterisation of side-channel information based on the most powerful neural networks in multi-class classification. For this purpose, the strengths and weaknesses of different neural-network architectures and their parameters are evaluated.
- Chapter 7: "Thesis Conclusion and future work" is devoted to thesis conclusion and recommendation for future works.

1.7 Tools

The software used in this thesis is:

- MATLAB R2015a
- Xilinx ISE Project Navigator 14.5
- Vivado 2015.2
- MSVMpack 1.5

The hardware used in this thesis is:

- FPGA board
- Oscilloscope

- Current probe
- Electromagnetic probes
- Amplifier

2

Overview of Side-Channel Cryptanalysis

2.1 Introduction

In a world where people's lives are increasingly entangled with the use of ubiquitous embedded devices, the threat of malicious abuse of data continues to grow at a high pace. Side-channel cryptanalysis is a branch of cryptography in which sensitive information is gained from the physical implementation of a target cryptosystem. This is in contrast with other forms of cryptanalysis where the algorithms and their underlying computational problems are attacked. All electronic devices leak information in a multitude of ways. Prominent examples of this are temperature, power consumption, time taken for computations, acoustics and electromagnetic emanations. In general, these types of information leakage may be tied in some way to the types of operations that the cryptographic algorithm is performing. This makes them a very powerful tool for gaining the desired information from the cryptosystem. All of the aforementioned leakages are passive, meaning that we only capture power consumption information as it normally comes off of the target device. Further, this type of attack requires no active manipulation of the device by the adversary.

In this chapter the necessary background and preliminary studies of the most common approach of cryptosystem implementation, physical side-channel measurements, side-channel analysis and new approaches to improve and expand the conventional algorithms, are provided. It aims to provide an insight into the characteristics of commonly used side-channel cryptanalysis algorithms, categorising and verifying the performance of the different approaches.

2.2 Application Fields

There are numerous applications of embedded cryptography; in this section, we describe present and future applications in order to show their growing involvement in our lives. The more devices are exchanging electronic messages, the more important becomes cryptography in our everyday life. Today, even though it is not always visible, cryptography is omnipresent.

- **Networks.** The age of electronic communication is only at its beginning: high speed networks are spreading, the price of DSL connections is falling, and wireless LANs are expected to develop fast in the near future. Intel developed the Centrino technology, which comprises wireless LAN antennas; IBM Bluetooth is also an important actor in the field of wireless communication. While the wireless technology is very interesting in terms of end-user flexibility and costs, its security is a critical problem because anyone can easily eavesdrop on radio communications. Therefore, encryption, authentication and integrity must be guaranteed [49, 50].

- **Electronic Cash.** Smart cards are nowadays common; however, the current technology is not always able to resist attacks from personal computers, which are more and more powerful. In 1998, the French hacker Serge Humpich managed to create a fake and functional smart card. Nowadays, it is very easy to find resources explaining how to create such cards with little money, a programmable smart card, a card writer and a personal computer. Therefore, increasing the security of smart cards for electronic cash is a critical issue. Harsh legislation again hackers and high secrecy are not sufficient to convince the end-user that electronic cash is secure compared to standard methods of payment. Since they have been extensively studied and used for years, public-key cryptosystems provide sufficient security if the key length is appropriately chosen [51, 52].

- **SIM Cards.** With the economic boom of cell phones, SIM cards had been a buoyant market. They authenticate a cell phone user in order to make out a bill of his communications. Therefore, authentication and non-repudiation are the main security concepts involved here. Encryption is also an important issue from the user's point of view. SIM cards should be able to reliably perform these tasks, but this is not the case: complaints about unexpected bills from cell-phone users are common. Here again, public-key cryptography and digital signatures are the next challenges for the SIM card industry [53].

- **ID Cards.** Governmental projects of electronic ID cards are in progress: in Japan, electronic ID cards allowing contact-less radio transactions, with a 32-bit CPU and a coprocessor handling encryption and digital signature, will be issued in the near future. The threat of terrorism, falsification of passports or ID cards is critical, and the new passports will be equipped with magnetic bands allowing easy and instant identification; however electronic ID cards are much harder to copy or falsify and provide greater security. Besides, administrative delay and

costs could be consequently reduced, for example for health-care reimbursements [50].

- **White Cards.** One could imagine only one single device able to perform multiple tasks in many different fields such as electronic payment, identification or health care reimbursement. The smart-card industry is investigating this subject, and commercial products using this technology are expected in the future. The end-user could choose what the functionalities of the card are. The security of such a device will definitely be a central question [50, 52].

- **Ubiquitous Computing.** Ubiquitous computing, i.e. the use of small devices and collaborating devices, is announced as the next technological, industrial and economic boom. It is expected that, in the near future, the objects of our everyday life will become intelligent: they will embed a microprocessor, memory chips and a network interface in order to work collaboratively. Network technologies like Bluetooth or 802.11 now enable personal or mobile networking, that is, the basic technology for ubiquitous computing is available. But, especially for radio communications, network communication is poorly defended against eavesdropping; that is the reason why cryptography must propose well-suited solutions for authentication and data encryption [54].

2.3 Cryptography

Digital security needs a tool to be practically implemented; this tool is called cryptography [55–57]. The field of cryptography provides a range of tools to ensure that information is kept secret from all unauthorised people (confidentiality), to certify data transfers from sender to receiver without being altered by unauthorised or unknown sources (data authentication), to prove the identity of an entity or the source of information (entity authentication), to obviate the denial of actions in the past (non-repudiation), etc. In compliance with Kerckhos' principle, we assume that the complete security of a cryptographic system relies completely on the secrecy of the key. When the key is the same for transforming forwards and backwards, the cryptosystem is called symmetric, whereas asymmetric cryptosystems have two different keys, one for the forward transformation, and one for the backward transformation.

2.3.1 Symmetric

The same key is used to encipher and decipher the plain text with symmetric cryptosystems. Therefore, the sender and the receiver have to share a common key, which must be exchanged through a secure channel. Although symmetric cryptosystems usually have the advantage of being fast, key exchange is problematic: if an attacker is able to eavesdrop the secure channel and retrieve the key, the communication is not secure any more. Besides, they are not well-suited for large-scale communication networks since they require one key for each pair of sender and receiver. For n users, n^2 keys are

necessary, therefore, the problems of key exchange and storage grow with the square of the number of users.

2.3.2 Asymmetric

Asymmetric cryptosystems are also known as public-key cryptosystems: the encryption stage and the decryption stage utilise two different keys. As a consequence, and unlike symmetric cryptosytems, asymmetric cryptosystems are suitable for wide-scale secure communications, since the encryption key can be easily distributed without compromising the decryption key. The security of current technologies often relies on a strict secrecy on the design of the hardware. First, this type of design is vulnerable to leakage from insiders. Second, since only a few experts works on the conception of such cryptosystems, they are more sensitive to security flaws. Third, most of the hardware dedicated to security utilises one key for one pair of communicating entities. This technique raises the complicated problem of key distribution, as well as key storage. On the contrary, asymmetric cryptosystems, also known as public-key cryptosystems (PKC) are based on open specifications, and the most famous public-key cryptosystems have been studied extensively by many cryptographers. We can expect a commercial product relying on this community of researchers to be free from security flaws. Besides, since the idea of public-key cryptosystems is to have two different keys for encryption and decryption, it is possible to distribute the encryption key without compromising the decryption key: in other words, key distribution is easy. The only problem with public-key cryptosystems is that they are inefficient. Since long computational delays are unacceptable for commercial products, efficiency is an unconditional design criterion, even for implementation on low-end processors. One of the major challenges for implementing public-key cryptosystems on cheap embedded devices is to achieve high efficiency without compromising security.

In this project, Elliptic-Curve Cryptography (ECC) is used as a popular public-key cryptosytem. ECC provides a security level equivalent to conventional cryptography, but with a shorter key length. Having a shorter key length means smaller bandwidth and memory requirements, which is a determining factor in most of applications. Another advantage is that there are many curves available to be chosen. Consequently, the curve can be changed periodically for extra security.

2.4 Security in Electronic Embedded Devices

Side-channel analysis forms a part of the group of implementation attacks. The latter set themselves apart from traditional cryptanalysis by putting emphasis on the weaknesses of the implementation rather than on mathematical imperfections in the design of the cipher. Figure 2.1 shows the general scheme of an attacker's scenario. As can be seen, the complete security of a cryptographic system only relies on the secrecy of the key implemented inside the system. Regarding some inherent leakages of data in embedded systems, attackers can take advantage of this vulnerability by analysing the side-channel data leakage (in this figure shown as electromagnet emission (EM))and

guessing the value of the secret key, compromising security.

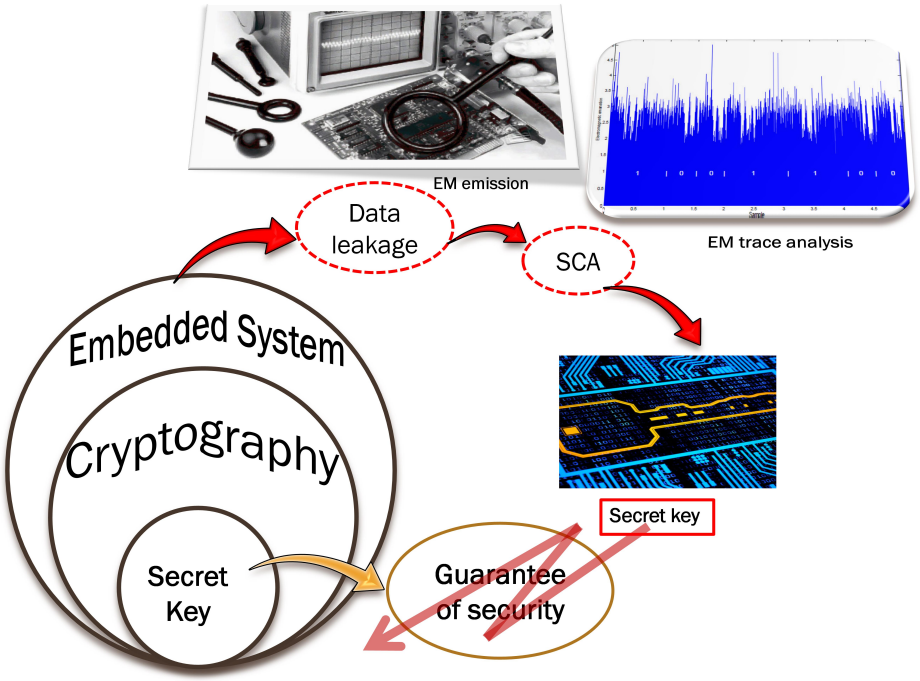


Figure 2.1: General scheme of attacker's scenario. Electro-magnetic (EM) emission is considered as side-channel data leakage

The overall scheme of the proposed side-channel data analysis based on PCA and multi-class classification is depicted in Figure 2.2. As can be seen, N different sample traces of side channel information, such as power consumption or electromagnet emission, with a big dimension of M_1, make a huge input matrix with the dimension of $N \times M_1$. After applying PCA the dimension will reduce to $N \times M_2$, while $M_1 << M_2$; therefore, the input data set would be ready for classification stage. Afterwards, the dataset need to be divided into two segments, training and testing segments, to be prepared for multi-class classification. For this purpose, a cross-validation algorithm is utilized to cross over the training and testing sets in successive rounds such that all data has a chance of being trained and tested. Finally, classification stage can be performed and the accuracy or error of the classifiers can be calculated.

2.4.1 Vulnerabilities of Electrical Embedded-System

As already mentioned, in 1998 Kocher et al. [21] suggested taking advantage of the power consumed by a microchip in order to get information about what the device actually processes. They used a somewhat specific power-consumption model based on the Hamming weight, which is the number of positions at which the corresponding bits

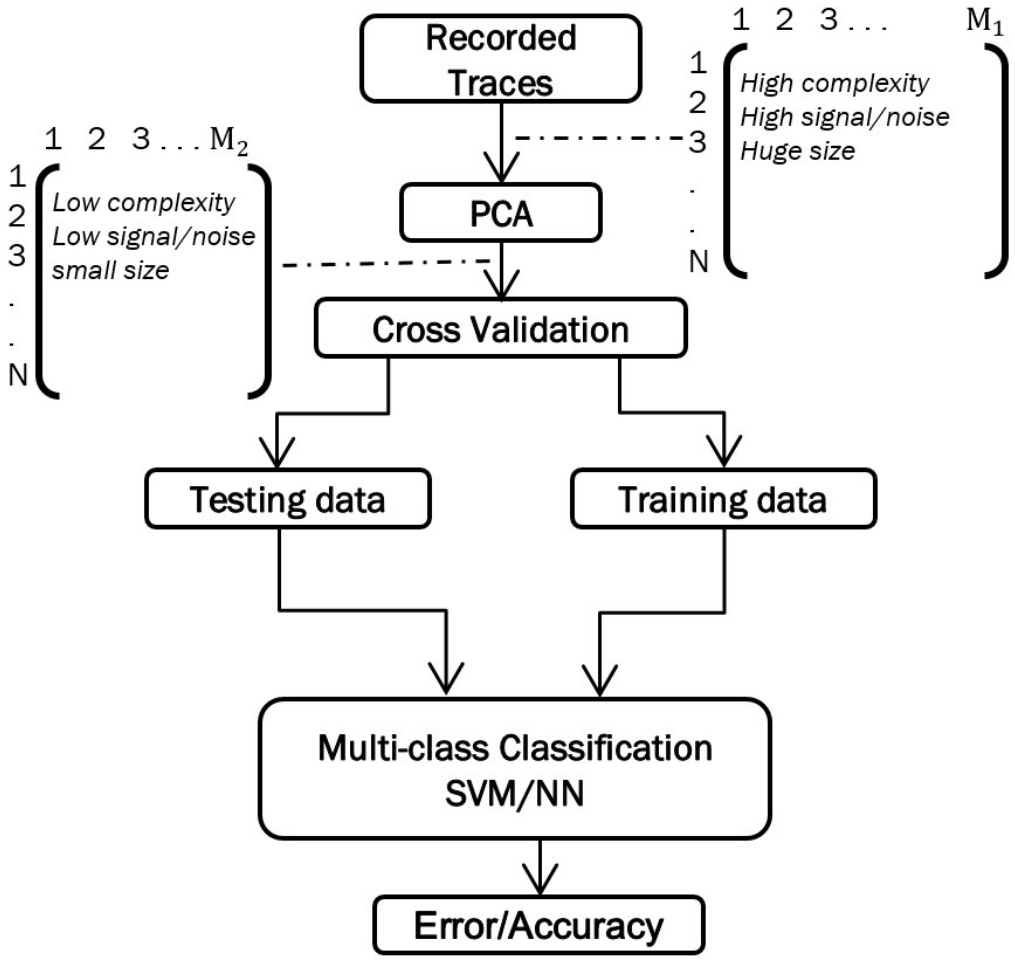


FIGURE 2.2: General scheme of machine learning-based SCA.

are different, of the data handled in the chip. This model was used in many publications [58–61]. A few years later, the model was extended in order to better integrate the behaviour of CMOS circuits. Their power consumption generally relates to the number of bit transitions in a target device. The resulting Hamming distance power-consumption model was applied to ASIC and FPGA implementations of cryptographic algorithms and demonstrated that any kind of implementation could potentially be the target of a side-channel attack [62–64]. In parallel, [65, 66] suggested using the electromagnetic emissions of microelectronic circuits as an alternative, and potentially more powerful, source of side-channel leakage. The approach was shown to provide significant advantages, both from the theoretical and practical points of view.

CMOS Power Consumption

The CMOS technology is certainly the most widely used in current digital design applications. Static CMOS gates have three distinct dissipation sources [67]. The first is due to the leakage currents in transistors. Its contribution to the overall dissipation is in general very small. However, with the important scale-down of silicon technology

nowadays, this source tends to become the highest one and can potentially lead to a new CMOS power model. The second is due to the so-called "direct path current": there exists a short period during the switching of a gate while the NMOS (pull-down transistor) and the PMOS (pull-up transistor) are conducting simultaneously. It is usually estimated that this source accounts for about 20% (Consider that this number can change from one technology to another and it is different from one circuit to another circuit) of the total power consumption. Finally, the most important dissipation, and the most relevant from a side-channel point of view, is due to the charge and discharge of the load capacitance C_L represented by the dotted paths in Figure 2.3. This capacitance is composed of the different parasitic capacitances (junctions, gates, ...) and the wiring capacitance (interconnections). In CMOS devices, when measuring the power consumption (either at the ground pin or at the power pin), the highest peak current will therefore appear during the charge of this capacitance. During the discharge, the only current we can measure is the direct-path current. We simulated and measured a simple CMOS gate to support this assumption.

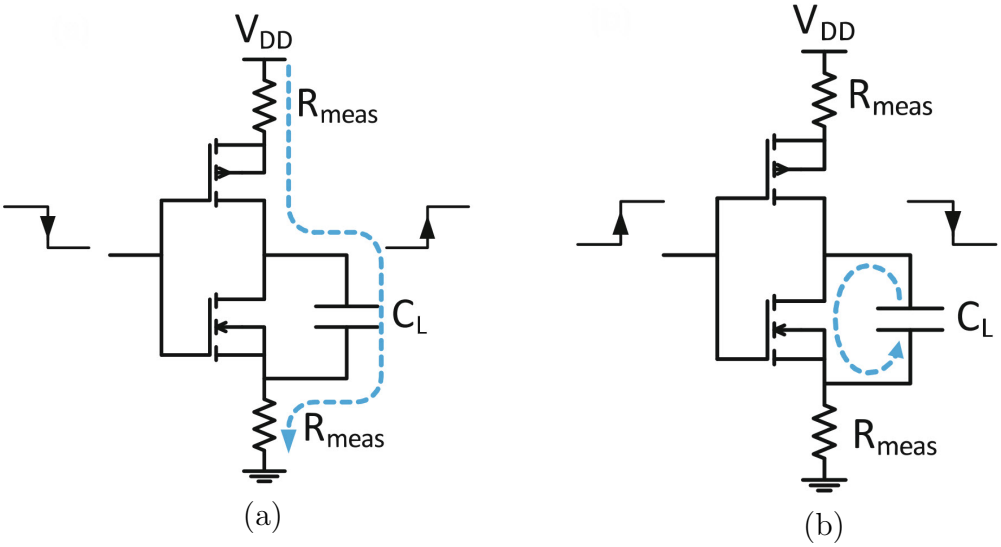


Figure 2.3: Charge versus discharge of the CMOS inverter. (a) Charge of the CMOS inverter's output. (b) Discharge of the CMOS inverter's output.

2.5 Side-Channel-Attack Categories

There are numerous approaches of side-channel attacks, and they can be classified in many ways. The literature usually sorts them in terms of hardware and software level.

2.5.1 Hardware-Level Taxonomy

Considering various physical methods of attack, side-channel attacks are classified as follows:

- **Invasive vs non-invasive:**

  Invasive attacks involve de-packaging to get direct access to the internal components of cryptographic modules or devices. Since invasive attacks typically require relatively expensive infrastructure, they are much harder to deploy. For example:

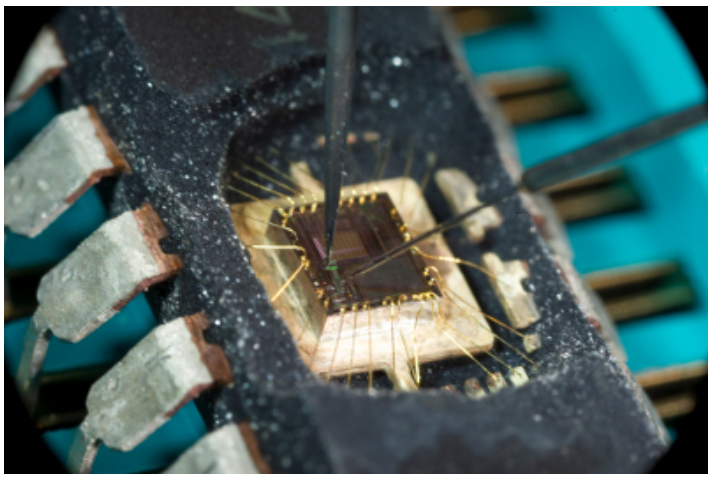
  - Micro-Probing. This technique uses a micro-probing workstation to remove part of the passivation layer (protecting the silicon) of an integrated circuit. Subsequently, an attacker can establish a direct contact with the system (usually the data bus). An attacker can then eavesdrop the data during the execution of cryptographic algorithms [68]. These attacks are obviously invasive and passive attacks.
  - Reverse Engineering. Several attack techniques target particular parts of the smart card namely the buses, memories, CPU, coprocessor, and sensors. Deploying such attacks (fault attacks, microprobing,...) requires access to the layout of the chip, in order to locate and distinguish the internals of the chip. One can make use of image processing and form recognition systems to retrieve the hardware structure from simple microscope pictures (e.g. optical microscope with a CCD camera). Recent techniques [69] illuminate the unplugged chip thanks to a focused laser spot and probe the variation of current between power and ground. Shining light on a transistor makes it generate a micro-current depending on its state. This technique can thus reveal the mapping of the integrated circuit as well as the data stored.

  Semi-invasive attacks involve access to the device, but without damaging the passivation layer or making electrical contact other than with the authorised surface. For example, fault Attacks in which fault induction techniques intend to manipulate the environmental conditions of the system (voltage, clock, temperature, radiation, light, eddy current, etc.) to generate faults and observe the related behaviour, for example simply illuminating a transistor with a laser beam, which causes it to conduct (the photovoltaic effect) [70, 71]. Most of these attacks target data being computed or manipulated by a cryptographic algorithm. Nevertheless, some of them attempt to corrupt the data directly in the memory. While there are many ways of producing a fault in a mobile device, these attacks can be termed semi-invasive, as knowledge of the architecture is often required.

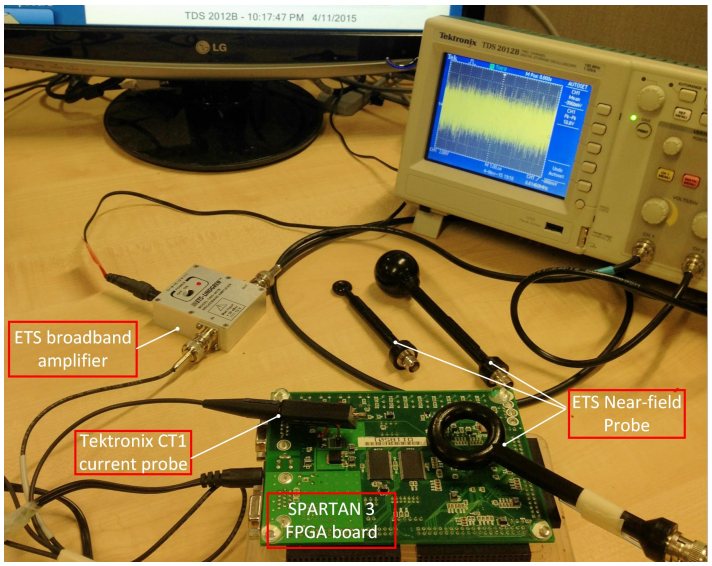
  Non-invasive attacks only exploit externally available information without opening the device. They tend to be cheap and scalable (compared to invasive attacks). There are many forms of non-invasive attacks.

  - Timing Attack: A timing attack exploits the observation that the computations performed in some cryptographic algorithms often take different amounts of time on different inputs. For example, a last reduction step is required in most modular multiplication techniques. Depending on the result at the end of the multiplication, this last reduction may (or may not) be necessary.

– Power Attack: This type of attack records and analyses the power traces leaked by a device.
– Electromagnetic Attack: This type of attack exploits the electromagnetic emanations due to the current flowing through the device.



(a) Invasive Attack.



(b) Non-invasive Attack.

Figure 2.4: Invasive vs Non-invasive attack

- **Active vs Passive:** Active attacks try to tamper with the device's proper functioning and may modify the message; for example, fault-induction attacks will try to induce errors in the computation. On the other hand, passive attacks will simply observe the device's behaviour during the processing, without disturbing it. A passive attack monitors unencrypted traffic but does not modify the message, and looks for clear-text passwords and sensitive information that can be used in other types of attack.

Comparison of Physical Side-Channel Attacks

Table 2.1 present a comparison of the advantages and disadvantages of different types of physical side-channel attacks. The side-channel attacks we consider in this thesis are

TABLE 2.1: Comparison of Physical Side-Channel Attacks

	Advantages	**Disadvantages**
Invasive	- Very strong to extract data	- Expensive - knowledgeable attackers - Time consuming - With physical evidence
Non-Invasive	- No physical evidence - Moderate cost - Proper for real-time attacks	- Not very strong to extract data
Semi-Invasive	Compared to invasive attacks - Inexpensive - Easy to set up and repeat	- Possibility of physical evidences
Passive	- Only observe the target - No evidence - Inexpensive	- Not capable of changing program flow
Active	- Changing program flow	- Expensive - With evidence

a class of physical attacks in which an adversary tries to exploit physical information leakages such as timing information [17], power consumption [21] or electromagnetic radiation [66]. Since they are non-invasive, passive, and can generally be performed using relatively cheap equipment, they pose a serious threat to the security of most cryptographic hardware devices.

2.5.2 Software-Level Taxonomy

Considering various methods of side-channel-information analysis, attacks can be classified as follows:

- **Unsupervised Attacks**
  This type of attack aims to find the patterns in non-labelled side-channel information.
  - **Simple Analysis of Side-Channel Information.**
    Simple Power Analysis (SPA) was introduced as a technique that directly interprets power consumption measurements that are collected during the cryptographic operation. SPA allows to make inner parts of computations of the algorithm to be visible, just by observing the instantaneous measurement outcomes. The alternative use of electromagnetic (EM) side channels was proposed and experiments conducted on EM side channels. Accordingly, the attack was named Simple Electromagnetic Analysis (SEMA).

– **Differential Analysis of Side-Channel Information.**
Besides Simple Power Analysis (SPA), the fundamental work of [7] is dedicated to Differential Power Analysis (DPA), and for the EM channel, [66] introduced the corresponding term Differential Electromagnetic Analysis (DEMA). Similar to SPA, DPA/DEMA also focuses on side-channel measurements at a particular instant of time. However, in DPA/DEMA multiple side-channel measurements are partitioned into two sets depending on the boolean state of a key-dependent intermediate variable, which is predicted by an adversary using known plain-text/cipher-text and a key hypothesis. The biggest advantage of differential analysis over simple analysis is the fact that neither side-channel leakage models nor timing characteristics of the target device must be known.

– **Comparative Side-Channel Attacks.**
Comparative SCA resides between a simple and a differential SCA. Two portions of the same or different leakage traces are compared to discover the reuse of values. The umbrella term was introduced in [72], but the first reported attack belonging to this category is a doubling attack. The doubling attack [73] on ECC is an attack with chosen inputs and has been shown to be powerful to attack some classic SPA-protected algorithms such as left-to-right (downward) double-and-add-always algorithms. The attacker does not need to know whether a computation being performed is a point doubling or an addition. To thwart this attack, blinding techniques can be effective. Care has to be taken however that neither blinding the base point or the scalar is applied solely. This has been proven to be insecure [73]. Combined use strengthens the security.

– **Refined Power Analysis.**
A refined power analysis (RPA) in side-channel attacks on elliptic-curve cryptography, directs its attention to the existence of a point P_0 on the elliptic curve such that one of the coordinates is 0 and $P_0 \neq 0$. Randomised projective coordinates, randomised EC isomorphisms and randomized field isomorphisms preserve this specific property of the point P_0. Feeding to a device a point P that leads to a special point $r(0, y)$ under the assumption of some specific key bits will generate exploitable side-channel leakage [74, 75].

The attack can be thwarted by using either a cofactor variant of a protocol for points of small order or by using isogenous curves for points of "large order". The zero-value point attack (ZPA) generalises this attack [24]: zero value points in intermediate results are also considered.

– **Carry-based Attack.**
The carry-based attack [76], reported by Fouque et al., does not attack the scalar multiplication itself but its countermeasures. It relies on the carry propagation occurring when long-integer additions are performed as repeated sub-word additions.

– **Machine-learning-Unsupervised Analysis**

Machine learning is defined as a study of how a machine improves its performance based on previous experience or training. Most machine learning problems deal with classification of the data, or finding the structure of the data. As an unsupervised learning [77], the machine is given a set of unlabelled data, and it tries to determine the hidden structure of the data, for example the clustering of the data based on a similarity metric.

- **Supervised Attacks**
  The purpose of supervised attacks is to construct models (classifiers) that are able to make predictions based on labelled side-channel information that were previously collected.
  - **Template Attacks**
    Template attacks were introduced in [27] to extract the maximal possible information from observed side channels like the power consumption of a cryptographic operation performed on a smart card. The underlying attack model is quite powerful: before attacking a device with an unknown key the attacker has full access to an identical device and is able to record side-channel data for chosen keys and plain texts. After this profiling phase the attacker should be able to extract the key of the attacked device from a single measurement with given or chosen plain text. The basic idea of the template attack is to model the power consumption as a high-dimensional Gaussian distribution dependent on a few key bits. Its key-dependent mean and covariance matrix are then estimated using the data recorded during the profiling phase. Finally the maximum likelihood method determines the correct key bits [28, 29].
  - **Machine-learning-Supervised Analysis**
    As a supervised attack, the machine-learning algorithm is given a set of training data with the label or class available, and it tries to determine the function that associates the data with the label. Its performance is then tested on an independent set of data, by evaluating the prediction (label) it outputs.

Verification of Side-Channel Analysis approaches

Table 2.2 verifies the strength and weakness of different approaches of side-channel attack on an ECC crytosystem. For each approach, the effective countermeasure as well as the vulnerable countermeasures are classified.

In this thesis, a non-invasive attack is performed by measuring the power consumption and electromagnetic radiation of an FPGA-based ECC cryptosystem. Also, the performance of machine-learning techniques as powerful multi-class classifiers in the form of a neural network and a support-vector machine is verified.

Table 2.2: Verification of Side-Channel Attacks on ECC and Countermeasures. (CB= carry-based attack, ML= machine-learning analysis and TA= template attack)

	Effective Countermeasures	Broken Countermeasures
SPA/SEMA	- Montgomery Powering Ladder [18] - Double-and-add-always [18] - Indistinguishable Point Addition [39]	(None yet)
DPA/DEMA	- Random scalar split [78] - Random Projective Coordinates [18] - Randomised EC Isomorphisms [78] - Randomised Field Isomorphisms [78]	- Scalar randomisation [18] - Base point blinding [18]
Comparative	- Randomised EC/field Isomorphisms [78] (might work, not being addressed yet)	- Double-and-add-always[18] - Montgomery Powering Ladder [79] - Scalar randomisation [18] - Base point blinding [18]
Refined Power	- Random scalar split [78] - Scalar randomisation [18] - Base point blinding [18]	- Montgomery Powering Ladder [79] - Random Projective Coordinates [18] - Randomised EC Isomorphisms [78] - Randomised Field Isomorphisms [78]
CB	(None yet)	- Random scalar split [78] - Scalar randomisation [18]
ML	(Random-based countermeasures might work, not being addressed yet)	- Double-and-add-always [18] - Montgomery Powering Ladder [79] (Nothing being addressed yet)
TA	- Random Projective Coordinates [18]	- Scalar randomisation [18] - Base point blinding [18]

3 Elliptic-Curve Cryptosystem Implementation

3.1 Introduction

In this chapter a strong public-key cryptosystem based on elliptic-curve cryptography (ECC) is introduced. Then its FPGA-based implementation and considerations are discussed.

Elliptic-Curve Cryptography (ECC) was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing public-key cryptography. This technique provides an equivalent security level to conventional cryptography, but with a shorter key length. According to some researchers, ECC can yield a level of security with a 164-bit key while other systems requires a 1,024-bit key to achieve. Having a shorter key length means smaller bandwidth and memory requirements. In some applications, these might be critical factors. Another advantage is that there are many curves available to be chosen. Consequently, the curve can be changed periodically for extra security.

Early public-key systems are secure assuming that it is difficult to factor a large integer composed of two or more large prime factors. For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic-curve element with respect to a publicly known base point is infeasible. The security of ECC depends on the ability to compute a point multiplication and the inability to compute the multiplicand given the original and product points. The size of the elliptic curve determines the difficulty of the problem.

Publications pertaining to this chapter:

- M. S. Hossain, Y. Kong, E. Saeedi, and N. C. Vayalil, "High-Performance Elliptic Curve Cryptography Processor over NIST Prime Field", *IET Computers and*

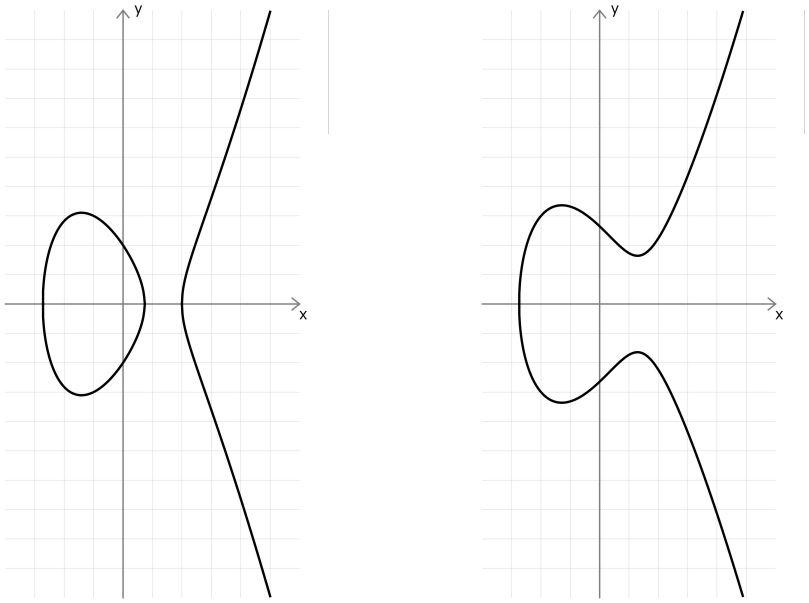
Digital Techniques.

- M. S. Hossain, Y. Kong and E. Saeedi, "High-Performance FPGA Implementation of Elliptic Curve Cryptography Processor over Binary Field GF (2^{163})", .

- M. S. Hossain, Y. Kong and E. Saeedi, "High-Speed, Area-Efficient, FPGA-Based Elliptic Curve Cryptographic Processor over NIST Binary Fields", *2015 IEEE International Conference on Data Science and Data Intensive Systems (DSDIS), UTS, Sydney, Australia, pp. 175-181, December 11-13, 2015.*

3.2 ECC Geometries

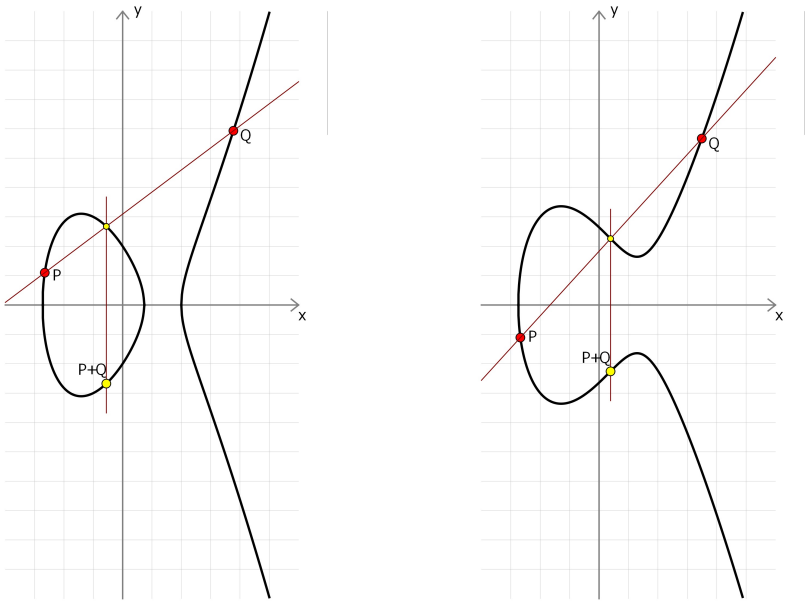
Elliptic curves are geometric objects that can be described by an equation of the form $y^2 = x^3 - ax + b$ where a and b are constants. Figure 3.2 illustrates these equations. The distinguishing property of elliptic curves is that they have an "addition



(a) $y^2 = x^3 - 6x + 4$ (b) $y^2 = x^3 - 5x + 7$

FIGURE 3.1: Elliptic-curve geometry

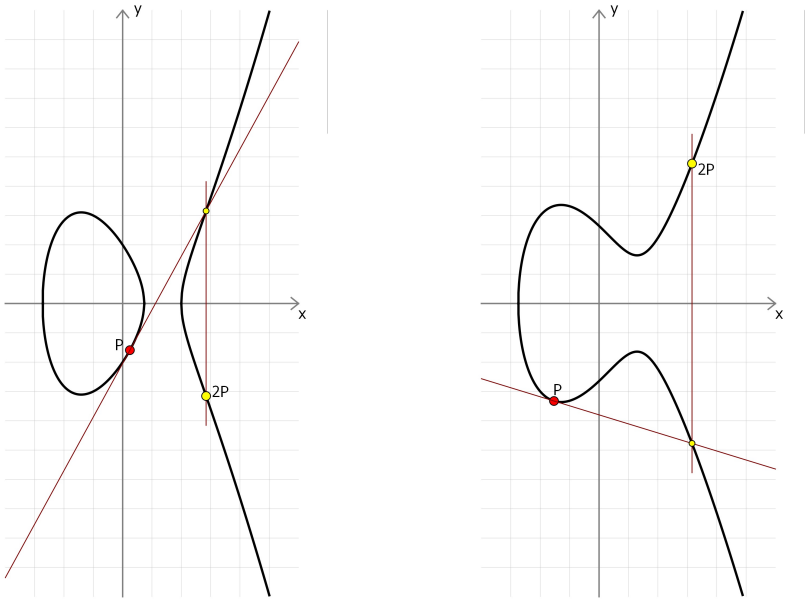
law". This means that, given two points on an elliptic curve, there is a natural and geometric way to "add" those two points to produce a third point. This process is called "addition" because it has many of the same properties as addition of integers, such as $P+Q = Q+P$ (commutativity) and $(P+Q)+R = P+(Q+R)$ (associativity). The idea is that, given two distinct points$P$ and Q on the elliptic curve, connect P and Q by a straight line, then the intersection of the curve and the straight line would be



(a) $y^2 = x^3 - 6x + 4$ (b) $y^2 = x^3 - 5x + 7$

Figure 3.2: Elliptic-curve addition geometry, $P + Q$.

$-(P + Q)$. Another important operation in elliptic curves is "doubling". In order to find $2P$, a straight line should be drawn which is tangent to the curve at P. Then the intersection of the curve and the line would be $-2P$; see Figure 3.2.



(a) $y^2 = x^3 - 6x + 4$ (b) $y^2 = x^3 - 5x + 7$

Figure 3.3: Elliptic-curve doubling geometry, $2P$.

3.3 ECC Preliminaries

ECC can be implemented in either prime fields $\mathbb{F}_p$ with p a 'large' prime or binary fields $\mathbb{F}_{2^m}$ with m a positive integer. But finite-field modular arithmetic (FFMA) over $\mathbb{F}_p$ will be the emphasis of this work, and has the capability of supporting RSA as well as ECC. Besides, the most widely used public key cryptography (PKC) is RSA, which uses modular arithmetic over a prime field $\mathbb{F}_p$. According to IEEE P1363 [80], an elliptic curve E over GF(p) is the set of solutions for an equation such as

$$y^2 = x^3 + ax + b \pmod{p} \tag{3.1}$$

where $x, y, a, b \in GF(p)$ with

$$4a^3 + 27b^2 \neq 0 \pmod{p},$$

together with a special point called the point at infinity. The coefficients $a, b \in \mathbb{F}_p$ specifying an elliptic curve $E(\mathbb{F}_p)$ are defined by (3.1). The number of points on an elliptic curve E is represented by $\#E(\mathbb{F}_p)$. It is defined over $\mathbb{F}_p$ as nh, where n is the prime order of the curve and integer h is a co-factor such as $h = \#E(\mathbb{F}_p)/n$ [81, 82]. All the parameters for the NIST elliptic curve over the prime field $\mathbb{F}_{256}$ are listed in Table 3.1 [9, 10].

TABLE 3.1: NIST-recommended elliptic curves over $\mathbb{F}_{256}$ on Koblitz Curve [9, 10]

P-256: p = $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$, $a = 0$, $b = 7$
p=0x FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFE FFFFFC2F
n=0x FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6
AF48A03B BFD25E8C D0364141
x=0x 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB
2DCE28D9 59F2815B 16F81798
y=0x 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448
A6855419 9C47D08F FB10D4B8

An EC defined over a GF provides a group structure that is used to implement cryptographic systems. The group operations are elliptic-curve point addition (ECPA) and elliptic-curve point doubling (ECPD).

3.4 ECC Implementation

There are various coordinate systems to represent elliptic-curve points but two well-known coordinate systems are often used for ECC: affine coordinate systems and projective coordinate systems. A point on the EC E(GF(p)) for affine coordinates can be represented by using two elements $x, y \in \mathbb{F}_p$, i.e. $P(x, y)$. In this coordinate system, the elliptic curve group operations such as ECPD and ECPA require a modular inversion, the most expensive operation. The modular inversion over a prime field for each group operation can be reduced by using projective coordinate systems by adding a few field operations. In projective coordinates, a point P on the EC needs three elements $X, Y, Z \in \mathbb{F}_p$, i.e. $P(X, Y, Z)$. In this project, we use projective coordinate systems of the EC points to avoiding modular inversion. However, EC group operations need more modular multiplication in projective coordinates, which is also a very costly operation for ECC. But a high-performance modular multiplication for ECC is proposed that is well suited for faster ECC operation. In practice, to convert projective to affine coordinates, one modular inversion is still needed for an elliptic-curve point multiplication (ECPM) [83]. There are plenty of projective coordinates in the available literature such as Jacobian, Lopez-Dahab, and Chudnovsky coordinates; a detailed coordinate system is discussed in [10]. Let $P = (x, y)$ be a point in an affine coordinate system, the projective coordinate system $P = (X, Y, Z)$ are given by:

$$X = x;\ Y = y;\ Z = 1. \tag{3.2}$$

The projective point $P = (X, Y, Z)$, $Z \neq 0$ corresponding to the affine point ($P = (x, y)$) is given by

$$x = X/Z^2;\ y = Y/Z^3. \tag{3.3}$$

Using (3.1), (3.2), and (3.3), the projective form of the Weierstrass equation of the elliptic curve becomes

$$Y^2 = X^3 + aXZ^4 + bZ^6. \tag{3.4}$$

Let $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ be two points on the elliptic curve, then the ECPD and ECPA formulae in Jacobian coordinates are given below.

$$\begin{aligned} R(X_3, Y_3, Z_3) &= 2P(X_1, Y_1, Z_1) \in E(\mathbb{F}_p), \\ X_3 &= (3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2, \\ Y_3 &= (3X_1^2 + aZ_1^4)(4X_1Y_1^2 - X3) - 8Y_1^4, \\ Z_3 &= 2Y_1Z_1; \end{aligned} \tag{3.5}$$

$$
\begin{aligned}
R(X_3, Y_3, Z_3) &= P(X_1, Y_1, Z_1) + Q(X_2, Y_2, Z_2) \in E(\mathbb{F}_p), \\
X_3 &= A^2 - B^3 - 2X_1 Z_2^2 B^2, \\
Y_3 &= A(X_1 Z_2^2 B^2 - X_3) - Y_1 Z_2^3 B^3, \\
Z_3 &= Z_1 Z_2 B, \\
\text{where A} &= Y_2 Z_1^3 - Y_1 Z_2^3 \text{ and B} = \mathrm{X_2 Z_1^2 - X_1 Z_2^2}.
\end{aligned} \tag{3.6}
$$

Hence when $P = Q$ we have the ECPD operation in (3.5) and when $P \neq Q$ we have the ECPA operation in (3.6) [84]. Using these operations, the ECPM $R = kP$, which is the most important operation in ECC, will be implemented in Jacobian coordinates [10, 81, 85].

In 2000, FIPS-2 was recommended with 10 finite fields: 5 prime fields ($\mathbb{F}_p$) and

TABLE 3.2: Comparison of key length for equivalent security of symmetric-key and public-key in prime field Cryptography [10, 11]

Symmetric key	Example algorithm	RSA/DH	ECC in GF(p)
80	SKIPJACK	1024	192
112	Triple-DES	2048	224
128	AES Small	3072	256
192	AES Medium	8192	384
256	AES Large	15360	521

5 binary fields ($\mathbb{F}_{2^m}$). The prime fields are $\mathbb{F}_{192}, \mathbb{F}_{224}, \mathbb{F}_{256}, \mathbb{F}_{384}$ and $\mathbb{F}_{521}$[10]. The comparison between symmetric cipher key length and key lengths for PKC such as RSA, Diffie-Hellman (DH), and ECC over a prime field GF(p) are given in Table 3.3. It demonstrates that smaller field sizes may be used in ECC than in RSA and DH systems for the same security level. For instance, 256-bit ECC gives equivalent security to 3072-bit RSA with a significantly smaller key size. Besides, in practice a 256-bit ECC system over a prime field is very useful for modern security applications. Though RSA is a popular PKC, ECC is many times more efficient than RSA and DH for either public-key operations (such as signature generation and decryption) or private-key operations (such as signature verification and encryption). This makes ECC a promising branch of PKC [10, 11, 83, 86].

3.4.1 Elliptic-Curve Cryptography based on Prime field and in Affine Coordinates

Elliptic-Curve Cryptography (ECC) is a powerful Public-Key Cryptography (PKC) algorithm, and nowadays it is very popular due to the smaller field size. ECC can be implemented in either prime fields GF(p) with p a 'large' prime or binary fields GF(2^m) with m a positive integer. Both fields are considered to provide almost the same level of security [87]. But EC over prime fields will be the emphasis of this work due to the use of efficient finite-field modular arithmetic (FFMA). EC defined over a GF provides a group structure that is used to implement the cryptographic systems. The group operations are EC point addition (ECPADD) and EC point doubling (ECPDBL).
In this work, all elliptic-curve operations in an affine coordinate system have been implemented. An elliptic curve E over GF(p) in affine coordinates is the set of solutions for an equation such as

$$y^2 = x^3 + ax + b \tag{3.7}$$

where $x, y, a, b \in GF(p)$ with

$$4a^3 + 27b^2 \neq 0.$$

The coefficients $a, b \in \mathbb{F}_p$ specifying an elliptic curve $E(\mathbb{F}_p)$ are defined by (3.7). The number of points on elliptic curve E is represented by $\#E(\mathbb{F}_p)$. It is defined over $\mathbb{F}_p$ as nh, where n is the prime order of the curve, and the integer h is a co-factor such as $h = \#E(\mathbb{F}_p)/n$ [10, 81, 85].
Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on the EC; then point addition (PADD) and point doubling (PDBL) formulae in affine coordinates are given below.

$$\begin{aligned} R(x_3, y_3) &= P(x_1, y_1) + Q(x_2, y_2) \in E, \\ x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \\ \text{where } \lambda &= (y_2 - y_1)/(x_2 - x_1) \text{ and } P \neq Q; \end{aligned} \tag{3.8}$$

$$\begin{aligned} R(x_3, y_3) &= 2P(x_1, y_1) \in E, \\ x_3 &= \lambda^2 - 2x_1, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \\ \text{where } \lambda &= (3x_1^2 + a)/2y_1 \text{ and } P = Q; \end{aligned} \tag{3.9}$$

where $R = 0$ when $x_1 = x_2$ and $y_2 \neq y_1$, or $x_1 = x_2 = 0$. Hence, when $P \neq Q$ we have the PADD operation in (3.8) and when $P = Q$ we have the PDBL operation in (3.9). Using these operations, the ECSM $R = kP$ will be implemented in affine coordinates using an ECC-based algorithm [10, 81, 85].

In 2000, FIPS-2 was recommended with 10 finite fields: 5 prime fields, and 5 binary fields. The prime fields are $\mathbb{F}_{192}, \mathbb{F}_{224}, \mathbb{F}_{256}, \mathbb{F}_{384}$ and $\mathbb{F}_{521}$[10]. The comparison between

symmetric cipher key length and key lengths for PKC like RSA, Diffie-Hellman (DH), and ECC over prime field GF(p) are given in Table 3.3. It demonstrates that smaller field sizes can be used in ECC than in RSA and DH systems at a given security level. For instance, 224-bit ECC gives equivalent security to 2048-bit RSA, and 256-bit ECC gives equivalent security to 3072-bit RSA with significantly smaller keys and smaller area. ECC is many times more efficient than RSA and DH for either public-key operations (such as signature generation and decryption) or private-key operations (such as signature verification and encryption). This makes ECC a promising branch of public-key cryptography [10, 11, 83, 86].

The implementation hierarchy of ECC operations over the prime field is presented in Figure 3.4. From this figure, we can see that ECC protocols and ECSM are the building blocks of elliptic curve group operations and finite-field modular arithmetic. The top level of the cryptosystem contains ECC protocols like EC-DH (EC-Diffie-Hellman) key exchange, EC-DSA (EC-Digital Signature Algorithm). The second level contains ECSM, which is the key operation of the ECC processor (ECP), and it comprises a series of point additions (PADD) and doubling (PDBL). The third level comprises ECPADD and ECPDBL, which are called elliptic curve group operations. These are the series of finite-field modular arithmetic (FFMA) such as modular addition, subtraction, multiplication, squaring, and inversion. The FFMA units are the bottom or fourth level in the hierarchy and these are the most crucial for the overall performance of the ECC processor.

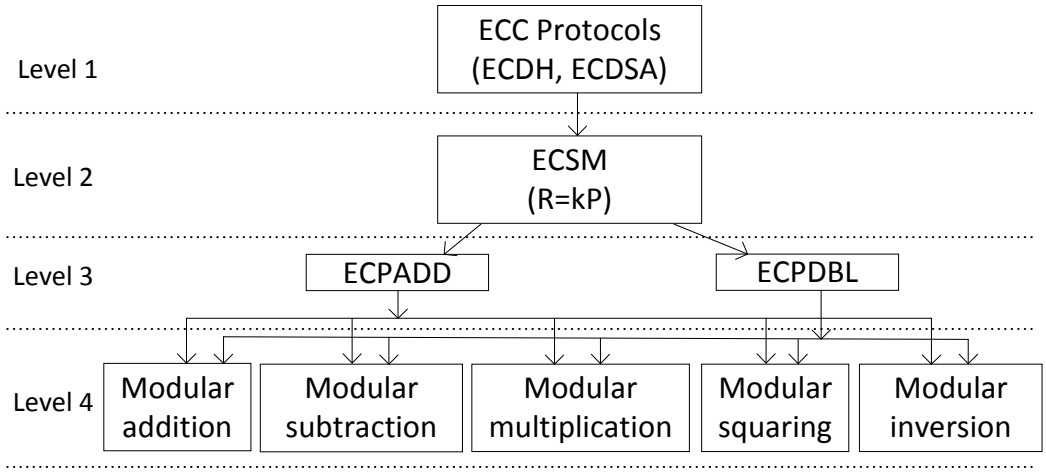


FIGURE 3.4: Implementation hierarchy of the ECC operations over GF(p).

3.4.2 Elliptic-Curve Cryptography based on Binary field and in Affine Coordinates

Elliptic-curve cryptography (ECC) is performed in either prime fields GF(p) or binary fields $GF(2^m)$. But ECs over $GF(2^m)$ will be the emphasis of this work because it

is very efficient for hardware implementation due to the use of modulo-2 arithmetic. An elliptic-curve defined over a finite field provides a group structure that is used to implement the cryptographic systems. The group operations are elliptic-curve point addition (ECPA) and elliptic-curve point doubling (ECPD). There have been different coordinate systems to represent elliptic-curve points. They vary in the number and type of field operations required to implement PA/PD. In our work, we implement all elliptic-curve operations in an affine coordinate system. A non-supersingular elliptic curve E over $GF(2^m)$ in affine coordinates is the set of solutions to the equation

$$y^2 + xy = x^3 + ax^2 + b \tag{3.10}$$

where $x, y, a, b \in GF(2^m), b \neq 0$. The coefficients $a, b \in \mathbb{F}_2{}^m$ specifying an elliptic curve $E(\mathbb{F}_2{}^m)$ are defined by (3.7). The number of points on an elliptic curve E is represented by $\#E(\mathbb{F}_2{}^m)$. It is defined over $\mathbb{F}_2{}^m$ as nh, where n is the prime order of the curve and h is an integer called the co-factor.
If $P = (x_1, y_1) \in E$ and $Q = (x_2, y_2) \in E$ (points on the EC), then summing PA and doubling PD can be respectively derived as

$$\begin{aligned} R(x_3, y_3) &= P(x_1, y_1) + Q(x_2, y_2) \in E, \\ x_3 &= \lambda_1^2 + \lambda_1 + x_1 + x_2 + a, \\ y_3 &= \lambda_1(x_1 + x_3) + x_3 + y_1, \\ \text{where } \lambda_1 &= (y_2 + y_1)/(x_2 + x_1) \text{ and P} \neq \text{Q}; \end{aligned} \tag{3.11}$$

$$\begin{aligned} R(x_3, y_3) &= 2P(x_1, y_1) \in E, \\ x_3 &= \lambda^2 + \lambda + a, \\ y_3 &= x_1^2 + \lambda x_3 + x_3, \\ \text{where } \lambda &= x_1 + y_1/x_1 \text{ and P} = \text{Q}; \end{aligned} \tag{3.12}$$

where $R = 0$ when $x_1 = x_2$ and $y_2 \neq y_1$, or $x_1 = x_2 = 0$. Hence, when $P \neq Q$ we have the PA operation in (2) and when $P = Q$ we have the PD operation in (3). Using these operations, ECSM kP will be implemented using an ECC-based algorithm [10, 81, 85, 88].

In 2000, FIPS-2 was recommended with 10 finite fields: 5 prime fields, and 5 binary fields. The binary fields are $\mathbb{F}_2{}^{163}, \mathbb{F}_2{}^{233}, \mathbb{F}_2{}^{283}, \mathbb{F}_2{}^{409}$ and $\mathbb{F}_2{}^{571}$[9]. Both prime fields GF(p) and GF(2^m) are considered to provide almost the same level of security [87]. Table 3.3 compares symmetric cipher key length, and key lengths for public-key cryptography like RSA, Diffie-Hellman (DH), and ECC in a binary field. It demonstrates that smaller field sizes can be used in ECC than in RSA and DH systems at a given security level. For instance, 283-bit ECC gives equivalent security to 3072-bit RSA with significantly smaller keys and area. This makes ECC a promising branch of public-key cryptography [10, 11].

TABLE 3.3: Comparison of key length for equivalent security of symmetric-key and public-key in binary field Cryptography [10, 11]

Symmetric key	Example algorithm	RSA/DH	ECC in GF(2^m)
80	SKIPJACK	1024	163
112	Triple-DES	2048	233
128	AES Small	3072	283
192	AES Medium	8192	409
256	AES Large	15360	571

3.4.3 Proposed EC scalar multiplication (ECSM)

Though EC scalar multiplication (ECSM) over $\mathbb{F}_p$ is the main operation of an ECC processor, it is computationally the most expensive. However, we have implemented a high-performance ECSM on a FPGA. ECSM is the building block of group operations and finite-field modular arithmetic (FFMA) units. The basic operation of EC scalar multiplication is defined as kP, where k is a positive integer and P is a point on the elliptic curve E defined over a prime field $\mathbb{F}_p$. A point on the elliptic curve E($\mathbb{F}_p$) for affine coordinates can be represented by using two elements $x, y \in \mathbb{F}_p$, i.e. P(x ,y). Thus, the inputs of the ECSM are Px and Py, and k, the scalar multiplier. An ECP architecture over GF(p) is presented in Figure 3.5. There are different methods to implement EC scalar multiplication: the binary method, the Non-adjacent form (NAF) method, and the Montgomery method. The easiest way to implement ECC is the binary method (left to right) [10]. Finally, we present the EC scalar multiplication Algorithm 2 using the binary method. It is implemented using the "Double-and-Add" algorithm concept. Using this algorithm, on average m PDBL and $m/2$ PADD operations are required for an ECSM, where m is the bit length of the multiplicand and $m \simeq \lceil \log_2 p \rceil$.

Algorithm 2: Binary method (Left to right) for point multiplication

Input: $k = (k_{m-1,\ldots,k_1,k_0})_2$, $P(x, y) \in E(\mathbb{F}_2{}^m)$
Output: $R(x, y) = k.P(x, y)$, where $R(x, y), P(x, y) \in E(\mathbb{F}_2{}^m)$

```
R = 0 ;
for i = m - 1 to 0 do
    R = 2R;
    if k(i) = '1' then
        R = R + P ;
    end
end for
Return (R(x, y))
```

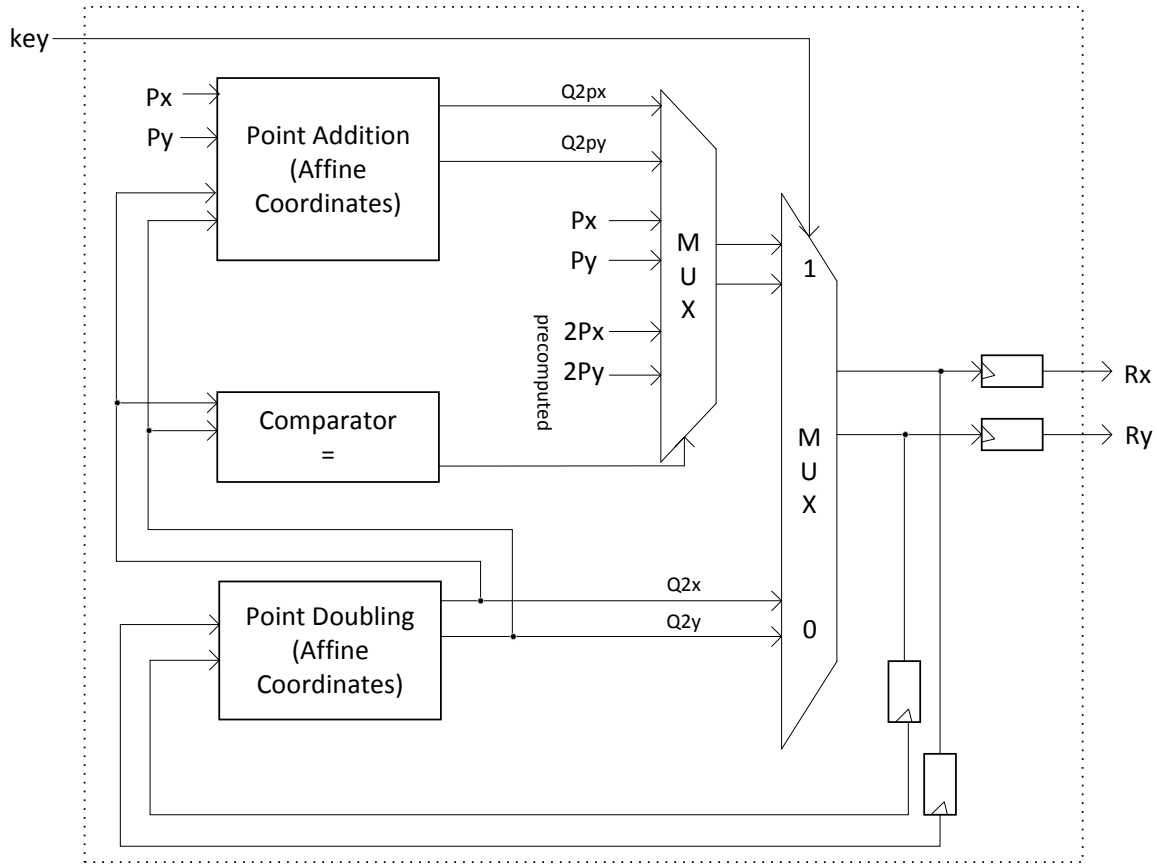


Figure 3.5: Overall hardware architecture of ECSM for prime field (this figure for ECC 2 or ECC 3).

Figure 3.6 shows the Register Transfer Level (RTL) schematic of our FPGA-based ECC.

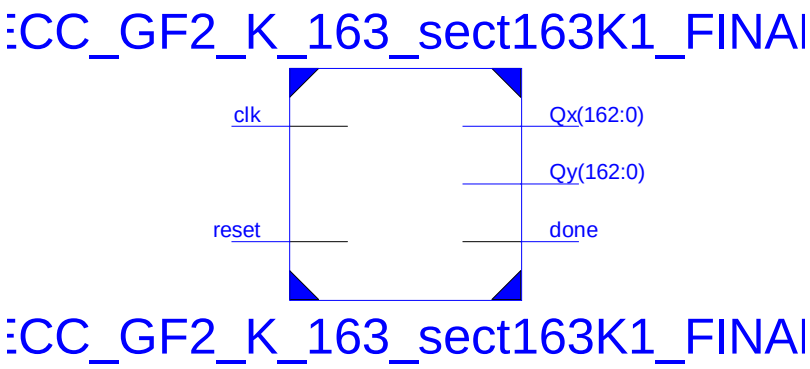
The final design summary is illustrated in Figure 3.7 and 3.8.

Figure 3.9 illustrates our simulation results.

3.5 Experimental Setup for Elliptic-Curve Cryptosystem Implementation

Depending on to the type of side-channel attack, different kinds of measurement tools and analytical software can be used for a successful SCA. Figure 3.10 shows our main measurement setup.

As can be seen, the basic requirements are:



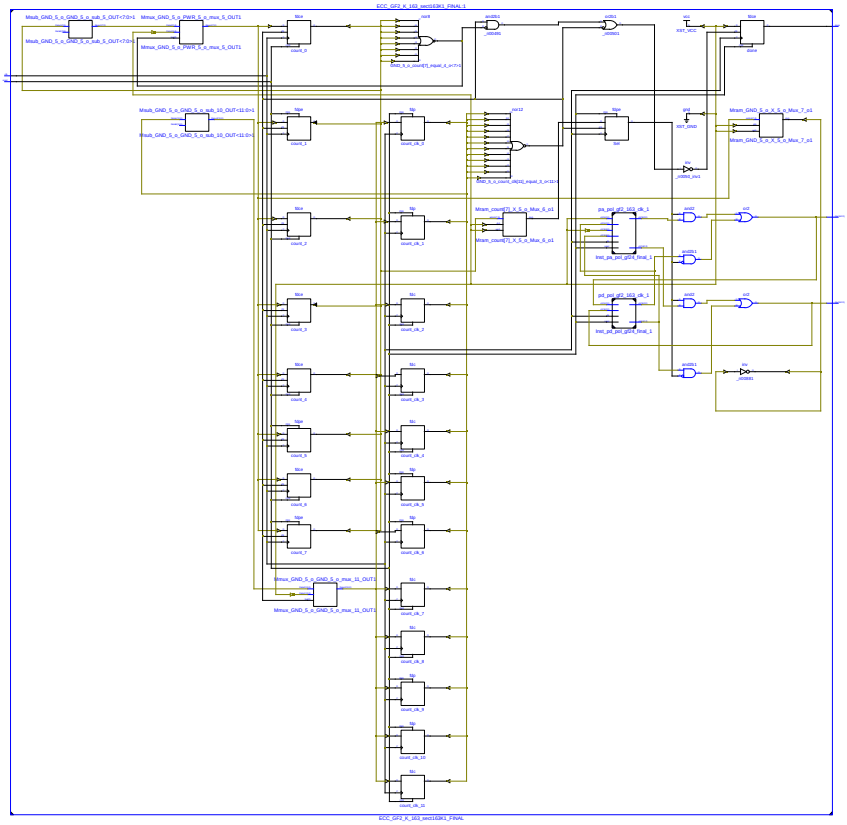


FIGURE 3.6: RTL Schematic of our FPGA-based ECC.

- **FPGA Board:** the ECC cryptosystem is implemented on an FPGA board with a SPARTAN 3 FPGA. The Spartan-3 board provides a powerful, self-contained development platform for designs targeting the Spartan-3 FPGA from Xilinx. Its features are a 200K-gate Spartan-3, on-board I/O devices, and 1 MB fast asyncronous SRAM. The board also contains a Platform Flash JTAG-programmable

ECC_GF2_K_163_sect163K1_FINAL Project Status (03/06/2016 - 14:44:16)			
Project File:	ECC_GF2_K-163_sect163K1_FINAL.xise	**Parser Errors:**	No Errors
Module Name:	ECC_GF2_K_163_sect163K1_FINAL	**Implementation State:**	Programming File Generated
Target Device:	xc7k325t-2ffg900	• Errors:	No Errors
Product Version:	ISE 14.7	• Warnings:	684 Warnings (0 new)
Design Goal:	Balanced	• Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	X 1 Failing Constraint
Environment:	System Settings	• Final Timing Score:	15224 (Timing Report)

Device Utilization Summary [-]				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	6,620	407,600	1%	
Number used as Flip Flops	6,620			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	7,338	203,800	3%	
Number used as logic	7,227	203,800	3%	
Number using O6 output only	6,441			
Number using O5 output only	2			
Number using O5 and O6	784			
Number used as ROM	0			
Number used as Memory	0	64,000	0%	
Number used exclusively as route-thrus	111			
Number with same-slice register load	111			
Number with same-slice carry load	0			
Number with other load	0			
Number of occupied Slices	2,408	50,950	4%	
Number of LUT Flip Flop pairs used	7,885			
Number with an unused Flip Flop	2,011	7,885	25%	
Number with an unused LUT	547	7,885	6%	
Number of fully used LUT-FF pairs	5,327	7,885	67%	
Number of unique control sets	15			
Number of slice register sites lost to control set restrictions	36	407,600	1%	
Number of bonded IOBs	329	500	65%	
Number of RAMB36E1/FIFO36E1s	0	445	0%	
Number of RAMB18E1/FIFO18E1s	0	890	0%	
Number of BUFG/BUFGCTRLs	2	32	6%	
Number used as BUFGs	2			
Number used as BUFGCTRLs	0			
Number of IDELAYE2/IDELAYE2_FINEDELAYs	0	500	0%	
Number of ILOGICE2/ILOGICE3/ISERDESE2s	0	500	0%	
Number of ODELAYE2/ODELAYE2_FINEDELAYs	0	150	0%	
Number of OLOGICE2/OLOGICE3/OSERDESE2s	0	500	0%	
Number of PHASER_IN/PHASER_IN_PHYs	0	40	0%	
Number of PHASER_OUT/PHASER_OUT_PHYs	0	40	0%	
Number of BSCANs	0	4	0%	

1

Figure 3.7: Final design summary, page 1.

Number of BUFHCEs	0	168	0%	
Number of BUFRs	0	40	0%	
Number of CAPTUREs	0	1	0%	
Number of DNA_PORTs	0	1	0%	
Number of DSP48E1s	0	840	0%	
Number of EFUSE_USRs	0	1	0%	
Number of FRAME_ECCs	0	1	0%	
Number of GTXE2_CHANNELs	0	16	0%	
Number of GTXE2_COMMONs	0	4	0%	
Number of IBUFDS_GTE2s	0	8	0%	
Number of ICAPs	0	2	0%	
Number of IDELAYCTRLs	0	10	0%	
Number of IN_FIFOs	0	40	0%	
Number of MMCME2_ADVs	0	10	0%	
Number of OUT_FIFOs	0	40	0%	
Number of PCIE_2_1s	0	1	0%	
Number of PHASER_REFs	0	10	0%	
Number of PHY_CONTROLs	0	10	0%	
Number of PLLE2_ADVs	0	10	0%	
Number of STARTUPs	0	1	0%	
Number of XADCs	0	1	0%	
Average Fanout of Non-Clock Nets	5.16			

Performance Summary			[-]
Final Timing Score:	15224 (Setup: 15224, Hold: 0)	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	X 1 Failing Constraint		

Detailed Reports					[-]
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Sat 5. Mar 22:18:57 2016	0	22 Warnings (0 new)	52 Infos (0 new)
Translation Report	Current	Sun 6. Mar 14:36:00 2016	0	1 Warning (0 new)	0
Map Report	Current	Sun 6. Mar 14:39:45 2016	0	331 Warnings (0 new)	5 Infos (0 new)
Place and Route Report	Current	Sun 6. Mar 14:42:04 2016	0	0	3 Infos (0 new)
Power Report					
Post-PAR Static Timing Report	Current	Sun 6. Mar 14:42:36 2016	0	0	4 Infos (0 new)
Bitgen Report	Current	Sun 6. Mar 14:44:11 2016	0	330 Warnings (0 new)	1 Info (0 new)

Secondary Reports			[-]
Report Name	Status	Generated	
ISIM Simulator Log	Current	Sun 6. Mar 14:19:55 2016	
WebTalk Report	Current	Sun 6. Mar 14:44:12 2016	
WebTalk Log File	Current	Sun 6. Mar 14:44:15 2016	

Date Generated: 03/06/2016 - 14:44:16

2

FIGURE 3.8: Final design summary, page 2.

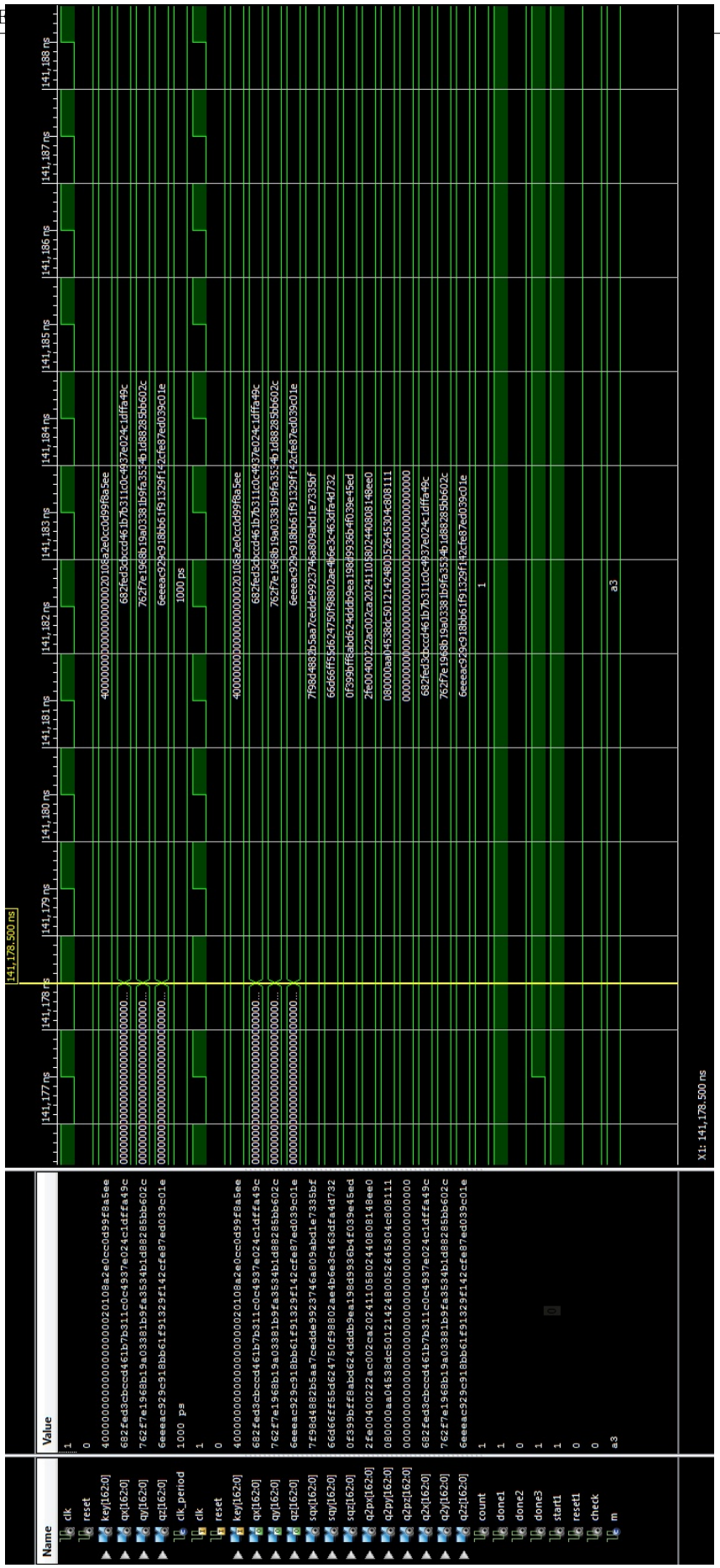


Figure 3.9: ECC-simulation results.

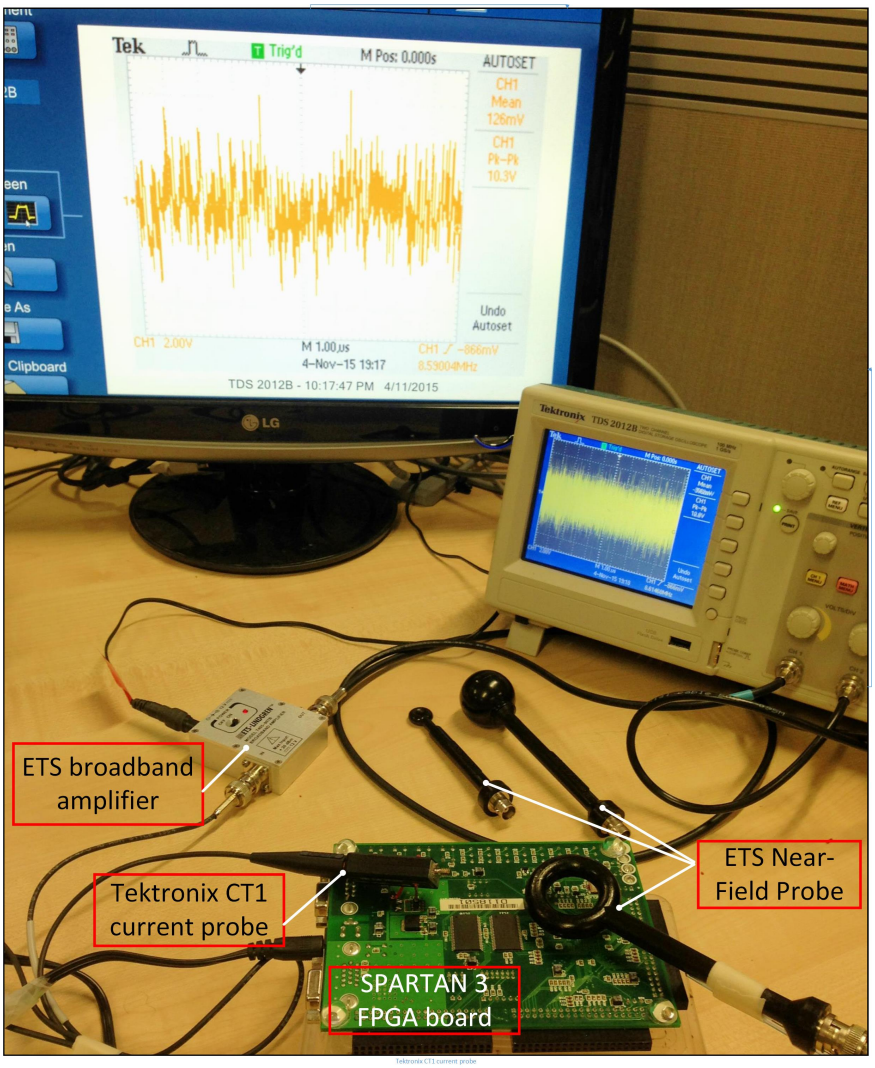


FIGURE 3.10: Measurement setup for side-channel attack.

ROM, so designs can easily be made non-volatile. It is fully compatible with all versions of the Xilinx ISE tools, including the free WebPack.

- **Oscilloscope:** In order to record and see power-signal traces, a Tektronix TDS2012 oscilloscope taking 1 Gigasamples per second is used.

- **Current probe:** for measuring the power consumption of our FPGA board a Tektronix CT1 is used. This Probe (Figure 3.11) is designed for permanent or semi-permanent in-circuit installation. It consists of a current transformer and an interconnecting cable. The current transformer has a small hole through which a current-carrying conductor is passed during circuit assembly. Details and considerations of using Tektronix CT1 are provided at (Appendix A).

- **Electromagnetic probe:** For measuring the electromagnetic emission of the

Figure 3.11: Tektronix CT1 current probe[1]

FPGA board, different commercial and hand-crafted probes are used and tested (Appendix A), and finally an ETS near-field probe set (model 7405) (Figure 3.12) is selected. This set consists of three loop probes, one stub and one ball probe, an extension handle and an optional battery-powered pre-amplifier. The handle of each probe terminates in a BNC connector. These probes are designed to be used with a signal-analysing device such as an oscilloscope or a spectrum analyser.

Figure 3.12: ETS near-field probe set (model 7405)[2]

- **ETS broadband amplifier:** An ETS broadband amplifier (model 7405-907b) (Figure 3.13) is utilised to increase the sensitivity of the oscilloscope and enhance the quality of the input signal traces.

- **PC:** This experiment was performed with a PC configuration of Intel Core i5, 2.8 GHz, and 16.00 GB RAM.

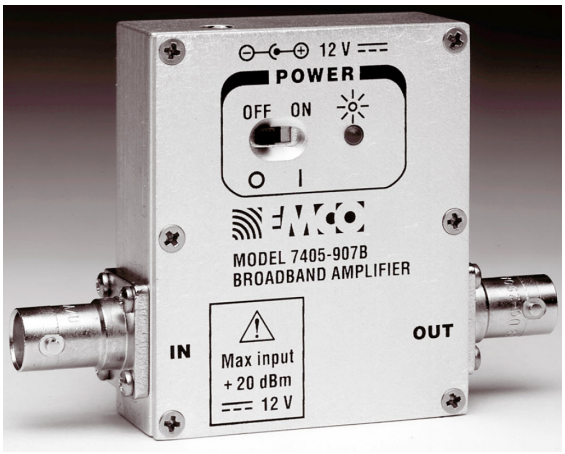


FIGURE 3.13: ETS broadband amplifier (model 7405-907b)[3]

3.5.1 Physical Attack Consideration

The FPGA board uses three discrete regulators to generate the necessary voltages, which can be used as the target of current measurement (Figure 3.14).

- V_{CCINT}: dedicated internal core logic power supply pin. The number of V_{CCINT} pins depends on the package used. All must be connected to +1.2V (red squares in Figure 3.14).
- V_{CCAUX}: dedicated auxiliary power supply pin. The number of V_{CCAUX} pins depends on the package used. All must be connected to +2.5V (orange squares in Figure 3.14).
- V_{CCO}: dedicated I/O bank, output buffer power supply pin. Along with other V_{CCO} pins in the same bank, this pin supplies power to the output buffers within the I/O bank and sets the input threshold voltage for some I/O standards (pink squares in Figure 3.14).

Regarding our experiment, V_{CCINT} is the best source for measuring the power consumption of FPGA for side-channel analysis purpose. Appendix A explain how to install current probe to measure this power.

3.6 Summary

In this chapter, elliptic-curve cryptography (ECC) is introduced as a promising alternative for public-key cryptosystems. In addition, its physical vulnerabilities and implementation considerations are discussed. ECC offers two major benefits over conventional cryptographic algorithms; it has more security per bit and a suitable key size for hardware and modern communications. Thus, this results in smaller public-key certificates, lower power requirements and smaller hardware processors. In this project

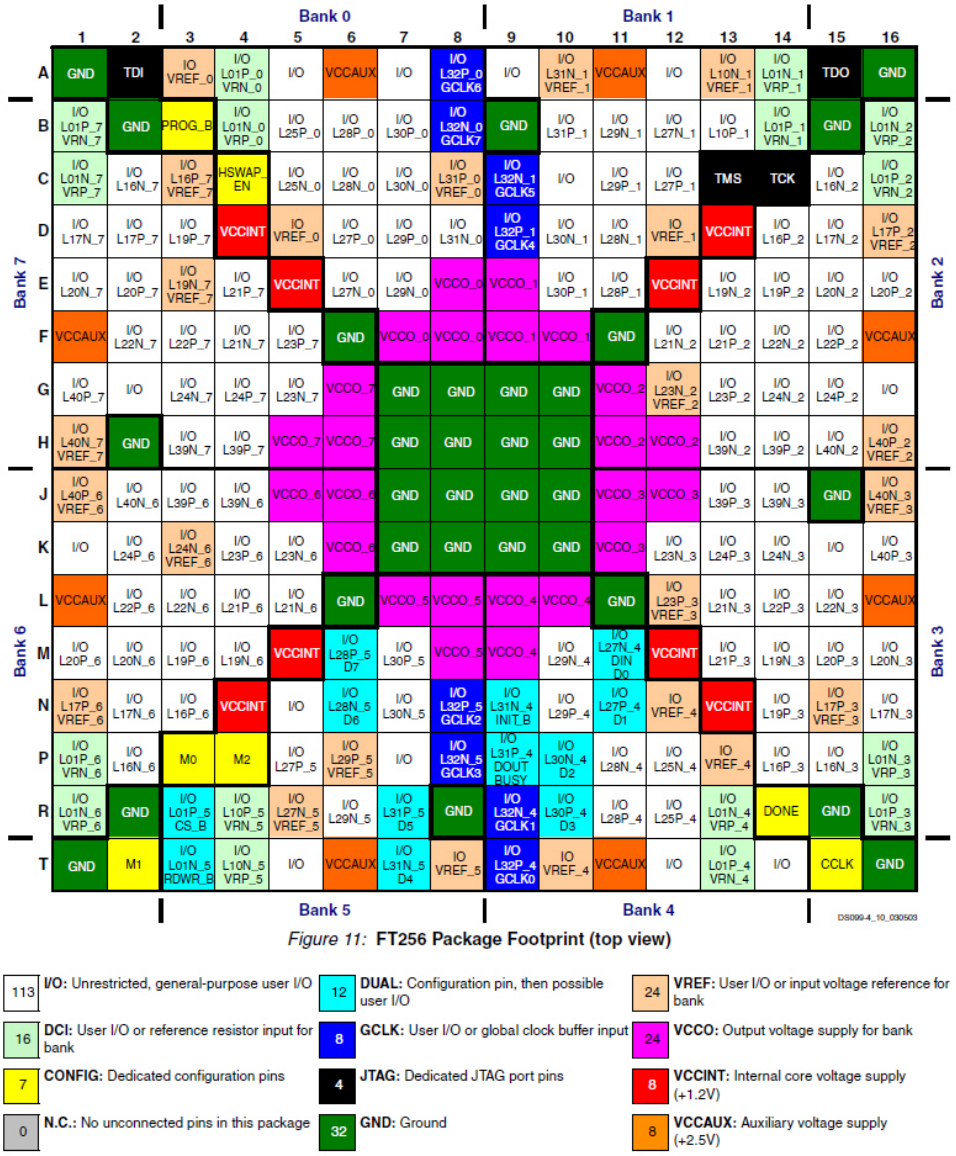


Figure 11: **FT256 Package Footprint (top view)**

FIGURE 3.14: Spartan-3 FPGA, FT256-package footprint. V_{CCINT} (shown with red squares) are the target of current measurement.

an EC over GF(2^m) will be the emphasis because it is very efficient for hardware implementation.

4 Pre-processing Stage

4.1 Introduction

In this chapter PCA is used to address a common problem of SCA. In practice, one difficulty in analysing the side-channel information is how to handle the huge amount of data collected in physical measurements. The huge size of data becomes a very challenging task for attackers to discover or to extract valuable information. With the help of data mining techniques this task became simplified. Data Mining is the process of searching for valuable information in a large volume of data stored in many databases, data warehouses or any other information repositories [89]. This technique is a highly interdisciplinary area spanning a range of disciplines like Statistics, Machine Learning, Databases, Pattern Recognition and others. Different terms are used for data mining techniques in the literature, such as feature extraction, data/pattern analysis and dimension reduction.

Side-channel information analysis also involves a the dataset with a huge amount of data. The increase of data size in terms of number of instances and number of features present a complex task to perform the analysis. This is due to rate of the sampling in a recording device. A high sampling rate is usually mandatory in order to retain the frequency content of the side channel. Hence, dimensionality reduction is an important research area in the field of side-channel data analysis. The main objective of dimensionality reduction is to transform high-dimensional data samples into a low-dimensional space. Once the dimensionality gets reduced, it helps to improve the robustness of the classifier and reduces the computational complexity.

Recently there are increasing interests in developing feature variable selection or dimension reduction approaches for side channel analysis approaches. Most of the methods are heuristic in nature. In the pattern recognition literature numerous statistical feature-selection criteria and search algorithms have been developed [90, 91].

It has been shown that neural networks are able to perform certain non-linear principal component analysis (PCA) [92–94]. In [95] and [96] PCA is used to reduce high-dimensional spectral data and consequently reduce the computational complexity. Karhunen and Joutsensalo [97] discussed many aspects of PCA performed by neural networks. Battiti [98] proposes to use mutual information as a guide to evaluate each feature's information content and select features with a high information content.

Publications pertaining to this chapter:

- E. Saeedi, Y. Kong, and M. S. Hossain, "Side Channel Attacks and Learning Vector Quantization", *Frontiers of Information Technology and Electronic Engineering, Springer, 2016.*(In press)
- E. Saeedi, M. S. Hossain, and Y. Kong, "Side Channel Analysis of an Elliptic Curve Crypto-system Based on Multi-Class Classification", *The Sixth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Denton, Texas, USA, pp. 1-7, July 13-15, 2015.*
- E. Saeedi and Y. Kong, "Side Channel Information Analysis Based on Machine Learning", *In Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on (pp. 1-7). IEEE.*

4.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) [99, 100] is a standard statistical tool which is widely used to reduce the dimensionality of a dataset consisting of an enormous number of interrelated variables. PCA reduces the dimensionality by transforming the original dataset into a new set of variables, called principal components, where the largest variance present in the original dataset is captured by the highest component in order to extract the most important information. In other words, it looks for a linear transformation that projects high-dimensional data into a low-dimensional subspace while preserving the data variance (i.e., it minimises the mean squared reconstruction error).

In order to minimise the loss of relevant information, PCA uses in two steps. First, it looks for a rotation of the original axes such that the new coordinate system indicates the successive directions in which the data have maximal variance. Second, it retains only the most important directions in order to reduce the dimensionality. It assumes therefore that the variability in the discarded directions corresponds to noise.

In order to illustrate the way PCA works, we give a small example for a two-dimensional (x,y) dataset with the same number of samples. In Figure 4.1 a plot of this dataset is given. The first principal component is required to have the largest variance. The second component must be orthogonal to the first component while capturing the largest variance within the dataset in that direction. These components are plotted in Figure 4.2 This results in components which are sorted by variance, where the first component captures the largest variance. If we transform the dataset using these principal components, the plot given in Figure 4.2 will be obtained. This

plot clearly shows that there is a larger variation in the direction of the first principal component.

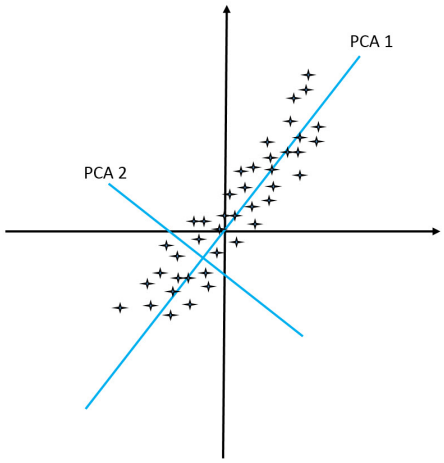


Figure 4.1: The plot of a dataset with first and second Principal Components.

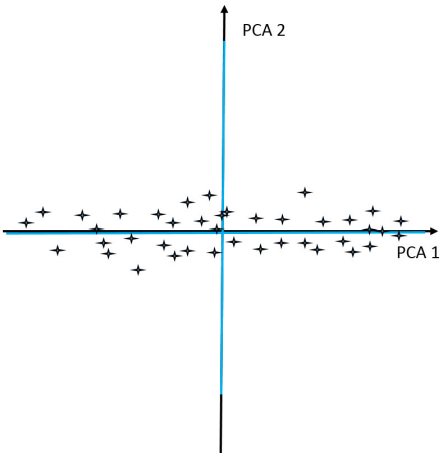


Figure 4.2: The dataset in Principal Component domain.

The computational steps of the PCA algorithm are given below [101].

- First, the mean from each dimension n of the traces T_i is calculated in a vector M_n.

$$M_n = \frac{\sum_{i=1}^{T} T_i, n}{n} \tag{4.1}$$

This mean M_n must be subtracted from each of the dimensions n for each trace T_i.

$$T_{i,n} = T_{i,n} - M_n \tag{4.2}$$

- Construct the covariance matrix $\sum$. This matrix will be an $n*n$ matrix where n is equal to the number of samples (dimension) of the power traces. The covariance for two dimensions X and Y is defined as follows:

$$\mathrm{Cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1} \tag{4.3}$$

  Using this formula for the covariance, the covariance matrix is defined as follows:

$$\sum^{n*n} = (Ci,j,Ci,j = \mathrm{Cov}(Dim_i, Dim_j)) \tag{4.4}$$

  where Dim_x is the xth dimension.

- Then calculate the eigenvectors and eigenvalues of the covariance matrix

$$\sum = U * A * U^{-1} \tag{4.5}$$

  where the eigenvalue matrix A is diagonal and U is the eigenvector matrix of $\sum$. These eigenvectors and eigenvalues provide information about the patterns in the data.
  The eigenvector corresponding to the largest eigenvalue is called the first principal component; this component corresponds to the direction with the most variance. Since n eigenvectors can be derived, there are n principal components. They have to be ordered from high to low based on their eigenvalues. Afterwards, the first principal component represents the largest amount of variance between the traces.

- Choose p components which one wishes to keep, and form a matrix with these vectors in the columns. This matrix is called the feature vector.

With this feature vector of length p, two things can be done. The original data can be transformed to retain only p dimensions, or the noise of the original dataset can be reduced using only some components while keeping all n dimensions.

4.2.1 PCA in Side-Channel Attacks

Side-channel information analysis also involves a dataset with a huge amount of data. Direct computations with the covariance matrices are inefficient as typically around five thousand measurements at about three thousand instants are taken for a practical attack. The number of samples depends on the sampling rate of the recording device. A high sampling rate is usually mandatory in order to retain the frequency content of the side channel. This leads to excessive computational loads and a prohibitively large memory usage. Furthermore, it is expected that only a limited number of time samples

are related to important information. Therefore, applying a signal preprocessing stage is necessary to identify a few important points in the signal (called points of interest or valuable features) and to simply discard the measurements at all other instants. This can be justified by the fact that usually the power consumption pattern depends only on few instants during which data is processed or when some internal state still depending on the data, for example the charge of a latch or a bus being overwritten. Principal component analysis as the most popular and efficient approach of preprocessing can be used in two ways, a PCA transformation (which can also be used to reduce the number of dimensions) and noise reduction of a dataset.

Figure 4.3 illustrate the overall scheme of the application of PCA in SCA. As can be seen, N different sample traces of side channel information, such as power consumption or electromagnet emission, with a big dimension of M_1, make a huge input matrix with the dimension of $N \times M_1$. After applying PCA as a space transformation tool, the dimension of dataset will reduce to $N \times M_2$, while $M_1 << M_2$; therefore, the input dataset would be ready for classification stage that explained at Chapter 5 and 6.

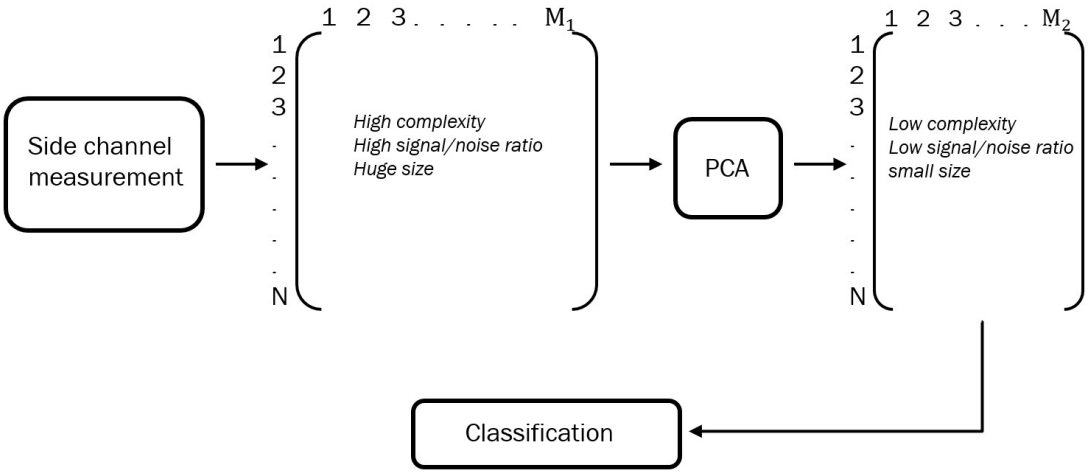


Figure 4.3: The application of PCA in SCA

PCA transformation

Power or electromagnetic traces usually have large dimensions (a high number of samples) which makes calculations computationally intensive. It would be a great improvement if the number of dimensions ccould be reduced without removing much relevant data. This might improve the computation time of analysis. PCA can be used to reduce the dimensionality of the dataset by removing the samples which do not relate to the leakage of the secret key. The feature vector U and the original data X can be used to derive this reduction. If we transpose the feature vector, we get the eigenvectors in the columns. By multiplying this vector with the transposed mean-adjusted data, we get the transformed dataset $\hat{X}$. This is the original data in terms of the chosen principal components.

$$Y = U^T * X^T = (X * U)^T$$

$$\hat{X} = Y^T = ((X * U)^T)^T = X * U$$

Noise reduction

The main objective of PCA is to reduce the number of features and to remove irrelevant, redundant and noisy data [102]. By reducing the features, one can reduce the system complexity, over-fitting and increase the computational speed. If we do not want to remove dimensions, but instead would like to reduce the noise in the traces in order to keep only the most relevant information, we can use the chosen Principal Components to retain only the most important information. If we choose to retain all principal components, this method returns the original data since no information is discarded. Normally, one would remove the components which contribute to the noise. The first step is the same as in a transformation: the feature vector U is transposed and multiplied with the transposed mean-adjusted data X. Then, in order to get the original data X back, this matrix is multiplied by the feature vector. In order to get the rows and columns correct, we transpose this final matrix Z. Because we want the original data back, we have to add the mean, which was subtracted in the first step, to each dimension.

$$Y^T = X * U$$
$$Z = U * (X * U)^T = U * U^T * X^T = X^T$$
$$\hat{X} = Z^T = (X^T)^T = X$$

4.3 Experiments based on PCA

The basic measurement setup used to performed this experiment are an FPGA board with a SPARTAN 3 FPGA, a Tektronix TDS2012 oscilloscope with 1 Gigasample per second to record the power/electromagnet signal traces, a Tektronix CT1 current probe for measuring power consumption of FPGA, an ETS near-field probe set (model 7405) for measuring electromagnetic emission and an ETS broadband amplifier (model 7405-907b) to enhance the quality of the input signal traces (see Figure 3.10). This experiment was performed through a MATLAB R2015a toolbox and a PC configuration of Intel Core i7, 2.8 GHz and 16.00 GB RAM.

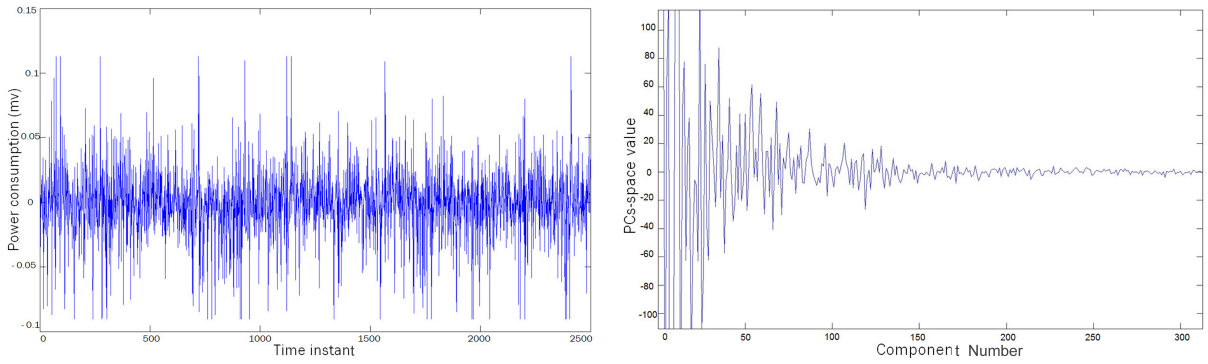
In order to utilise machine learning techniques for analysing side-channel information of a cryptosystem, several steps should be accomplished; for example physical probing is needed to measure the data leakage of the cryptosystem, a preprocessing stage in which the recorded signals should be prepared for the classification stage. Having applied PCA as a preprocessing stage, different techniques of machine learning are used to characterise the side-channel information. In this project, machine learning is used in terms of Support Vector Machine (SVM) and Neural Network classifiers. The theoretical and experimental background of these classifiers is discussed in Chapters 5 and 6. In this section, the experimental results in terms of PCA application will be discussed.

Our input dataset contains 16 classes, each of which relates to a particular key value and includes 320 different samples; therefore we have a huge dataset where the

dimensionality is equal to the product of the number of sample traces (N = 16 * 320 = 5120) and the number of sampling times (2500). Because of the high-frequency sampling in our measurements and oscilloscope, the recorded signal traces are noisy and high-dimensional which cause computational complexity and inaccuracy. Therefore, PCA is utilised to prepare the input dataset for the classification algorithm by reducing the noise and number of dimensions of the signals without missing their main features.

4.3.1 Experimental Results and Discussion

Figure 4.4.a shows a sample trace before applying PCA in the time domain, an Figure 4.4.b shows the signal after transforming into the space of the PCs. As can been seen, the most important components are the first hundred, and then by roughly 300 components the values decrease considerably to a range of ±5 and finally reach zero by 987 components (note that only the first 300 components, which are the most important ones, are shown in Figure 4.4.b). This reduction is due to the fact that PCA transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric.



(a) Power consumption trace in time domain. (b) Power consumption trace in PCA domain.

Figure 4.4: Comparison of input dataset in time and PCA domains.

In an experiment based on a Support Vector Machine (SVM), the accuracy of classifiers with different kernel functions (MLP, POLY and RBF) and also RBF sigma orders are investigated. As expected, using the recorded data in the time domain as the input dataset for the classification stage resulted in inaccuracy (accuracy range between [35% 57%]), while after applying PCA the accuracy significantly improved. Figure 4.5 illustrates the accuracy of an RBF kernel function with different sigma orders in regard to the number of components. As can be seen, the accuracy swiftly improves to a maximum of approximately 98% for PCs $\leq$ 300, because the most important and valuable

information is within the first 300 components. Afterwards, adding more components up to about 1500 does not affect the accuracy. The decreased accuracy when using the last 1000 components is because of these components belong to the least important data and noise. In terms of computational complexity, the processing time of SVM

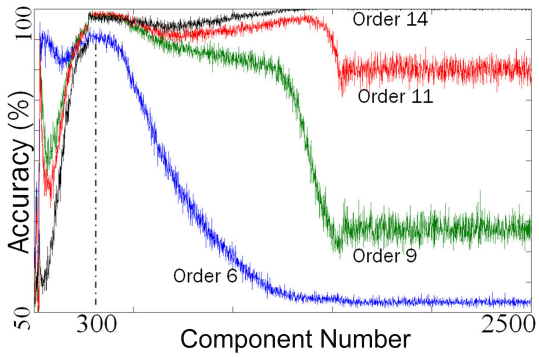


FIGURE 4.5: The observed electromagnetic emission signal trace

classifiers with different kernel functions (Poly, Mlp and RBF) is calculated. Figure 4.6 shows the experimental computational complexity as a function of the component number. From this figure, as the number of components increases, the computational complexity proportionally increases.

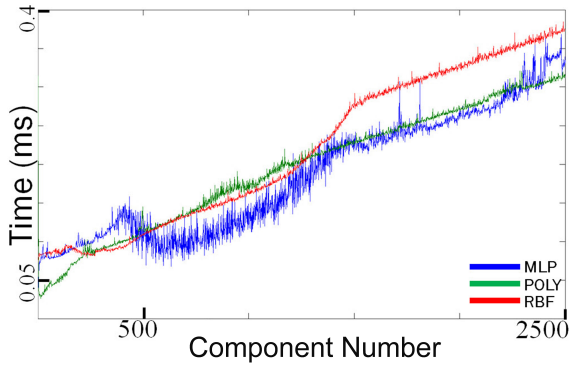


FIGURE 4.6: Computational complexity of SVM classifiers with different kernels

In another experiment, SVM multi-class classification was implemented through four different approaches (WW [103], CS [104], LLW [105], M-SVM^2 [106]), more details are provided in Chapter 5. Figure 4.7 presents the performance of these algorithms as a function of component number. As shown, the error ratios of the CS and M-SVM^2 algorithms are about 25% by considering roughly 250 components while those of LLW and WW get stable at about 3% by using roughly the same number of components. Hence by utilising PCA the complexity can be reduced by decreasing the dimension of the input data from 2500 to 250 components, without affecting accuracy.

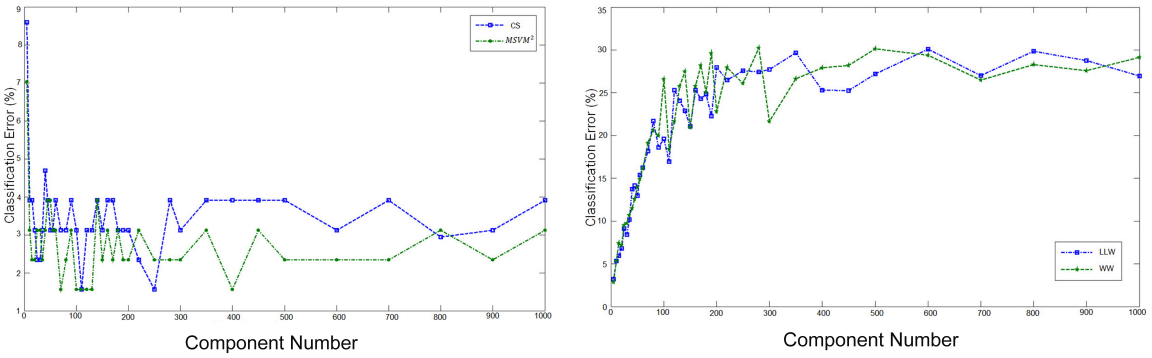


Figure 4.7: The performance of SVM multi-class classifiers as a function of the number of principal components.

Neural-network analysis of side-channel information strongly depends on PCA dimension reduction. We used different architectures of neural networks to verify the performance of side-channel characterisation based on neural networks (see Chapter 6). This experiment was performed through a MATLAB toolbox and a PC configuration of Intel Core i5, 2.8 GHz and 4.00 GB RAM. Without using PCA the computational complexity of the program was very high; some network architectures get stuck because of insufficient memory space, and other architectures were significantly slow. Therefore, for an efficient and applicable attack in real time, PCA needs to be applied to decrease the time and memory consumption. Having applied PCA , some successful attacks performed with the same pc configuration. The memory consumption of these attacks ranged between [1 3] GB and processing time between [10 8000] seconds.

4.3.2 Summary

In this chapter PCA is used to address a common problem of SCA. For the measurement stage of side-channel attacks, a high sampling rate is usually mandatory in order to retain the frequency content of the side channel. This, results in a huge input dataset, high computational complexity and inaccuracy, presenting a very challenging task for attackers to discover or extract valuable information. To overcome these problems, PCA is utilised as a powerful tools for dimensional reduction and feature extraction of the input dataset. Regarding our experimental results, not only can PCA significantly reduce the time and memory consumption of the program, but it can reduce the noise without affecting accuracy.

5

Support-Vector Machine as a Side-Channel Classifier

5.1 Introduction

In this chapter a powerful and efficient method of SCA based on machine learning techniques in the form of support-vector-machine (SVM) classification is verified. In literature, SVM has been used a few times as a powerful tool of classification for different types of side channel analysis, such as AES algorithm based on ATMEGA-256 micro controller in [33] and [95]. To the best of our knowledge, there has not been any attack on FPGA implementation of ECC to date. In this project, after applying PCA as a preprocessing stage, a SVM with an appropriate kernel and parameters is proposed to characterise the power consumption of the FPGA, measured while it executes an elliptic curve point multiplication, which is the main operation for an elliptic curve cryptosystem (ECC). In addition, a multi-class SVM classifier is utilised as a powerful and robust algorithm for characterisation of side-channel information. Finally, different models of classification, kernel functions and parameters, which can significantly affect the performance of classifiers, are investigated.

Publications pertaining to this chapter:

- E. Saeedi and Y. Kong, "Side Channel Information Analysis Based on Machine Learning", *In Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on (pp. 1-7). IEEE.*

- E. Saeedi, and Y. Kong, "Support Vector Machines in Side Channel Analysis", *the International Symposium on Information Theory and Its Applications. ISITA2014.*

- E. Saeedi, M. S. Hossain, and Y. Kong, 'Multi-class SVMs Analysis of Side-Channel Information of Elliptic Curve Cryptosystem'", *2015 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), pp. 1-6, Chicago, IL, USA, July 26-29, 2015.*

5.2 Support-Vector Machine (SVM)

Support-Vector Machines are among the most robust and successful pattern classification algorithms [4, 107]. There are different types of patterns, namely linear and non-linear. Linear patterns are patterns that are easily distinguishable or can be easily separated in low dimensions, whereas non-linear patterns are patterns that are not easily distinguishable or cannot be easily separated, and hence these patterns need to be further manipulated so that they can be easily separated.

Basically, the main idea behind SVM is the construction of an optimal hyperplane, which can be used for classification, for linearly separable patterns. The optimal hyperplane is a hyperplane selected from the set of hyperplanes for classifying patterns that maximises the margin of the hyperplane, i.e. the distance from the hyperplane to the nearest point of each patterns. The main objective of SVM is to maximise the margin so that it can correctly classify the given patterns, i.e. the larger the margin size the more correctly it classifies the pattern.

In the literature, SVMs are known to generalise well even in high-dimensional spaces with small training samples [108], and have been shown to be superior to the traditional empirical risk minimisation principle employed by most neural networks [109] SVMs have been successfully applied to a number of applications ranging from face detection, verification, and recognition [108–110] to object detection and recognition [111–113].

5.2.1 Linear Support Vector Machines for Linearly Separable Case

The basic idea of SVMs is to construct a hyperplane as the decision plane, which separates the positive (+1) and negative (-1) classes with the largest margin, and is related to minimising the VC dimension of the SVM. In a binary classification problem where feature extraction is initially performed, let us label the training data $x_i \in R^d$ with a label $y_i \in$ {-1, +1}, for all training data $i = 1, ..., l$, where l is the number of data value, and d is the dimension of the problem. When the two classes are linearly separable in R^d , we wish to find a separating hyperplane which gives the smallest generalisation error among the infinite number of possible hyperplanes. Such an optimal hyperplane is the one with the maximum margin of separation between the two classes, where the margin is the sum of the distances from the hyperplane to the closest data points of each of the two classes. These closest data points are called Support Vectors (SVs). The solid line on Figure 5.1 represents the optimal separating hyperplane . Let's suppose they are completely separated by a d-dimensional

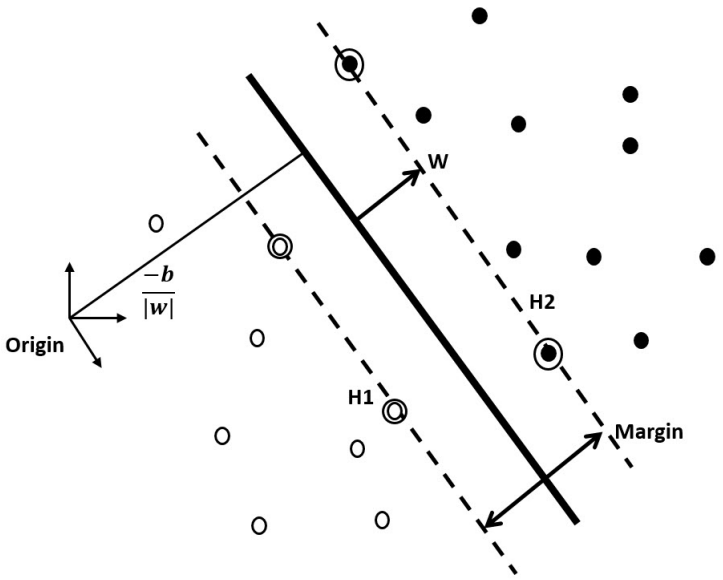


Figure 5.1: Linear separating hyperplanes for the separable case. The support vectors are circled (taken from [4]).

hyperplane described by

$$W \ . \ X + b = 0 \tag{5.1}$$

The separation problem is to determine the hyperplane such that $W \ . \ X_i + b \geq +1$ for positive examples and $W \ . \ X_i + b \leq -1$ for negative examples. Since the SVM finds the hyperplane which has the largest margin, it can be found by minimising $1/2 \parallel W \parallel^2$

$$\min_{W,b} \Phi(W) = \frac{\|W\|^2}{2} \tag{5.2}$$

The optimal separating hyperplane can thus be found by minimising equation 5.2 under the constraint 5.3 to correctly separate the training data.

$$y_i \ (x_i \ . \ W + b) - 1 \geq 0, \forall_i \tag{5.3}$$

This is a Quadratic Programming (QP) problem for which standard techniques (Lagrange Multipliers, Wolfe dual) can be used [114–116].

5.2.2 Linear Support-Vector Machines for Non-Separable Case

In practical applications for real-time data, the two classes are not completely separable, but a hyperplane that maximises the margin while minimising a quantity proportional to the misclassification errors can still be determined. This can be done by introducing positive slack variables ξ_i in constraint 5.3, which then becomes

$$y_i \ (x_i \ . \ W + b) \geq 1 - \xi_i, \forall_i \tag{5.4}$$

If an error occurs, the corresponding ξ_i must exceed unity, so $\Sigma_i \xi_i$ is an upper bound for the number of misclassification errors. Hence the objective function 5.2 to be minimised can be changed into

$$\min \{ \frac{\|W\|^2}{2} + C \sum_{i=1}^{l} \xi_i \} \tag{5.5}$$

where C is a parameter chosen by the user that controls the trade-off between the margin and the misclassification errors. A larger C means that a higher penalty is assigned to misclassification errors. Minimising equation 5.5 under constraint 5.4 gives the Generalized Separating Hyperplane. This remains a QP problem. The non-separable case is illustrated in Figure 5.2.

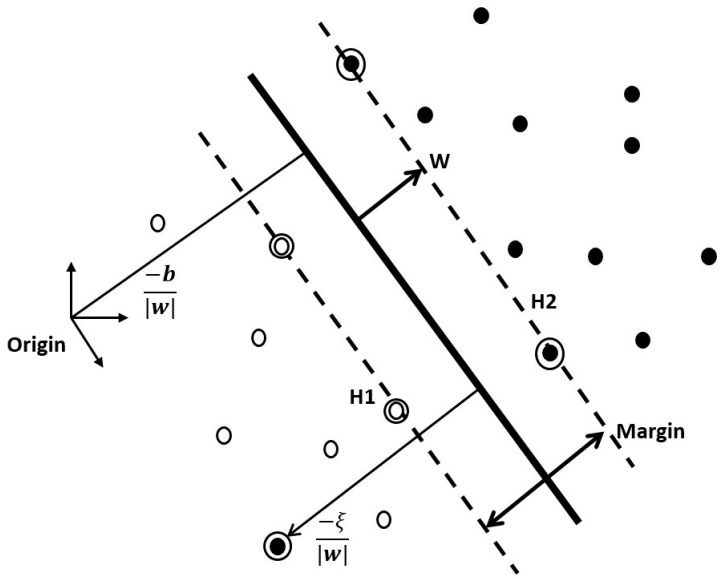
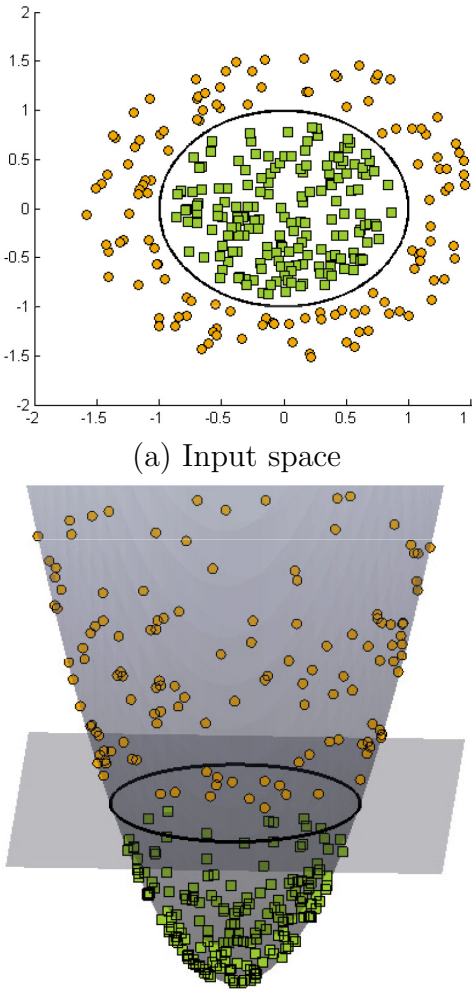


FIGURE 5.2: Linear separating hyperplane for the non-separable case (taken from [4])

.

5.2.3 Non-linear Support-Vector Machines

An extension to non-linear decision surfaces is necessary since real-life classification problems are hard to solve using a linear classifier [117]. When the decision function is not a linear function of the data, the data will be mapped from the input space into a high dimensional feature space by a non-linear transformation. In this high dimensional feature space, the generalised optimal separating hyperplane shown in Figure 5.8 is constructed [116]. Cover's theorem states that, if the transformation is non-linear and the dimensionality of the feature space is high enough, then the input space may be transformed into a new feature space where the patterns are linearly separable with high probability. This non-linear transformation is performed in an implicit way through so-called kernel functions.



(a) Input space

(b) Feature space

Figure 5.3: Feature space is related to input space via a non-linear map, causing the decision surface to be nonlinear in the input space (taken from [5])

Non-linear Support Vector Machines and Kernels

Kernel methods are a set of approaches to map data from the original space onto the kernel space (Hilbert space) without ever knowing the mapping function explicitly. The concept of kernel methods is best known in SVM due to its successful application to enable SVMs to deal with problems of non-linear separation in the original feature space. SVM could find a linearly separable hyperplane in the kernel space by the aid of kernel methods. Many kernels are used in SVM, such as the linear kernel, the polynomial kernel, and the Gaussian RBF kernel. Among these kernels, the Gaussian RBF kernel is most commonly used because of its attractive features [118, 119], such as structure preservation. This kernel is adopted in many papers. The parameter p in this kernel is crucial and is essential to robust performance of the SVM [119], while an arbitrary predefined p cannot guarantee a satisfactory performance. An exhaustive search for an appropriate p is intractable since the domain of definition for p ranges

from zero to infinite. The SVM becomes over-fitting when p is set as zero, since all training instances are support vectors in this case. As a result, the SVM has a perfect prediction for a training set but may have a poor performance on a test set. When p is set as infinite, under-fitting occurs in an SVM because all training instances are considered the same as one instance. In some papers [120, 121], p is specified by a default value, such as one, instead of doing p selection.

In order to accomplish a non-linear decision function, an initial mapping Φ of the data into a (usually significantly higher-dimensional) Euclidean space H is performed as $\Phi : R^n \rightarrow H$, and the linear classification problem is formulated in the new space with dimension d. The training algorithm then depends on the data only through a dot product in H of the form $\Phi(x_i).\Phi(x_j)$. Since the computation of the dot products is prohibitive if the number of training vectors $\Phi(x_i)$ is very large, and since Φ is not known a priori, Mercer's theorem [4] for positive definite functions allows us to replace $\Phi(x_i).\Phi(x_j)$ by a positive definite symmetric kernel function $K(x_i, x_j)$, that is, $K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$. In the training phase, we need only the kernel function $K(x_i, x_j)$, and $\Phi(x_i)$ does not need to be known since it is implicitly defined by the choice of kernel $K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$. The data can become linearly separable in feature space although the original input is not linearly separable in the input space. Hence kernel substitution provides a route for obtaining non-linear algorithms from algorithms previously restricted to handling linearly separable datasets [122]. The use of implicit kernels allows reducing the dimension of the problem and overcoming the so-called "dimension curse" [123]. Variant learning machines are constructed according to the different kernel functions $K(x, x_i)$ and thus construct different hyperplanes in feature space.

5.2.4 SVMs Applied to Multi-Class Classification

The basic SVMs are for the two-class problem. However it should be extended to multi class to classify into more than two classes [124, 125]. There are two basic strategies for solving q-class problems with SVMs.

Multi-class SVMs: One to Others

Take the training samples with the same label to be one class and the others to be the other class, then it becomes a two-class problem. For the q-class problem $(q \geq 2), q$ SVM classifiers are formed and denoted by SVM_i, $i = 1, 2, ..., q$. As for the testing sample x, $d_i(x) = W_i \ . \ X + b_i$ can be obtained by using SVMi. The testing sample x belongs to the jth class where

$$d_j(x) = \max_{i=1 \sim q} d_i(x) \tag{5.6}$$

Multi-class SVMs: Pairwise SVMs

In the pairwise approach, q^2 machines are trained for q-class problems [114] . The pairwise classifiers are arranged in trees, where each tree node represents an SVM,

Figure 5.4. A bottom-up tree, which is similar to the elimination tree used in tennis tournaments, was originally proposed in [114] for recognition of 3D objects. A top-down tree structure has been recently published in [6]. There is no theoretical analysis of the two strategies with respect to classification performance [126]. Regarding the training effort, the one-to-others approach is preferable since only q SVMs have to be trained compared to q^2 SVMs in the pairwise approach. However, at runtime both strategies require the evaluation of $q-1$ SVMs [126]. Recent experiments on people recognition show similar classification performances for the two strategies [113].

Multi-class SVMs: Implementation

Different approaches have been proposed for the canonical extension of binary SVMs to multiple classes; some of the most popular ones are

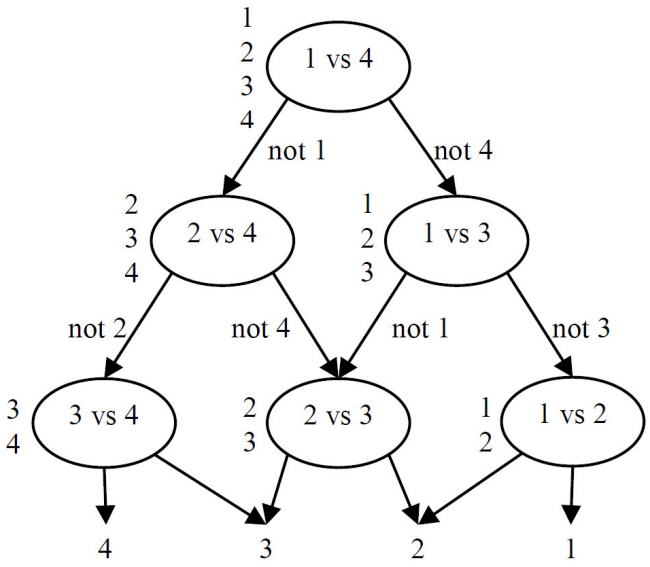
- Weston and Watkins (**WW**) SVM [103].
- Crammer and Singer (**CS**) SVM [104].
- Lee, Line and Wahba (**LLW**) SVM [105].
- Guermeur and Monfrini, **M-SVM^2** [106].

The WW and LLW methods are theoretically sound, and experiments indicate that they lead to well-generalised hypotheses, but efficient training algorithms are not available. In the CS approach the most popular modification of the WW formulation, mainly to speed-up the training process, is proposed. The M-SVM^2 can be seen as a direct extension of the 2-norm SVM to the multi-class case, which is established by deriving the corresponding generalised radius-margin bound.

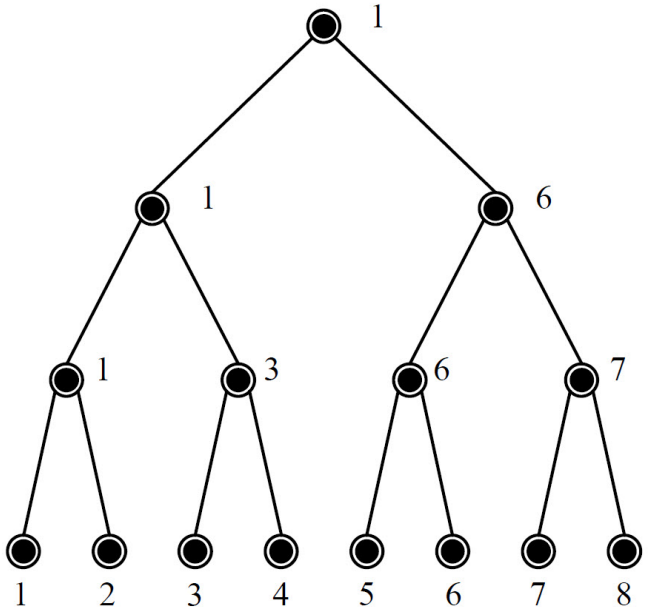
5.3 Experiments based on SVMs

The basic measurement setup used to perform this experiment inclydes an FPGA board with a SPARTAN 3 FPGA, a Tektronix TDS2012 oscilloscope with 1 Gigasample per second to record the power/electromagnet signal traces, a Tektronix CT1 current probe for measuring power consumption of FPGA, an ETS near-field probe set (model 7405) for measuring electromagnetic emission and an ETS broadband amplifier (model 7405-907b) to enhance the quality of the input signal traces (see Figure 3.10). This experiment was performed through a MATLAB R2015a toolbox and a PC configuration of Intel Core i7, 2.8 GHz and 16.00 GB RAM. For more details, refer to 3.5.

One of the main problems in this experiment is handling a huge input dataset. The increase of data size in terms of number of instances and number of features make it a complex task to perform the analysis. This is due to the rate of the sampling in recording device. A high sampling rate is usually mandatory in order to retain the frequency content of the side channel. Hence, principal component analysis (PCA) is applied as a preprocessing stage to reduced the noise and dimensions of the signals



(a) Example of top-down tree structure



(b) Example of bottom-up tree structure

FIGURE 5.4: Tree structure for multi-class SVMs. (a) The decision Directed Acyclic Graph (DAG) for finding the best class out of four classes. The equivalent list state for each node is shown next to that node, (b) The binary tree structure for 8 classes (taken from [6]).

without missing the in main features. The theoretical and experimental results in regard to PCA are discussed in Chapter 4. In this section, the input dataset is the power consumption signal traces after applying PCA (in the PCA domain). To the best of our knowledge, there has not been any attack on the same cryptosystem specification (FPGA implementation of ECC) to date; therefore, in this project, different models of SVM classification, kernel functions and parameters, which can significantly affect the performance of classifiers, are investigated.

5.3.1 Experimental Results based on SVM Binary Classification

In this experiment power/electromagnetic signal traces were measured while it executed an elliptic curve point multiplication, which is the main operation for an elliptic curve cryptography (ECC). Our dataset contains 1,000 traces each of which has 2,500 sampling data. According to SVM classifiers, the two most popular kernel functions (RBF and POLY) are applied, and their overall classification accuracy along with computational complexity as a function of the number of principal components is provided. To calculate the accuracy with a given number of principal components, we used a ten-fold cross-validation for each number of components, because it is a popular type to compute the accuracy of statistical problems. In this algorithm the dataset is divided into ten subsets, and the classifier algorithm is repeated ten times. Each time, one of the ten subsets is used as the test set and the other nine subsets are put together to form a training set. Then the average accuracy across all ten trials is computed.

Figure 5.5 illustrates the accuracy of an SVM classifier with RBF and POLY kernel functions as a function of the number of PCs. As can be seen from this graph, as PCs are added, accuracy is increased for both kernel functions. Considering the graph, no more than 600 PCs are required to achieve a maximum accuracy of approximately 95%, therefore the dimension of the traces can be reduced from 2500 to 600 without affecting the accuracy. In side-channel data classification, the RBF kernel function seems to be more accurate than the POLY kernel function, especially for less than 600 PCs, however the accuracy of the RBF function does not change when the number of principal components increases from 100 to 250.

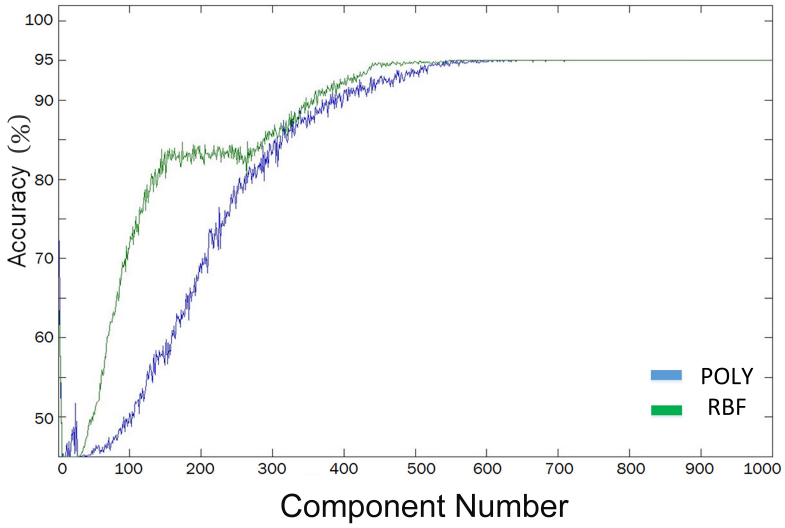


Figure 5.5: The accuracy of RBF and POLY kernel functions in an SVM classifier as a function of the number of PCs.

To calculate the computational complexity, the processing duration of the algorithm is calculated and used as a scale of complexity. Figure 5.6 shows the comparison

between the RBF and POLY kernel functions based on the computational complexity as a function of the number of principal components. As the graph illustrates, the complexity increases two to three-fold when the number of PCs increases from 100 to 2500. In addition, although the complexity of the RBF kernel function seems to be always greater than that of the POLY kernel function, from 500 to 600 components, which is the ideal number of component in term of the accuracy (see Figure 5.5), both functions have almost the same complexity.

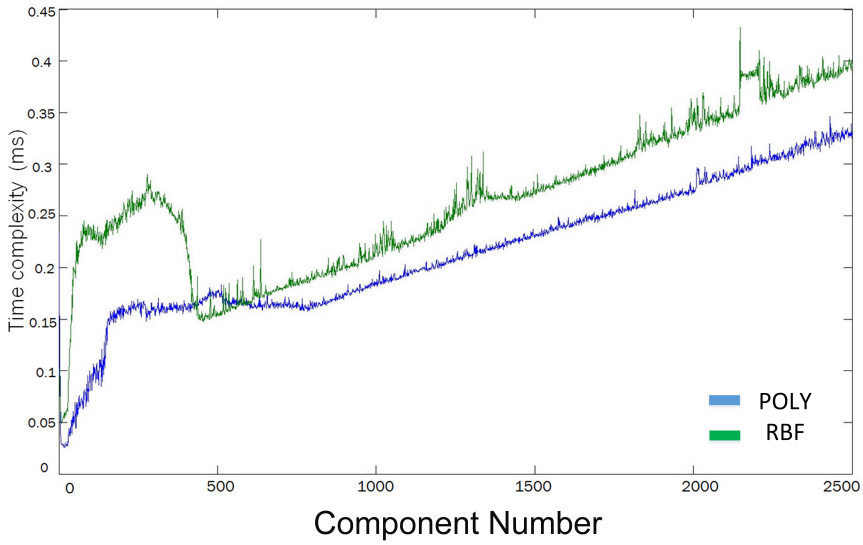


FIGURE 5.6: The computational complexity of RBF and POLY kernel functions in SVM classifier as a function of the number of PCs.

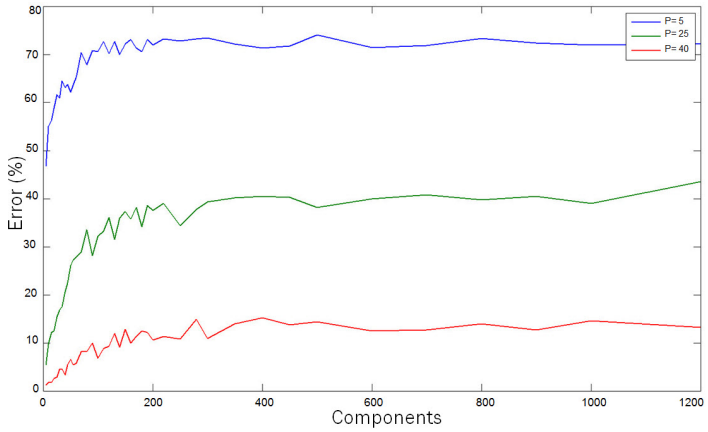
5.3.2 Experimental Results based on different Kernels of SVM Multi-class Classification

In this experiment, the multi-class classification procedure is performed through MSVMpack 1.5 [127] which is an open-source package dedicated to multi-class support-vector machines and can handle classification problems with more than two classes. Also different kernel functions in this package are verified, such as the linear Homogeneous polynomial, Non-homogeneous polynomial kernel and the Gaussian RBF kernel. The performance of these kernels and their parameters, which can significantly affect the result, are investigated. The accuracy of algorithm is calculated based on the difference between the instances in a classified class dataset (Output) and the instances in an actual class dataset (Target).

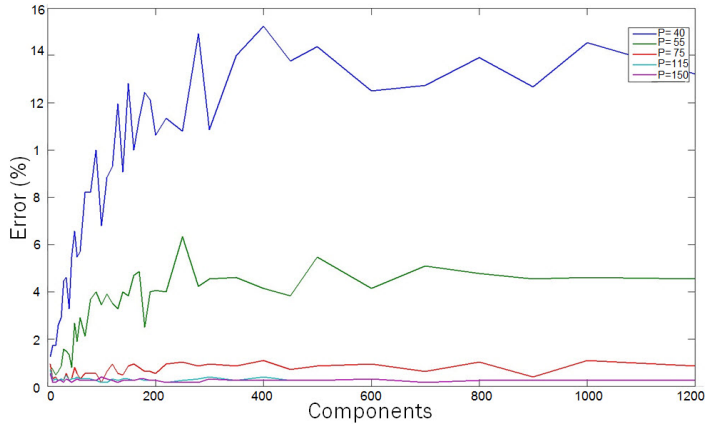
In order to show the efficiency of the applied classification approach based on the MSVM2 model, we tested and compared different kernel functions (Linear, Gaussian RBF, Homogeneous polynomial, Non-homogeneous polynomial). Based on our experiment, we infer that of all kernels only the Gaussian RBF is appropriate for our application, since the accuracy of the other kernel functions is less than 1%. Therefore,

we focused on the Gaussian RBF kernel function and verified the influence of p, which is the parameter in the Gaussian RBF kernel, namely the width of features (spread parameter).

Figure 5.7 illustrates the error of the multi-class classification of a 4-bit sub-key based on the Gaussian RBF kernel function with p in the range of 5 to 150. As can be seen, the error ratio decreases considerably as the parameter value increases. For example, the error of parameter 5 stabilised around 70% after 300 components while the corresponding errors for parameters 25, 40, 55 and 75 are roughly 35, 10, 4 and 1% respectively. Afterwards, the error ratio remains around 0.5% as the parameters increase to 150. Consequently, the most efficient classification would be by 100 components and an RBF parameter $p > 150$. The final accuracy of the classification of



(a)



(b)

Figure 5.7: The error ratio of different multi-class classifiers as a function of the number of principal components. (a) Error ratio for the Gaussian RBF with parameters 5, 25 and 40. (b) Error ratio for the Gaussian RBF with parameters 40, 55, 115 and 150

4-bit sub-key and 163-bit key values are provided in Table 5.1. This table compares the performance of different kernels based on the minimum number of components needed to achieve the best accuracy of classification. As shown, the classification accuracy of

Linear, Homogeneous polynomial and Non-homogeneous polynomial kernels are even less than 1% by applying 2500 components, while a Gaussian RBF with parameter $p > 115$ and applying about 100 components can reach 81% accuracy. Therefore, for side-channel analysis applications, a Gaussian RBF kernel with $p > 115$ is an efficient choice.

TABLE 5.1: Efficiency comparison of different kernel functions (Linear, Gaussian RBF, Homogeneous polynomial, Non-homogeneous polynomial) based on the minimum number of components and the best accuracy

Kernel	Parameter	4b Best Accuracy	163b Best Accuracy	Min Components
Linear	-	1 %	-	2500
Homo	-	1 %	-	2500
Non-Ho	-	1 %	-	2500
	p=5	30%	-	200
	p=25	63%	-	300
Gaussian	p=40	90 %	1.5%	350
RBF	p=55	96 %	19%	700
	p=115	99%	66%	250
	p=150	99.5%	81%	100

5.3.3 Experimental Results based on different Models of SVM Multi-class Classification

In order to show the efficiency of the multi-class classification approaches, we tested and compared different kernel functions (Linear, Gaussian RBF, Homogeneous polynomial, Non-homogeneous polynomial). Based on our experiment, we infer that of all the kernels only the Gaussian RBF is appropriate for our application, since the accuracy of the other kernel functions seems to be very low, under 3%. Therefore, we focused on the Gaussian RBF kernel function and verified the influence of p, which is an effectual parameter in the Gaussian RBF kernel, namely the width of features (spread parameter). Having determined a proper kernel function and parameters, the efficiency of different models of multi-classifiers (Weston and Watkins (**WW**) SVM [103], Crammer and Singer (**CS**) SVM [104], Lee, Line and Wahba (**LLW**) SVM [105], Guermeur and Monfrini, **M-SVM²** [106]) can be verified.

Table 5.2 presents the influence of the spread parameter p on the efficiency of multi-classifiers based on the Gaussian RBF kernel function. As can be seen, the error ratio decreases considerably for $p > 150$, however, to avoid over-fitting a limited range for

p should be determined. For this purpose, and considering Table 5.2, proper p-values for WW, CS, LLW and M-SVM^2 can be about 50, 5, 150 and 50 respectively.

Table 5.2: Error ratio of different models of multi-class classification based on Gaussian RBF.

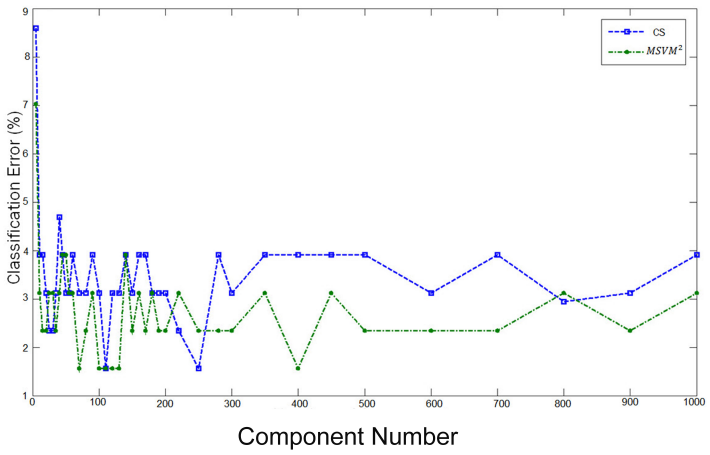
Models	**Spread Parameters of Gaussian RBF kernel**					
	s.p=([0,1]	**[1,3]**	**[5,10]**	**[20,30]**	**[50,70]**	**[150,200])**
WW	0.735	0.728	0.718	0.455	0.015	0.002
CS	0.716	0.314	0.003	0.001	0.001	0.001
LLW	0.718	0.739	0.736	0.376	0.02	0.003
M-SVM^2	0.718	0.726	0.432	0.028	0.003	0.002

By applying proper Gaussian parameters, we performed different models of multi-classifier on the outcome of PCA. Figure 5.8.a illustrates the error ratio of the multi-class classification based on **CS** and **M-SVM^2** algorithms as a function of the number of principal components, and Figure 5.8.b compares the corresponding error ratio of **LLW** and **WW** algorithms. As shown, the error ratios of **CS** and **M-SVM^2** algorithms are about 25% using roughly 250 components while those of **LLW** and **WW** get stable at about 3% by using roughly the same number of component. Hence the **CS** and **M-SVM^2** algorithms seem to be more efficient than **LLW** and **WW** in this application. In addition to our outcomes, by utilising PCA, the input data dimension can be reduced from 2500 to 250 without affecting accuracy.

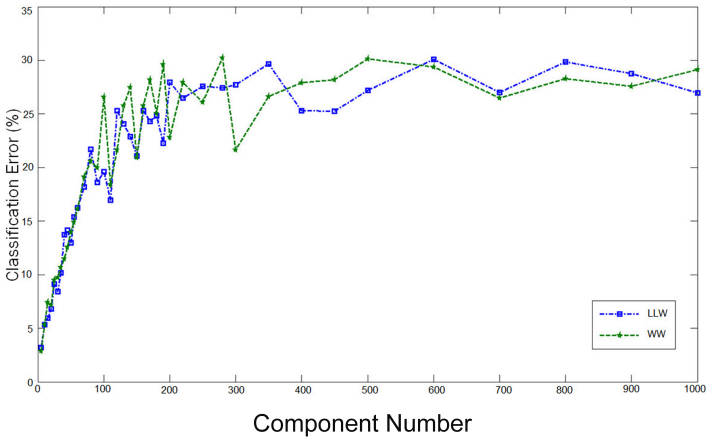
Table 5.3 compares the performance of different models of Gaussian-based multi-class classification, based on the minimum number of components and the best accuracy of classification. As shown, **WW** and **LLW** can classify the side-channel information to 77% accuracy, and for this accuracy they need to apply at least 400 components, while **CS** and **M-SVM^2** can reach 97% accuracy by applying about 350 and 500 components respectively. Therefore, for side-channel analysis applications, **CS** can be considered as the most efficient classifier model and **M-SVM^2** as the second most.

Table 5.3: The efficiency comparison of different models of Gaussian RBF classifiers.

Multi-class Classifiers	Best Accuracy	Min Components
CS	96%	350
MSVM^2	97 %	500
LLW	77 %	400
WW	77 %	400



(a)



(b)

FIGURE 5.8: The error ratio of different multi-class classifiers as a function of the number of principal components. (a) Classification based on **CS** and **M-SVM**^{2} algorithms. (b) Classification based on **LLW** and **WW** algorithms.

5.4 Conclusion

SVM-based characterisation of side-channel information is a powerful and feasible way to attack secure cryptosystems. In this chapter, an efficient approach based on SVM classification is presented. For this purpose, first PCA is applied as a preprocessing stage to improve the input dataset. Afterwards, the performance of different models of SVM is verified. Considering our experimental results, based on an FPGA implementation of ECC, by utilising PCA the input data dimension can be reduced from 2500 to 250 components without affecting accuracy. Moreover, after verifying the efficiency of different kernels, SVM models and parameters, it can be concluded that for SCA applications the best accuracy of multi-class classification would be based on the Gaussian RBF kernel function with parameter p value of 5 and 50 for **CS** and **M-SVM**^{2} SVM models respectively.

6 Neural Networks as Side-Channel-Information Classifiers

6.1 Introduction

In this chapter, in order to characterise side-channel information, multi-class classification based on different architectures of neural networks is investigated.

Neural networks have emerged as an important tool for multi-class classification. The recent vast research activities in neural classification have established that neural networks are a promising alternative to various conventional classification methods, see [34]. Neural networks are typically organised in layers; see Figure 6.1. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output. Most ANNs contain some form of 'learning rule' which modifies the weights of the connections according to the input patterns that it is presented with. The learning process involves updating network architecture and connection weights so that the network can efficiently perform a specific classification/clustering task.

Publications pertaining to this chapter:

- Ehsan Saeedi, Yinan Kong and Md Selim Hossain *The Evaluation of Neural Networks in Characterisation of Side-channel information.* (IEEE Embedded Systems Letters), (To be submitted).
- E. Saeedi, Y. Kong, and M. S. Hossain, "Feed-Forward Back-Propagation Neural Networks in Side-Channel Information Characterisation", *IEICE TRANSACTIONS on Communications.* (Under major revision)

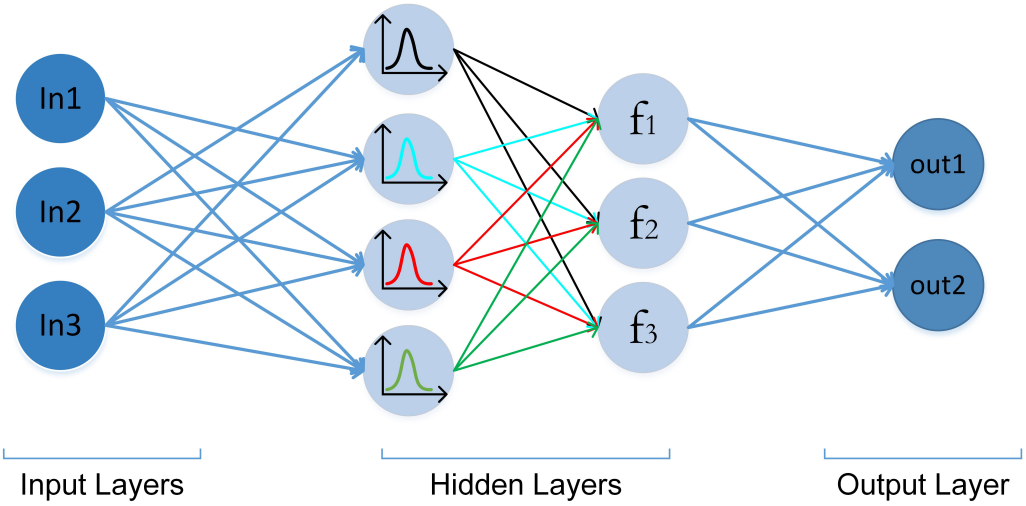


FIGURE 6.1: A simple two-layer hidden-layer neural-network architecture for pattern classification purposes.

- E. Saeedi, M. S. Hossain, and Y. Kong, "Side Channel Information Characterisation based on Cascade Feed-Forward Back-Propagation Neural Network", *Journal of Electronic Testing, May 2016, Springer.* (In press)

- E. Saeedi, Y. Kong, and M. S. Hossain, "Side Channel Attacks and Learning Vector Quantisation", *Frontiers of Information Technology and Electronic Engineering, Springer, 2016.*(In press)

6.2 Why Use Neural Networks?

The main characteristics of neural networks are that they have the ability to learn complex nonlinear input-output relationships, use sequential training procedures, and adapt themselves to the data. The advantage of neural networks lies in the following theoretical aspects. First, neural networks are data driven self-adaptive methods in that they can adjust themselves to the data without any explicit specification of functional or distributional form for the underlying model. Second, they are universal functional approximators in that neural networks can approximate any function with arbitrary accuracy [35], [90],[128]. Since any classification procedure seeks a functional relationship between the group membership and the attributes of the object, accurate identification of this underlying function is doubtlessly important. Third, neural networks are nonlinear models, which makes them flexible in modeling real world complex relationships. Finally, neural networks are able to estimate the posterior probabilities, which provides the basis for establishing classification rule and performing statistical analysis [129]. A list of neural network classifiers, which can be used as a promising approach of side-channel information analysis is provided below.

1. *Feed-Forward Back-Propagation (FFBP)*

2. *Probabilistic Neural Network (PNN)*

3. *Cascade-Forward Neural Network (CFNN)*

4. *learning Vector Quantisation neural networks (LVQ)*

In this chapter, each network will be introduced briefly and their efficiency for side-channel analysis of ECC cryptosystem will be verified.

6.3 Application of Neural Networks in SCA

A neural network learns the signature (power consumption and electromagnetic analysis) of an instruction, and then recognises it later automatically. For each instruction hundreds of structures need to be stored for a cryptosystem-processor. In more details, the general goal of attack is to obtain the secret key value which is stored in the cryptographic module from the measured power trace. Considering the value K_{sec} as a secret key stored in the attacked cryptographic module and K_{est} as the estimate value of the secret key, which was determined with a neural network, if the method works correctly the values K_{est} and K_{sec} will be equal at the end of the classification process. The first proposal of the method excepted sequential classification. In other words, classification is realised byte by byte similarly as in most DPA attacks. The secret key could be expressed as follows: $K_{sec} = \{k_1, k_2, ..., k_{16}\}$ for $0 \leq k_i \leq 255$ where i = 0 to 16 represents each step of the method. This method determines the first byte value k_1 of the secret key in the first step and the second byte value k_2 in the second step and so on. Modelling the power leakage is considered as the basis for launching side-channel attacks, and the effectiveness of these attacks strongly depends on the accuracy of underlying side-channel leakage characterisation. In order to attack cryptographic devices, a power leakage characterisation based on neural network is proposed to take the full intrinsic advantage of a neural network in profiling a non-linear mapping relationship as one basic machine-learning tool, transforms the task of leakage profiling into a neural network study process.

6.4 Analysis Based on Neural Network

The basic measurement setup used to performed this experiment includes an FPGA board with a SPARTAN 3 FPGA, a Tektronix TDS2012 oscilloscope with 1 Gigasample per second to record the power/electromagnet signal traces, a Tektronix CT1 current probe for measuring power consumption of FPGA, an ETS near-field probe set (model 7405) for measuring electromagnetic emission and an ETS broadband amplifier (model 7405-907b) to enhance the quality of the input signal traces (see Figure 3.10). This experiment was performed through a MATLAB R2015a toolbox and a PC configuration of Intel Core i7, 2.8 GHz and 16.00 GB RAM. For more details, refer to 3.5.

Basic Definitions

Some basic expressions used in our experiments are:

- **Training set**: the training set is used to fit the models.
- **Validation set**: the validation set is used to estimate prediction error for model selection.
- **Test set**: the test set is used for assessment of the generalisation error of the final chosen model.
- **Epoch**: An epoch is a measure of the number of times all of the training vectors are used once to update the weights.

Training Functions

The process of training a neural network involves tuning the values of the weights and biases of the network to optimise network performance; therefore, selecting a training function plays a significant role in the efficiency of training and consequently the output of neural network. There are many standard numerical optimisation algorithms that can be used to optimize the performance function, but there are a few key ones that have shown excellent performance for neural network training, some of which are:

- **Levenberg-Marquardt:** It is considered as the fastest back-propagation algorithm in the MATLAB toolbox, and is recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms. For more details, refer to [130].
- **Bayesian Regularization:** It is a good option to produce a network that generalize well. It works by minimizing a combination of squared errors and weights, and then determining the correct combination. For more details, refer to [131].
- **BFGS Quasi-Newton:** It is an alternative to the conjugate gradient methods for fast optimization, although it is a complex method. For more details, refer to [132].
- **Resilient Backpropagation:** It is often faster than training with regular back propagation and doesn't require any free parameter values to be specified, as opposed to back propagation which needs values for the learning rate. However, it's a more complex algorithm to implement than back propagation. For more details, refer to [133].
- **Scaled Conjugate Gradient:** It was designed to avoid the time-consuming line search. For more details, refer to [134].
- **Conjugate Gradient with Powell/Beale Restarts:** It is a network training function that updates weight and bias values according to the conjugate gradient back-propagation with Powell-Beale restarts. It has shown a good performance

on some problems, although the performance on any given problem is difficult to predict. For more details, refer to [135].

- **Fletcher-Powell Conjugate Gradient:** It updates weight and bias values according to conjugate gradient back-propagation with Fletcher-Reeves updates. For more details, refer to [136].
- **Polak-Ribiere Conjugate Gradient:** It updates weight and bias values according to conjugate gradient backpropagation with Polak-Ribiere updates. For more details, refer to [136].
- **One Step Secant:** this method is an attempt to bridge the gap between the conjugate gradient algorithms and the quasi-Newton (BFGS) algorithms. This algorithm requires less storage and computation per epoch than the BFGS algorithm and slightly more storage and computation per epoch than the conjugate gradient algorithms. For more details, refer to [137].
- **Variable Learning Rate Gradient Descent:** It can be used for training any network as long as its weight, net input, and transfer functions have derivative functions. For more details, refer to [138].
- **Gradient Descent with Momentum:** This method allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Momentum allows the network to ignore small features in the error surface. Without momentum a network can get stuck in a shallow local minimum. With momentum a network can slide through such a minimum. See page 129 of. For more details, refer to [136].
- **Gradient Descent:** In this method the weights and biases are updated in the direction of the negative gradient of the performance function. For more details, refer to [136].

It is very difficult to know which training algorithm will be the fastest or the most efficient for a given dataset. It depends on many factors, including the complexity of the dataset, the number of data points in the training set, the number of weights and biases in the network, the error goal; hence, verifying and comparing the influence of various training functions is necessary.

Number of Neurons

Regarding the network topological structure, the number of neurons in input, hidden layer and output should be determined. The input layer must have the same number of neurons as the number of samples or the numbers of chosen interesting points in the case that only interesting points are used because of memory limitation and the time-consuming training process. The output layer classifies the input vector to key value and therefore it must contain enough neurons for all combinations of key values.

Deciding on the number of neurons in the hidden layers is a very important part of overall neural network architecture. Though these layers do not directly interact

with the external environment, they have a tremendous influence on the final output. Both the number of hidden layers and the number of neurons in each of these hidden layers must be carefully considered. Using too few neurons can lead to under-fitting while too many neurons can contribute to computational complexity or over-fitting, in which all training points are well fitted but the fitting curve oscillates wildly between these points. There is no specific method or formula but experiment to determine this number, however the following possible rules are recommended to be considered; see [34], [136].

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

These three rules provide a starting point, ultimately the selection of an architecture for network should be based on experiment and trial and error.

Result Improvement

In order to achieve accurate results, the following approaches can be applied:

- Resetting the initial network weights and biases to new values.
- Increasing the number of hidden neurons.
- Increasing the number of training vectors.
- Increasing the number of input values, if more relevant information is available.
- Applying a different training algorithm.

Applying these approaches is recommended for getting satisfactory results.

6.5 Feed-Forward Back-Propagation (FFBP)

In a feed-forward neural network, the first term "feed forward" describes how this neural network processes and recalls patterns. In a feed-forward neural network, neurons are only connected forward. Each layer of the neural network contains connections to the next layer. In each feed-forward step, an input pattern is applied to the input layer and its effect propagates, layer by layer, through the network until an output is produced. The network's actual output value is then compared to the expected output, and an error signal is computed for each of the output nodes.
The term "back-propagation" describes how this type of neural network is trained. Back-propagation is a form of supervised training that means the network must be

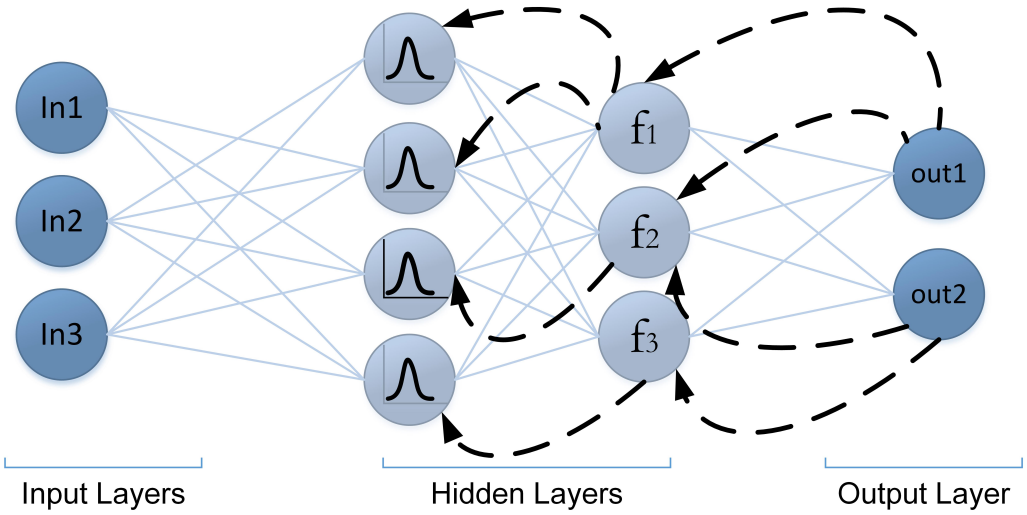


FIGURE 6.2: A feed-forward back-propagation neural network.

provided with both sample inputs and anticipated outputs. The anticipated outputs are compared against the actual outputs for given input. Using the anticipated outputs, the back-propagation training algorithm then takes a calculated error and adjusts the weights of the various layers backwards from the output layer to the input layer. In other words, the output error signals are transmitted backwards from the output layer to each node in the hidden layer that immediately contributed to the output layer. This process is then repeated, layer by layer, until each node in the network has received an error signal that describes its relative contribution to the overall error; see Figure 6.2. The **FFBP** algorithm can be summarised in the following steps (for more details see [139],[140]).

- **Step 1: Determining of neural network topological structure.** In this step, the details of topological structure of neural network should be determined. For example, the number of hidden layer and the number of nodes in input layer, hidden layer and output layer.

- **Step 2: Initialisation**. Weights and variables need an initial value which is usually a random a small real value within [-1,1].

- **Step 3: Forward propagation**. Applying the input to the network and calculating the output. When a specified training pattern is fed to the input layer, the weighted sum of the input to the j^th node in the hidden layer is given by

$$Net_j = \sum W_{i,j} X_j + \Theta_j \tag{6.1}$$

Equation 6.1 is used to calculate the aggregate input to the neuron. The Θ_j term is the weighted value from a bias node that always has an output value of 1. The bias node is considered a "pseudo input" to each neuron in the hidden layer and the output

layer, and is used to overcome the problems associated with situations where the values of an input pattern are zero. If any input pattern has zero values, the neural network could not be trained without a bias node.
In order to decide whether a neuron should fire, the "Net" term is passed onto an appropriate *activation function*. The resulting value from the activation function determines the neuron's output, and becomes the input value for the neurons in the next layer connected to it. Sigmoid equation (see equation 6.2) is known as a typical activation function.

$$O_j = X_k = \frac{1}{1 + e^{-Net_j}} \tag{6.2}$$

- **Step 4: Error calculation.** If the actual activation value of the output node, k, is O_k, and the expected target output for node k is t_k, the difference between the actual output and the expected output is given by:

$$\Delta_k = t_k - O_k \tag{6.3}$$

Then the error signal for node k in the output layer can be calculated as

$$\delta_k = \Delta_k O_k (1 - O_k) \tag{6.4}$$

where the $O_k(1 - O_k)$ term is the derivative of the Sigmoid function.

- **Step 5: Updating weights and biases.** In order to update the weight, $w_{j,k}$, between the output node, k, and the node, j the following formula are applied.

$$\Delta W_j, k = l_r \delta_k X_k \tag{6.5}$$

$$W_j, k = W_j, k + \Delta W_j, k \tag{6.6}$$

where $\Delta W_j, k$ is the change in the weight between nodes j and k, l_r is the learning rate which is a relatively small constant within [0 1] that indicates the relative change in weights.

- **Step 6: Termination**. Finally, Back-propagation is derived by assuming that it is desirable to minimise the error on the output nodes over all the patterns presented to the neural network or the number of iterations exceeds its maximum. The following equation is used to calculate the error function, E, for all patterns.

$$E = \frac{1}{2} \sum (\sum (t_k - O_k)^2) \tag{6.7}$$

Ideally, the error function should have a value of zero when the neural network has been correctly trained. This, however, is numerically unrealistic.

6.5.1 Experimental Results Based on FFBP Analysis

The basic measurement setup used to performed this experiment are an FPGA board with a SPARTAN 3 FPGA, a Tektronix TDS2012 oscilloscope with 1 Gigasample

TABLE 6.1: FFBP network performance comparison. The accuracy comparison of various training algorithms with a proper number of hidden layers and based on timing complexity and memory consumption

Training Algorithms	**H.L** Number	**MSE** Min	Avg	Range	**Time** (Sec)	**Mem** (GB)
Levenberg-Marquardt	[23 27]	0.059	0.061	[0.05 0.08]	16.23	1.155
Bayesian Regularisation	[26 31]	0.017	0.018	[0.01 0.04]	6584	0.019
BFGS quasi-Newton	[30 33]	0.084	0.093	[0.08 0.15]	335	0.015
Resilient Back-propagation	[20 24]	0.072	0.097	[0.07 0.12]	0.42	1.694
Scaled conjugate gradient	[21 23]	0.074	0.083	[0.07 0.09]	0.51	1.684
C.G.Powell-Beale	[85 92]	0.049	0.072	[0.04 0.11]	1.11	1.771
C.G.Fletcher-Reeves	[55 59]	0.057	0.084	[0.05 0.16]	0.62	1.033
C.G.Polka-Ribiere	[49 51]	0.051	0.085	[0.05 0.15]	1.49	1.065
One-Step Secant	[87 94]	0.153	0.171	[0.15 0.18]	11.05	1.141
Gradient Descent	[89 97]	0.162	0.177	[0.16 0.18]	3.88	1.662
G.D.Adaptive Learning	[9 15]	0.079	0.084	[0.07 0.11]	0.42	1.698
G.D.Momentum	[41 47]	0.171	0.178	[0.17 0.18]	6.89	1.116

per second to record the power/electromagnet signal traces, a Tektronix CT1 current probe for measuring power consumption of FPGA, an ETS near-field probe set (model 7405) for measuring electromagnetic emission and an ETS broadband amplifier (model 7405-907b) to enhance the quality of the input signal traces (see Figure 3.10). This experiment was performed through a MATLAB R2015a toolbox and a PC configuration of Intel Core i7, 2.8 GHz and 16.00 GB RAM. For more details, refer to 3.5. Due to the fact that back-propagation performance depends on various factors. Assigning a proper training function and number of hidden layers plays a significant role in the efficiency of the neural network and must be carefully determined for a given dataset. Therefore, an experimental investigation was conducted to explore the efficiency of various training functions with different numbers of hidden layers. a FFBP network has been trained with twelve different training algorithms, and for each algorithm various numbers of hidden layers ranging from 1 to 110 have been tested. This range is selected to avoid over-fitting, inaccuracy and computational complexity.

Table 6.1 illustrates a performance comparison of various training algorithms in FFBP network analysis. This performance is with a proper range of hidden-layers number and based on mean-squared-error (MSE) which shows the accuracy of the applied approach, time consumption of algorithm which shows the timing complexity and memory consumption showing the required memory for this algorithm. From this table, *Bayesian Regularisation* with [26 31] hidden layers can be considered as the most accurate training function with the minimum MSE of (0.017) and MSE range of [0.01 0.04], while other MSE ranged between $[0.04, 0.18]$. On the other hand, the maximum MSE belongs to *Gradient Descent* and *Gradient Descent with Momentum* with 0.162 and 0.171 respectively.

In terms of computational complexity, as shown in this table, *Bayesian Regularisation* is significantly the slowest algorithm with the processing time of 6584 seconds which is significantly greater than other algorithm. Afterward, *BFGS quasi-Newton* and *Levenberg-Marquardt* with 335 seconds and 16.23 seconds respectively. All other algorithms are quite fast and their computational time consumption are not more than 3 seconds.

Memory consumption usually increases exponentially as the number of hidden layer increase. Judging from the memory consumption information in this table, *Bayesian Regularisation* and *BFGS quasi-Newton* consume the least amount of memory by 0.019 and 0.015 GB respectively, while all other training functions needs almost the same size of memory around 1.5 GB.

Overall, regarding the accuracy, time and memory consumption, *Levenberg-Marquardt* can be considered as the most efficient FFBP training function. Because, although *Bayesian Regularisation* has the best accuracy, it is significantly the slowest algorithm. In addition, however the minimum MSE of *C.G.Powell-Beale* and *C.G.Polka-Ribiere* are smaller than that of *Levenberg-Marquardt*, the accuracy of *Levenberg-Marquardt* is more stable and reliable since its range of MSE variation and average are smaller (the MSE variation for *Levenberg-Marquardt* is $0.08 - 0.05 = 0.03$ and the range of MSE variation for *C.G.Powell-Beale* is $0.11 - 0.04 = 0.07$).

Figure 6.3 illustrates a particular comparison of *Bayesian Regularisation* and *Levenberg-Marquardt* performance with the number of hidden layer from 1 to 40 by which the

processing time of *Bayesian Regularisation* increased dramatically to 6000 seconds. In this Figure, memory-consumption, time-consumption and mean squared error (MSE) of FFBP analysis with respect to the number of hidden layers is compared. The perfor-

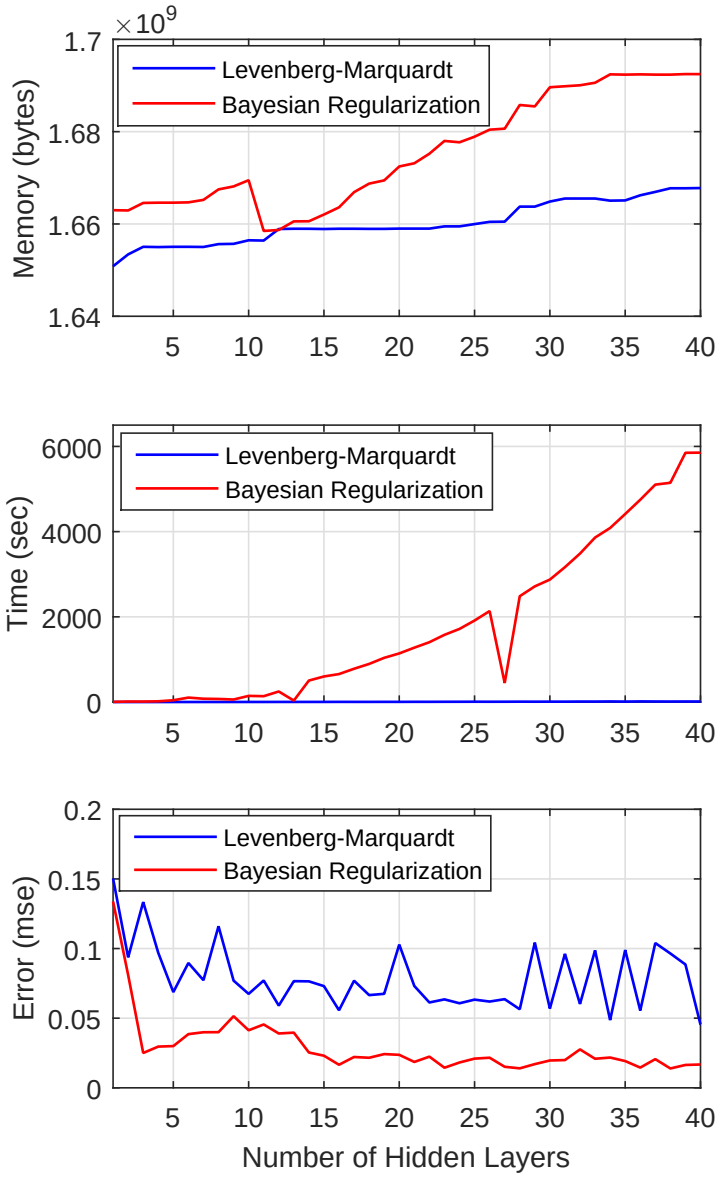


FIGURE 6.3: The performance comparison of *Bayesian Regularisation* and *Levenberg-Marquardt* algorithm based on memory consumption, Time consumption and Mean Squared Error (MSE).

mance of *Levenberg-Marquardt* as the most efficient training algorithm with the number of hidden layers in range of 1 to 110 is presented in Figure 6.4. As can be seen, the MSE

shows a fluctuation with a decrease from 0.1 to 0.04 as the number of hidden layers increases to 30, and after that rises to 0.09 and then remains stable around 0.06. From this Figure, as the number of hidden layers increases from 1 to 200, the computational timing exponentially increases from 1 seconds to beyond 500 seconds. Moreover, the consumption of memory surprisingly declined by 0.04 GB and dropped to a minimum of 1.67 GB with 90 hidden layers. Generally, it can be inferred that the more hidden layer are used, the more time is required for processing the network and the less error can be expected. Regarding the graph, FFBP architectures with a number of hidden layers between 20 and 30 can be considered as the most efficient networks in terms of accuracy and time, however their required memory is not the minimum.

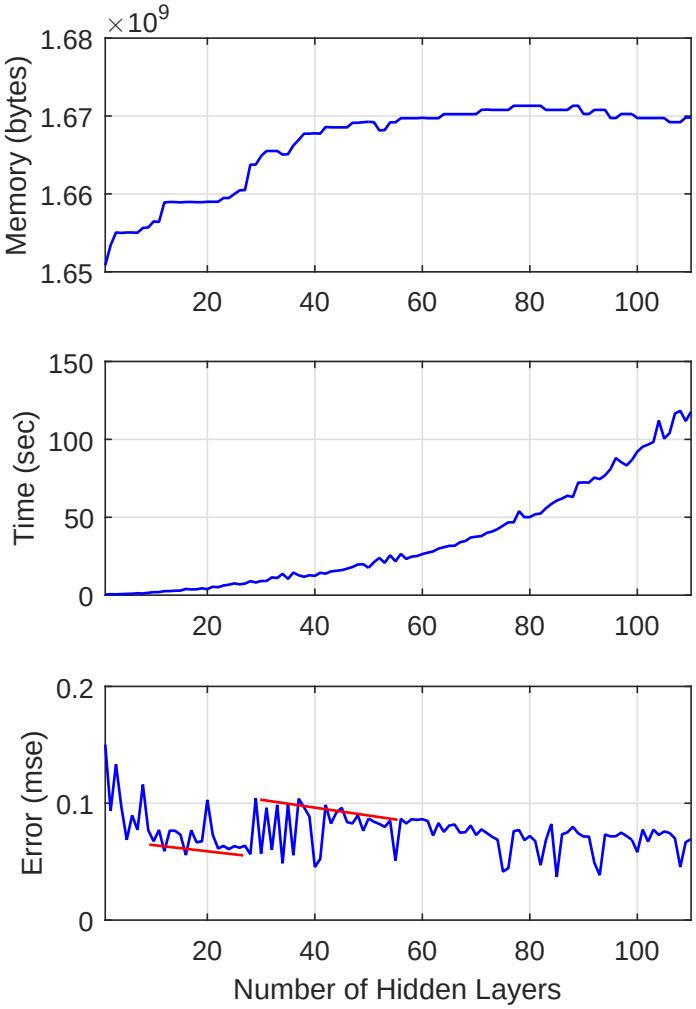


FIGURE 6.4: The performance of *Levenberg-Marquardt* (as the most efficient training algorithm) based on the memory consumption, Time-consumption and Mean Squared Error (MSE).

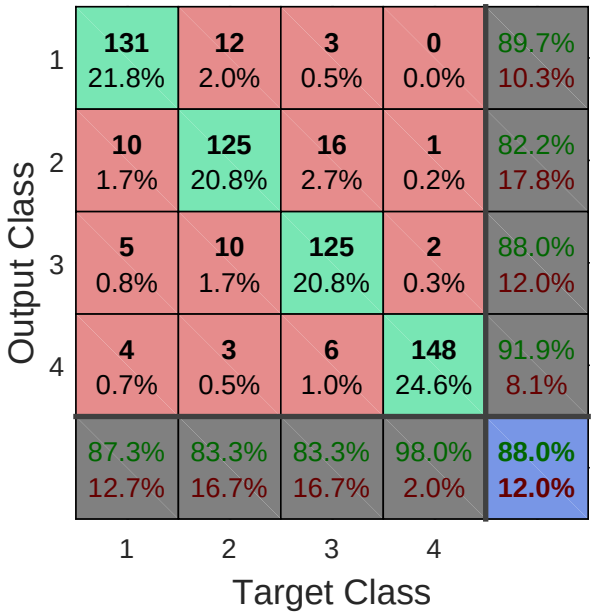


FIGURE 6.5: Confusion matrix. The diagonal cells (green): the number of correctly classified cases, The off-diagonal cells (red): the number of misclassified cases, The blue cell: the overall percentage.

In order to verify the efficiency of our FFBP based on the *Levenberg-Marquardt* training function, a confusion matrix is provided which is a table that is used to describe the performance of the classifier model on our dataset that shows four different key-bit values of ECC as four classes. Each row of the matrix represents the instances in a classified class (Output), while each column represents the instances in an actual class (Target). Figure 6.5 shows the confusion matrices for the training, testing, and validation sets. The diagonal cells show the number of cases that the key value were correctly classified, and the off-diagonal cells show the misclassified cases. The blue cell in the bottom right shows the total percentage of correctly classified cases (in green) and the total percentage of misclassified cases (in red). As can be seen, the results shows a good classification because of the higher numbers of correct responses in the diagonal squares (125, 131 and 148) compared to the low numbers of incorrect responses in the off-diagonal squares (ranging [1 16]). The lower right blue square illustrates the overall accuracies of 88% correctly classified and 12% misclassified. In Figure 6.6 the performance of our FFBP-based classification in terms of the mean-square error of training, validation and test sets is indicated. This figure gives us information about the performance and status of our FFBP training, validating and testing process and how the network can be improved. For example, if the training performance is poor, then the number of neurons should be increased. If the performance on the training set is good, but the test-set performance is significantly worse, which could indicate over-fitting, then reducing the number of neurons or hidden layers can improve the results. Considering this figure, the test-set error and validation set error have similar characteristics and the best performance occurred at the first epoch. In addition, the

final mean-square error is small and no significant over-fitting has occurred by the first iteration (where the best validation performance occurs); therefore the result is reasonable.

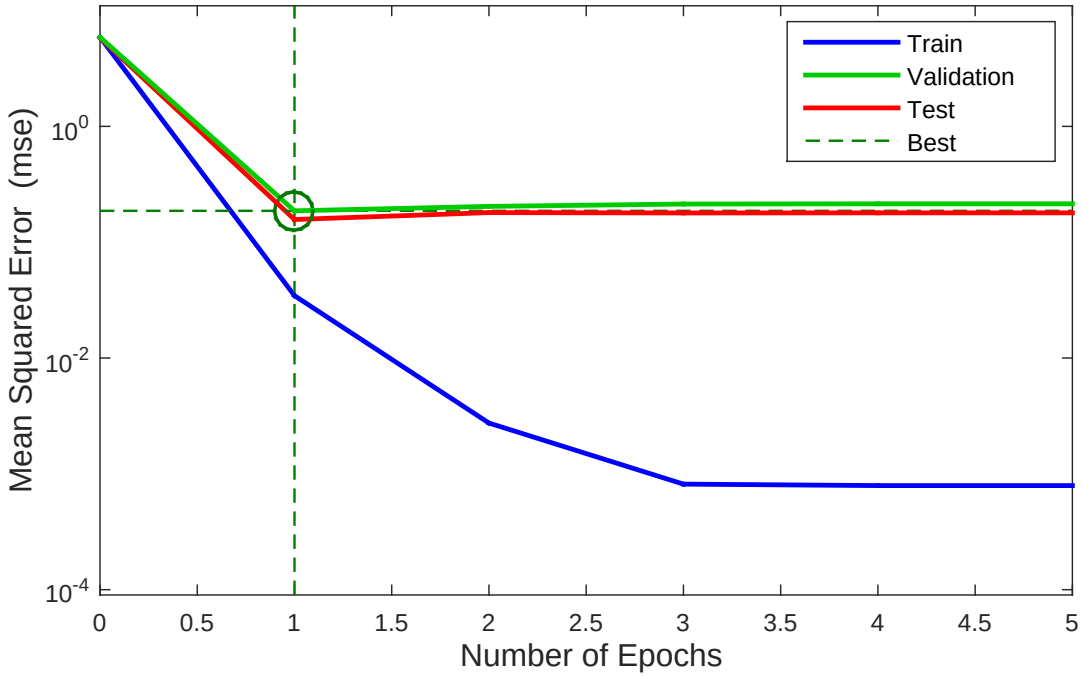


FIGURE 6.6: The performance of our FFBP-based classification in terms of the mean-square error of training, validation and test sets. Best validation performance is 0.18663 at epoch 1.

Figure 6.7 shows the error histogram to obtain additional verification of our method performance. Due to variability in measurement or some unavoidable experimental and noise, some recorded data are distant from others and their fit is significantly worse than the majority of data and are considered as outliers. It is reasonable to check the outliers to determine if the data is bad, or if those data points are different from the rest of the dataset. If the outliers are valid data points, but are unlike the rest of the data, then the network is extrapolating for these points; therefore, more data should be collected that looks like the outliers points, and the network should be retrained. Figure 6.7 gives an indication of outliers. It can be seen that while most errors fall between -0.6 and 0.8, there are two training points with an error of -1 and -1.2 and two testing points with errors of 1.0 and 0.96. These outliers correspond to analysis based on only a few number of instances. Expectedly, as the number of instances increases, the errors decrease.

In order to check the quality of classifiers, Receiver Operating Characteristic (ROC) curves are provided, see Figure 6.8. For each class of a classifier, ROC applies threshold

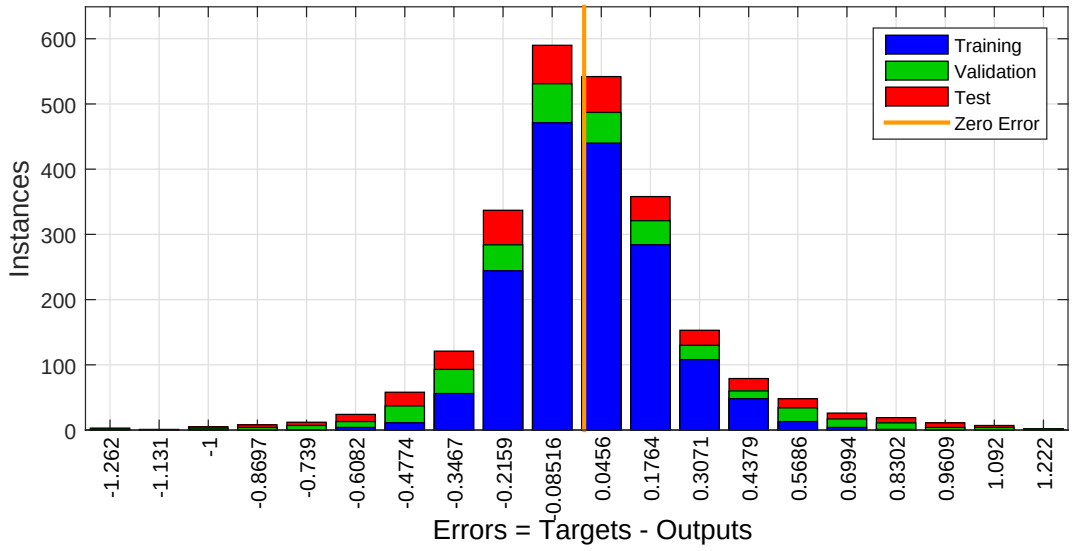


FIGURE 6.7: Error histogram as an indication of outliers.

values across the interval [0,1] to outputs. For each threshold, two values are calculated, the True Positive Ratio (the number of outputs greater or equal to the threshold, divided by the number of one targets), and the False Positive Ratio (the number of outputs less than the threshold, divided by the number of zero targets). In Figure 6.8, the colored lines in each axis represent the ROC curves which is a plot of the true positive rate versus the false positive rate as the threshold is varied. A perfect test would show points in the upper-left corner. As illustrated in Figure 6.8, the best classification was for key-bit 4, then classification for key-bit 1, 3 and 2 respectively.

6.5.2 Summary

This Section has investigated pattern recognition and classification of side-channel information based on feed-forward back-propagation (FFBP). Regarding the result of this section, we can infer that FFBP architectures with between 20 and 30 hidden layers can be considered as the most efficient architectures in terms of accuracy and time complexity, however the required memory is not the minimum. Moreover, *Levenberg-Marquardt* can be considered as the most efficient FFBP training function. Because, although *Bayesian Regularisation* has the best accuracy, it is significantly the slowest algorithm. In addition, our results indicate an overall accuracy of 88% correctly classified and 12% misclassified, and no significant over-fitting.

6.6 Probabilistic Neural Network (PNN)

A probabilistic neural network (PNN) is a feed-forward neural network, which provides a general solution to pattern classification problems by following an approach developed

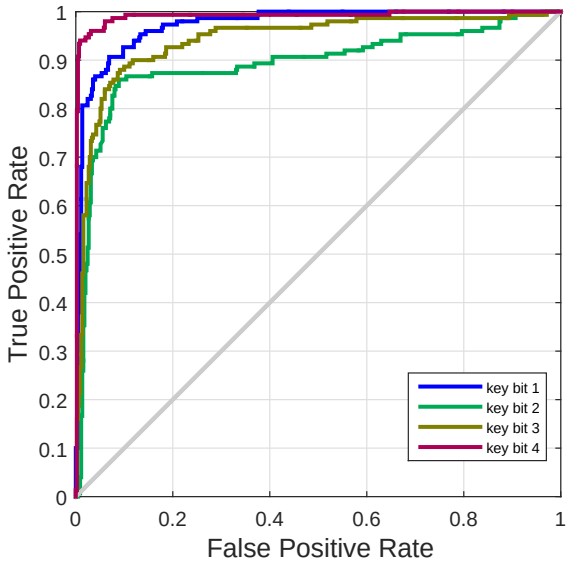


FIGURE 6.8: Receiver Operating Characteristic (ROC) curves; the best classification was for key bits 4, then for key bits 1, 3 and 2.

in statistics, called Bayesian classifiers; see [141]. PNN can compute non-linear decision boundaries by replacing the sigmoid activation function often used in neural networks with an exponential function. One outstanding issue associated with the PNN is the determination of the network structure. This includes determining the network size, the pattern layer neurons and an appropriate smoothing parameter. Some algorithms for pattern layer neurons selection have been proposed ([142, 143]).

Advantages: PNN is adopted because of several advantages; its training speed is many times greater than for a FFBP network. PNN can approach a Bayes optimal result under certain easily met conditions. Additionally, it is robust to noise samples, which is a important factor for side-channel information analysis. The most important advantage of PNN is that training is easy and instantaneous. The weights of neurones are not trained but assigned. The existing weights of neurones will never be alternated but only new vectors are inserted into weight matrices when training, so it can be used in real time. Since the training and running procedure can be implemented by matrix manipulation, PNN is very fast.

Disadvantages: due to each pattern layer Gaussian component density being derived from one training vector, the PNN is limited to applications involving relatively small datasets. Large datasets would lead to large network architectures, which would have an adverse impact on computational complexity. In addition, this could saturate the feature space with overlapping Gaussian functions that would increase the rate of misclassification.

6.6.1 PNN Algorithm

The PNN operations are organised into a multi-layered feed-forward network with the four layers explained below (see Figure 6.9):

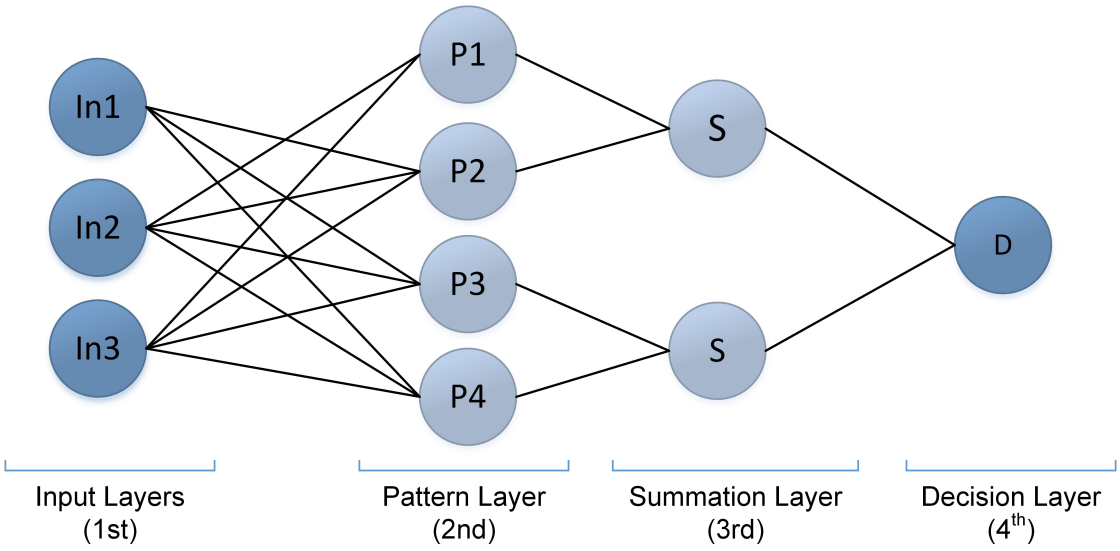


FIGURE 6.9: PNN architecture

1. *Input layer:* Each neuron in the input layer represents a predictor variable. Generally, $N-1$ neurons are used when there are N patterns. Then the input neurons feed values to each neuron in the hidden layer.

2. *Pattern layer:* This layer contains one neuron for each case in the training dataset. It stores the values of the predictor variables for the case along with the target value. A hidden neuron computes the distance of the test case from the neuron's centre point and then applies the RBF kernel function using the sigma values. On receiving a pattern from the input layer, the neuron x_{ij} of the pattern layer computes its output

$$\Phi_{ij} = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left[-\frac{(x - xij)^T(x - x_{ij})}{2\sigma^2}\right] \tag{6.8}$$

   where d denotes the dimension of the pattern vector x,σ is the smoothing parameter and x_{ij} is the neuron vector.

3. *Summation layer:* For PNN networks there is one pattern neuron for each category of the target variable. The actual target category of each training case is stored with each hidden neuron; the weighted value coming out of a hidden neuron is fed only to the pattern neuron that corresponds to the hidden neurons category. The pattern neurons add the values for the class they represent. In more details, the summation layer neurons compute the maximum likelihood of

pattern x being classified into class C_i by summarizing and averaging the output of all neurons that belong to the same class

$$p_i(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{N_i} \sum_{j=1}^{N_i} \exp\left[-\frac{(x - xij)^T(x - x_{ij})}{2\sigma^2}\right] \tag{6.9}$$

where N_i denotes the total number of samples in class C_i.

4. *Output layer:* The output layer compares the weighted votes for each target category accumulated in the pattern layer and uses the largest vote to predict the target category. If the *apriori* probabilities for each class are the same, and the losses associated with making an incorrect decision for each class are the same, the decision layer unit classifies the pattern x in accordance with the Bayess decision rule based on the output of all the summation layer neurons

$$\hat{C}(x) = \arg\max\left\{p_i(x)\right\}, i = 1, 2, ..., m \tag{6.10}$$

where $\hat{C}(x)$ denotes the estimated class of the pattern x and m is the total number of classes in the training samples.

6.6.2 Experimental results based on PNN

The basic measurement setup used to performed this experiment are an FPGA board with a SPARTAN 3 FPGA, a Tektronix TDS2012 oscilloscope with 1 Gigasample per second to record the power/electromagnet signal traces, a Tektronix CT1 current probe for measuring power consumption of FPGA, an ETS near-field probe set (model 7405) for measuring electromagnetic emission and an ETS broadband amplifier (model 7405-907b) to enhance the quality of the input signal traces. This experiment was performed through a MATLAB R2015a toolbox and a PC configuration of Intel Core i7, 2.8 GHz and 16.00 GB RAM. For more details, refer to 3.5.

There have been several studies in the literature reporting how the activation function in PNN depends on the chosen *Bayesian* classification parameters. Due to the impact of spread parameter selection on classification learning performance and generalisation, the magnitude of the spread value should be selected with care. Therefore, in this section, the selection of the optimal spread parameter is verified. In Figure 6.10 there is a clear trend of memory consumption, time consumption and mean squared error (MSE) of PNN analysis in respect with the values of the spread parameters ranging from 0.001 to 90. As can be seen, with spread parameters between 17 and 26 the MSE shows fluctuation and minimises at 0.37 which considered as the best accuracy. From this figure, it can be inferred that PNN is a fast approach, since the computational timing is mostly stable at about 0.2 seconds for different values of the spread parameter. In terms of memory consumption, as the value of the spread parameter increased from 0.001 to 20, the consumption of memory declined from 1.66 GB to 1.55 GB and afterwards remained stable. Judging from this figure, PNN architectures with

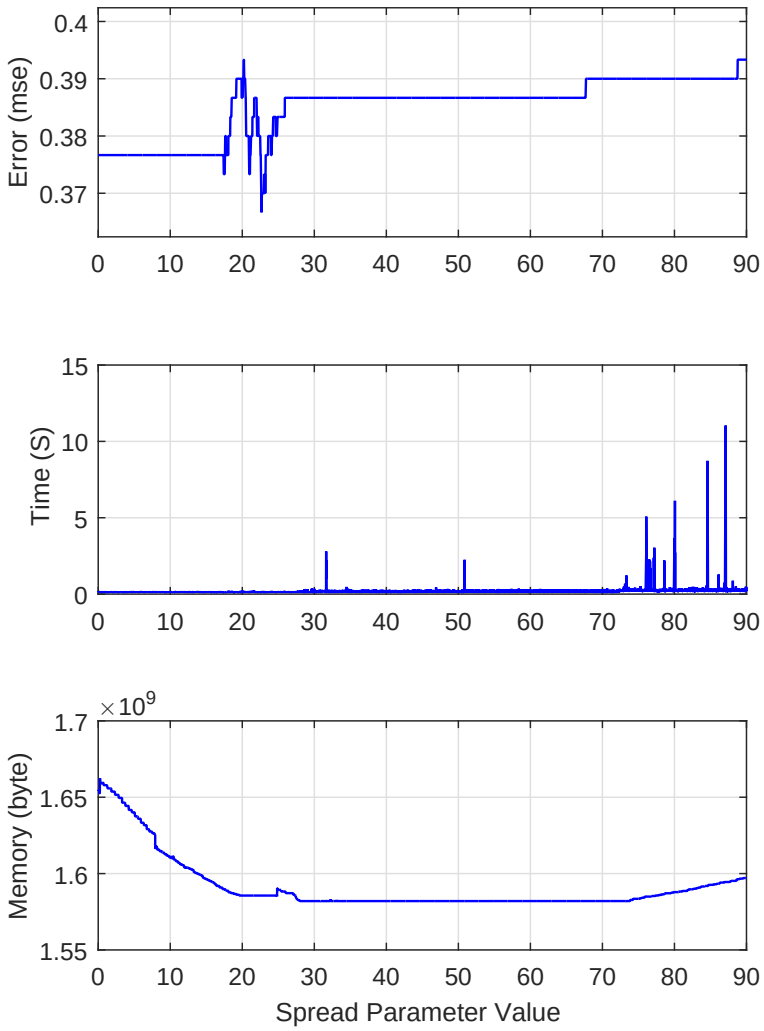


FIGURE 6.10: Memory consumption, Time consumption and Mean-Squared Error (MSE) of PNN analysis as functions of the spread-parameter value.

spread parameter values between 22.5 and 23.2 can be considered as the most efficient networks in terms of accuracy, time and memory consumption.

6.6.3 Summary

In this section pattern recognition and classification of side-channel information based on a probabilistic neural network (PNN) is investigated. PNN is adopted because of several advantages. Some of the most important advantages of PNN are

- its training speed is many times greater than for a FFBP network.

- robustness to noise samples.
- easy and instantaneous training.
- efficient for real time analysis.

Regarding the experimental results, it can be inferred that the best expected accuracy of PNN is with the spread parameters between 17 and 26, where the mean squared error (MSE) minimises at 0.37. Overall, considering the computational timing and memory consumption, the most efficient PNN architectures is with spread parameter values between 22.5 and 23.2.

6.7 Cascade and Forward Back-Propagation Neural Network (CFBP)

A Cascade Forward Back Propagation network (CFBP) is similar to a Feed-Forward Back-Propagation network. The main indication of CFBP architecture is to build up the cascade architecture by adding new neurons together with their connections to all the inputs as well as to the previous hidden neurons. This configuration is not changed at the following layers [144]. The other special feature of CFBP is to learn only the newly created neuron by fitting its weights so that to minimise the residual error of the network. The new neurons are added to the network while its performance increases. So, the common cascade-correlation technique assumes that all m variables $x_1, ..., x_m$ characterising the training data are relevant to the classification problem

6.7.1 The Dynamic Network Architecture of CFBP

Figure 6.11 shows the architecture of a **CFBP** neural network with one input layer of three neurons, one hidden layer of two neurons and an output layer. At the beginning, a cascade network with m inputs and one output neuron starts to learn without hidden neurons. The output neuron is connected to every input by weights $w_1, ..., w_m$ adjustable during learning. The output y of neurons in the network is given by the standard sigmoid function f as follows

$$y = f(x; w) = 1/(1 + exp(-w_0 - \sum_{i}^{m} w_i x_i)) \tag{6.11}$$

where $x = (x_1, ..., x_m)$ is a $m \times 1$ input vector, $w = (w_1, ..., w_m)$ is a $m \times 1$ weight vector and w_0 is the bias term which is hereinafter omitted.

Then the new neurons are added to the network one-by-one. Each new neuron is connected to all m inputs as well as to all the previous hidden neurons. As shown in Figure 6.11 by p1 and p2 each time only the output neuron is trained.

For training, the learning algorithm grows a network of near optimal complexity which can generalise well; for more details see [145]. For updating network weights,

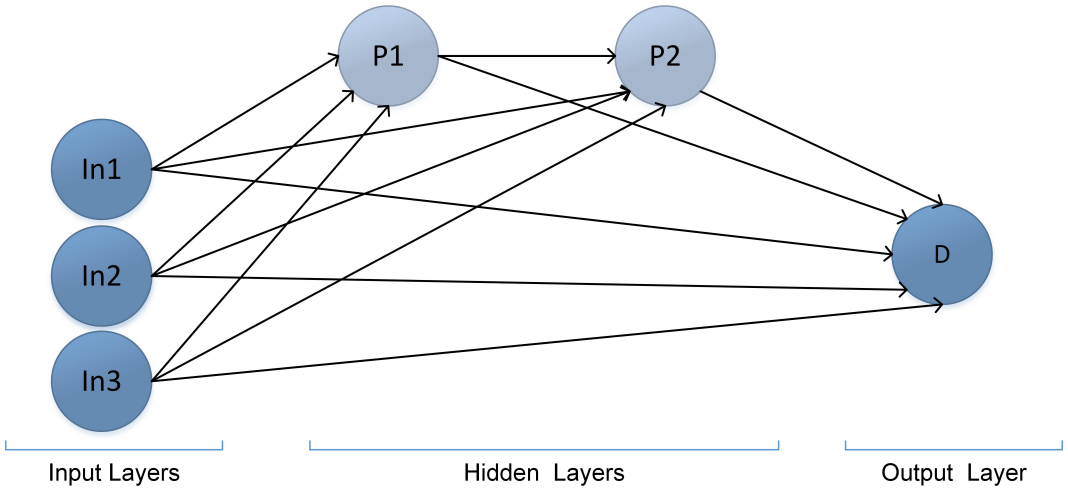


FIGURE 6.11: Architecture of Cascade Forward Neural Network.

two training algorithms including *Levenberg-Marquardt* and *Bayesian Regulation* algorithms can be used.

Advantages: The advantages of the cascade neural networks are well known. First, no structure of the networks is predefined, that is, the network is automatically built up from the training data. Second, By controlling the connectivity of neurons, a smaller number of neurons can be applied which can lead to faster learning. For example, a three-layer network has connections from first layer to second layer, second layer to third layer, and first layer to third layer. The additional connections might improve the speed at which the network learns the desired relationship; for more details see [146].

Disadvantages: a disadvantage is that the cascade networks can be over-fitted in the presence of noisy features.

6.7.2 Experimental Results Based on CFBP

The basic measurement setup used to performed this experiment are an FPGA board with a SPARTAN 3 FPGA, a Tektronix TDS2012 oscilloscope with 1 Gigasample per second to record the power/electromagnet signal traces, a Tektronix CT1 current probe for measuring power consumption of FPGA, an ETS near-field probe set (model 7405) for measuring electromagnetic emission and an ETS broadband amplifier (model 7405-907b) to enhance the quality of the input signal traces (see Figure 3.10). This experiment was performed through a MATLAB R2015a toolbox and a PC configuration of Intel Core i7, 2.8 GHz and 16.00 GB RAM. For more details, refer to 3.5.

For our experiment, based on a CFBP network, twelve different training algorithms are verified. For each algorithm, various numbers of hidden layers ranging from 1 to 110 have been tested. This range is selected to avoid over-fitting, inaccuracy and computational complexity.

Table 6.2 illustrates a performance comparison of various training algorithms in CFBP network analysis. This performance is with a proper range of hidden-layers number and

based on mean-squared-error (MSE) which shows the accuracy of the applied approach, time consumption of algorithm which shows the timing complexity and memory consumption showing the required memory for this algorithm. From this table, *Bayesian Regularisation* with [26 32] hidden layers can be considered as the most accurate training function with the minimum MSE of (0.014) and MSE range of [0.01 0.04], while other MSE ranged between $[0.05, 0.27]$. On the other hand, the maximum MSE belongs to *Gradient Descent* and *Gradient Descent with Momentum* with 0.177 and 0.271 respectively.

From this table, *Bayesian Regularisation* with 27 hidden layers can be considered as the most accurate training function with 0.0140 error, while other errors ranged between $[0.05, 0.27]$. On the other hand, the maximum error belongs to *Gradient Descent* and *Gradient Descent with Momentum* with 0.177 and 0.271 respectively.
In terms of computational complexity, as shown in this table, *Bayesian Regularisation* is significantly the slowest algorithm, with a processing time of 8600 seconds. Afterward, *BFGS quasi-Newton* and *Levenberg-Marquardt*, with 154 and 46.23 seconds respectively. All other algorithms are quite fast and their computational time consumption are not considerable (less than 6 seconds).
Memory consumption usually increases potentially as the number of hidden layers increases. Judging from the memory consumption information in this table, *Bayesian Regularisation* and *BFGS quasi-Newton* consume the least amount of memory with 0.02 GB, while all other training functions use almost the same size of memory, around 1.5 GB.
Overall, regarding the accuracy, time and memory consumption, *Levenberg-Marquardt* can be considered as the most efficient FFBP training function. Because, although *Bayesian Regularisation* has the best accuracy, it is significantly the slowest algorithm. In addition, however the minimum MSE of *Resilient Back-propagation* and *C.G.Polka-Ribiere* are smaller than that of *Levenberg-Marquardt*, the accuracy of *Levenberg-Marquardt* is more stable and reliable since its range of MSE variation and average are smaller (the MSE variation for *Levenberg-Marquardt* is $0.08 - 0.06 = 0.02$ and the range of MSE variation for *Resilient Back-propagation* is $0.16 - 0.05 = 0.11$).

Figure 6.12 presents the performance of *Levenberg-Marquardt* as the most efficient training algorithm with the number of hidden layers in the range of 1 to 110. As can be seen, the MSE shows a fluctuation, with a decrease from 0.08 to 0.05 as the number of hidden layers increases to 35, and after that remains stable around 0.05. From this figure, as the number of hidden layers increases from 1 to 110, the computational timing exponentially increases from 1 second to beyond 400 seconds. Moreover, the consumption of memory remains stable at around 1.2 GB, except for the architecture with 40 hidden layers that needs the maximum memory, of 1.5 GB. Generally, from Figure 6.12, it can be inferred that the more hidden layers are used in the CFBP network, the more time is required for processing the network and the less error can be expected. Regarding the graph, FFBP architectures with a number of hidden layers between 20 and 30 can be considered as the most efficient networks in terms of accuracy and time, however their required memory is not the minimum. In order to verify the

TABLE 6.2: CFBP network performance comparison. The accuracy comparison of various training algorithms with a proper number of hidden layers and based on timing complexity and memory consumption.

Training Algorithms	**H.L** Number	**MSE** Min	Avg	Range	**Time** (Sec)	**Mem** (GB)
Levenberg-Marquardt	[33 36]	0.067	0.069	[0.06 0.08]	46.23	1.184
Bayesian Regularisation	[26 32]	0.014	0.019	[0.01 0.04]	8600	0.020
BFGS quasi-Newton	[40 47]	0.092	0.098	[0.09 0.14]	154	0.020
Resilient Back-propagation	[41 47]	0.050	0.084	[0.05 0.16]	1.01	1.213
Scaled conjugate gradient	[91 97]	0.051	0.10	[0.05 0.17]	0.67	1.199
C.G.Powell-Beale	[86 97]	0.052	0.082	[0.05 0.13]	0.71	1.184
C.G.Fletcher-Reeves	[65 69]	0.052	0.094	[0.05 0.16]	0.92	1.173
C.G.Polka-Ribiere	[74 80]	0.054	0.085	[0.05 0.15]	0.70	1.165
One-Step Secant	[82 90]	0.062	0.08	[0.06 0.98]	1.21	1.169
Gradient Descent	[84 91]	0.177	0.187	[0.17 0.19]	5.69	1.170
G.D.Adaptive Learning	[18 23]	0.078	0.085	[0.07 0.11]	0.82	1.170
G.D.Momentum	[68 71]	0.271	0.275	[0.27 0.29]	4.96	1.170

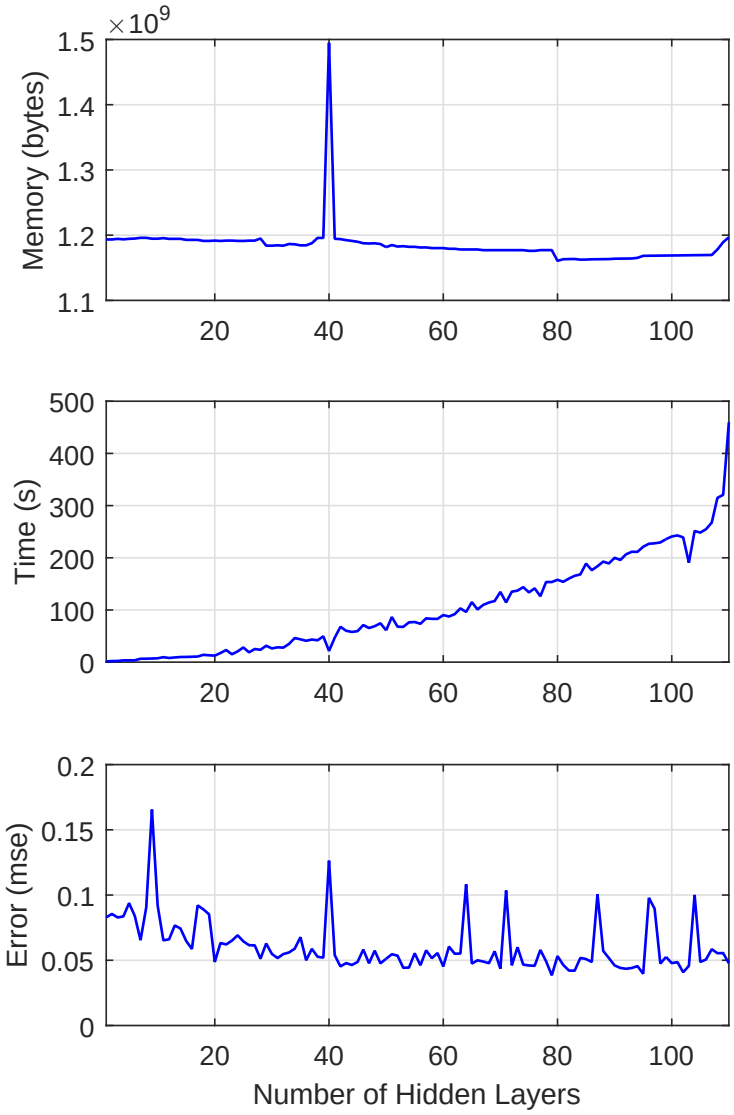


FIGURE 6.12: The performance of *Levenberg-Marquardt* (as the most efficient training algorithm for CFBP network) based on memory consumption, Time consumption and Mean Squared Error (MSE).

efficiency of a CfBP network based on the *Levenberg-Marquardt* training function, a confusion matrix is provided in which the number of times that a particular key value was correctly classified or misclassified are presented. Figure 6.13 shows the confusion matrices for training, testing, and validation sets. The diagonal cells show the number of times that the key values were correctly classified, and the off-diagonal cells show the misclassified cases. The blue cell at the bottom right shows the total percentage of correctly classified cases (in green) and the total percentage of misclassified cases (in red). As can be seen, the results shows a good classification because of the higher

numbers of correct responses in the diagonal squares (103, 110 and 139) compared to the low numbers of incorrect responses in the off-diagonal squares (ranging [3 32]). The lower-right blue square illustrates the overall accuracies of 75.7% correctly classified and 24.3% misclassified. In Figure 6.14 the performance of our CF-based classification in

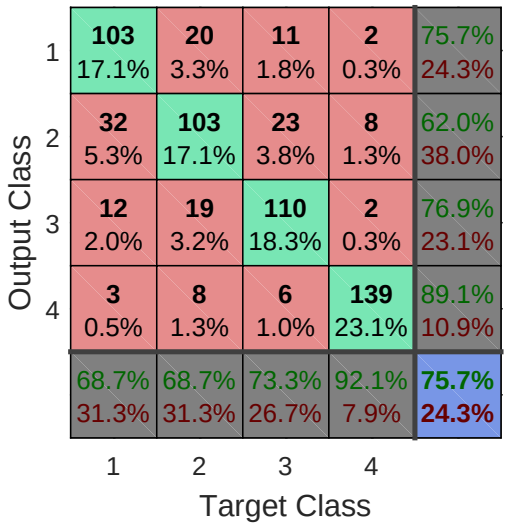


FIGURE 6.13: CFBP network Confusion matrix. The diagonal cells (green): the number of correctly classified cases, The off-diagonal cells (red): the number of misclassified cases, The blue cell: the total percentages.

terms of the mean-square error of training, validation and test sets is indicated. This figure gives us information about the performance and status of our CFBP network training, validating and testing process and how the network can be improved. For example, if the training performance is poor, then the number of neurons should be increased. If the performance on the training set is good, but the test-set performance is significantly worse, which could indicate over-fitting, then reducing the number of neurons or hidden layers can improve the results. Considering this figure, the test-set error and validation-set errors have similar characteristics, and the best performance occurred at the first epoch. In addition, the final mean-square error is small and no significant over-fitting has occurred by the first iteration (where the best validation performance occurs); therefore the result is reasonable.

Figure 6.15 shows the error histogram to obtain additional verification of our method's performance. Due to variability in measurement or some unavoidable experimental and noise, some recorded data are distant from others and their fit is significantly worse than the majority of data, and they are considered as outliers. It is reasonable to check the outliers to determine if the data is bad, or if those data points are different from the rest of the dataset. If the outliers are valid data points, but are unlike the rest of the data, then the network is extrapolating for these points; therefore,

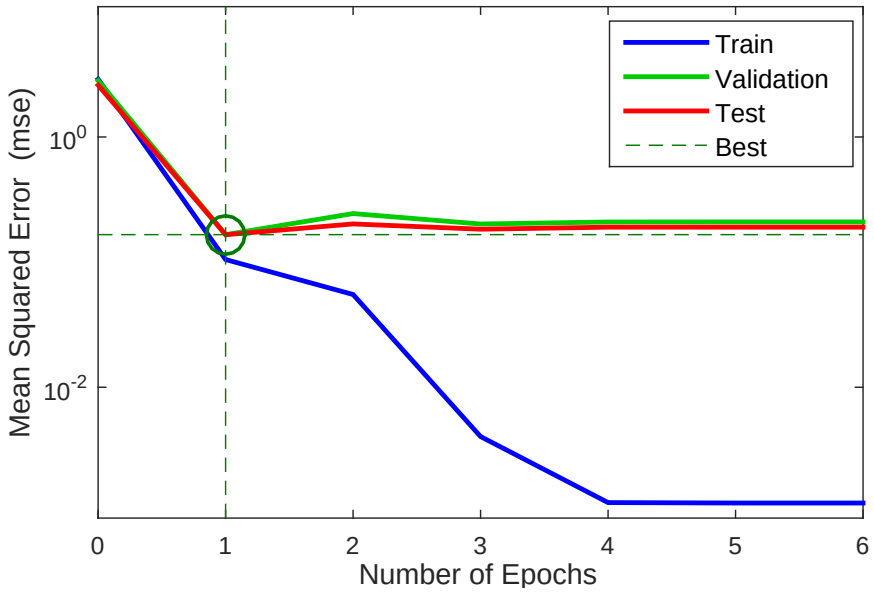


FIGURE 6.14: The performance of our CFBP-based classification in terms of the mean-square error of training, validation and test sets. Best validation performance is 0.16579 at epoch 1.

more data should be collected that looks like the outlier points, and the network should be retrained. Figure 6.15 gives an indication of outliers. It can be seen that most errors fall between -0.5 and 0.5 and the outliers correspond to analysis based on only a few instances. Expectedly, as the number of instances increase, the errors decrease. In oder

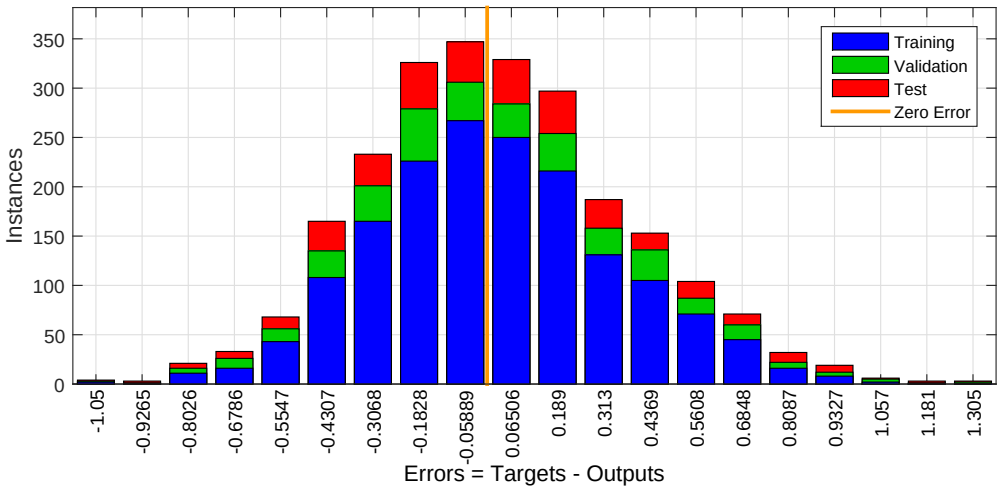


FIGURE 6.15: CFBP network Error histogram as an indication of outliers.

to check the quality of classifiers, Receiver Operating Characteristic (ROC) curves are

provided, see Figure 6.16. For each class of a classifier (class of key-bit 1, key-bit 2, key-bit 3 and key-bit 4), ROC applies threshold values across the interval [0,1] to the outputs. For each threshold, two values are calculated, the True Positive Ratio (the number of outputs greater than or equal to the threshold, divided by the number of one targets), and the False Positive Ratio (the number of outputs less than the threshold, divided by the number of zero targets). In Figure 6.16, the coloured lines represent the ROC curves, which are plots of the true positive rate versus the false positive rate as the threshold is varied. A perfect test would show points in the upper-left corner. As illustrated in Figure 6.16, because of the ratio of True-Positive-Rate to False-Positive-Rate the best classification was for key-bit 4, then classification for key-bits 1, 3 and 2.

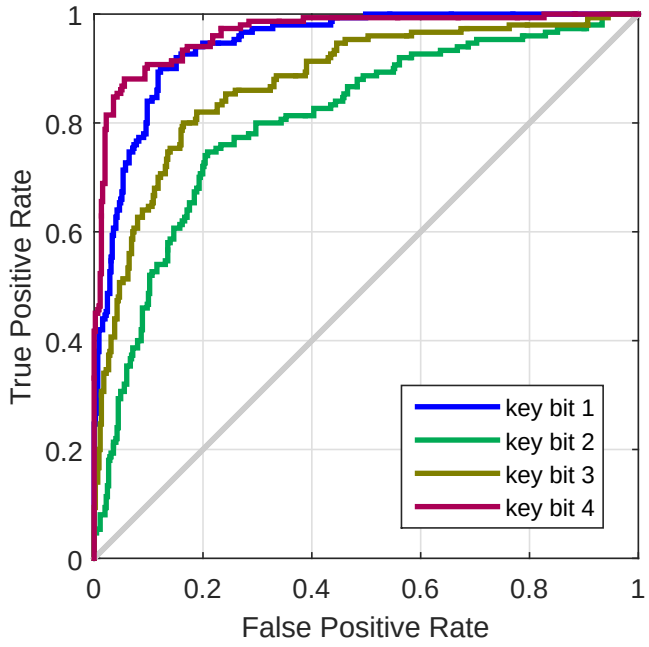


FIGURE 6.16: Receiver Operating Characteristic (ROC) curves based on CFBP classification; the best classification was for key-bit 4, then key bits 1, 3 and 2.

6.7.3 Summary

This section has investigated pattern recognition and classification of side-channel information based on cascade and feed-forward back propagation neural network (CFBP). This, is adopted because of several special features, some of which are

- The network is automatically built up from the training data.
- Having faster learning process.
- Controlling the number and connectivity of neurons.

Judging from the result of this section, it can be inferred that *Levenberg-Marquardt* is the most efficient CFBP training function for stability, accuracy, time and memory consumption. Moreover, the provided confusion-matrix and error-histogram indicate an overall accuracy of 75.7% correctly classified and 24.3% misclassified and no significant over-fitting. Hence, side channel data characterisation based on CFBP can be considered as an promising approach for side-channel attacks.

6.8 Learning-Vector-Quantisation Neural Network (LVQ)

Vector Quantisation (VQ) has been extensively explored from theoretical and applied points of view. [147] and [148] are classical reviews. A vector quantiser maps k-dimensional vectors in the vector space R^k into a finite set of vectors $Y = \{y_i : i = 1, 2, ..., N\}$. Each vector y_i is called a code vector (CV) or a codeword and the set of all the codewords is called a codebook. Associated with each codeword y_i is a nearest-neighbour region called the Voronoi region, and it is defined by:

$$V_i = \{x \in R^k : ||x - y_i|| \leq ||x - y_j||, \quad \text{for all} \quad j \neq i\} \tag{6.12}$$

The set of Voronoi regions partitions the entire space R_k such that:

$$\bigcup_{i=1}^{N} V_i = R^k$$

$$\bigcap_{i=1}^{N} V_i = \phi$$

The main idea is to cover the input space of samples with code-book vectors, each representing a region labelled with a class. A code-book vector can be seen as a prototype of a class member, localised in the center of a class region in the input space. A class can be represented by an arbitrarily number of code-book vectors, but one code-book vector represents one class only; see Figure 6.17.

Moving into a supervised context, Learning Vector Quantisation (LVQ) [149] found a very important role in statistical pattern classification [150]. LVQ is a learning algorithm that combines competitive learning with supervision. In terms of neural networks a LVQ is a feed-forward net with a two-layer neural network, including a competitive layer and a linear layer. The competitive layer is the core layer that performs classification through learning. Each neuron in the competitive layer of the LVQ network learns to recognise a prototype vector, which allows it to classify a region of the input space. In using LVQ networks, the distances between the input vectors and the prototype vectors will be directly calculated to achieve classification. If two input vectors are close to each other, they belong to the same class. LVQ algorithms do not approximate density functions of class samples as do Vector Quantisation or

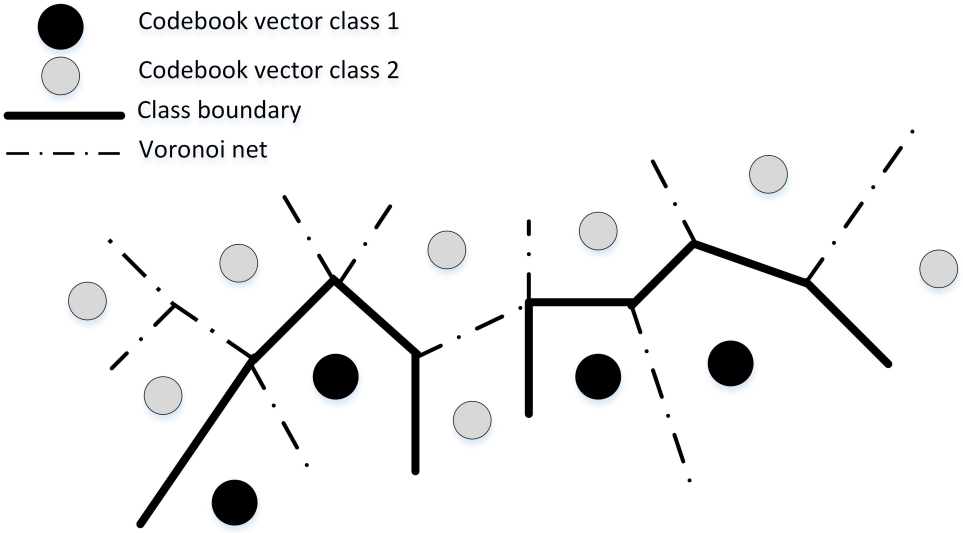


FIGURE 6.17: Classification of input space into class regions by codebook vectors in a two-dimensional feature space.

Probabilistic Neural Networks (PNN) do, but directly define class boundaries based on prototypes, Figure 6.17 shows the classification of input space into class regions by codebook vectors represented as neurons positioned in a two-dimensional feature space. The LVQ architecture in a neural network is shown in Figure 6.18. In LVQ there are no

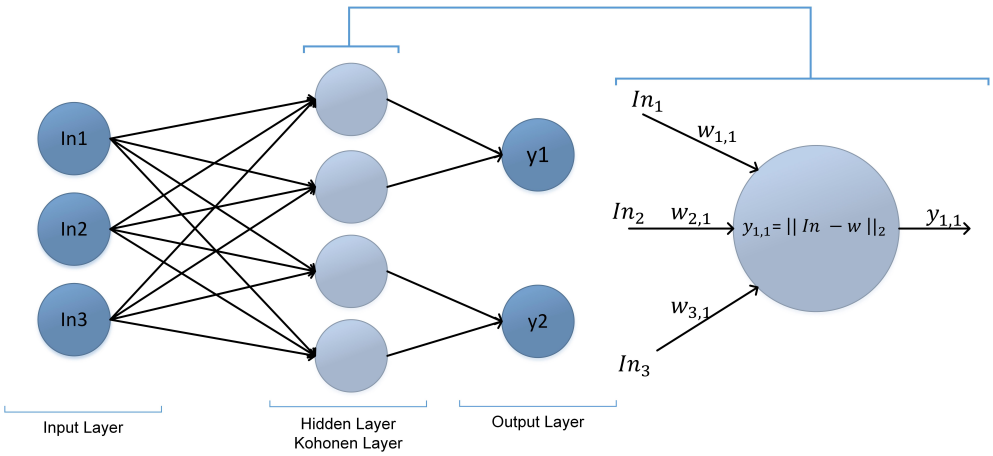


FIGURE 6.18: LVQ architecture: one hidden layer with Kohonen neurons, adjustable weights between input and hidden layer and a winner-takes-all mechanism.

general restrictions on the complexity of the problem domain like, for example, with simple neural network structures such as the classical Perceptron [151]. Compared with more complex neural network structures, such as multilayer Perceptrons, the simple LVQ topology is superior in transparency and speed with appropriate training algorithms.

A major disadvantage of the LVQ classifier is that it does not perform an interior scaling transformation. Therefore, the success of a classification scheme may be directly associated with an appropriate data preprocessing transformation to normalise data and discard non-relevant input features. Unfortunately, such preprocessing methods are quite inflexible: once defined, they can hardly be altered without discarding the information already gathered within the successive classifiers. Concerning the preprocessing stage, a common approach is to impose transformations on the original variables, generating more appropriate representations, e.g. principal component analysis (PCA).

6.8.1 LVQ Algorithm

The basic LVQ algorithm LVQ1 rewards correct classifications by moving the code vector (CV) towards a presented input vector, whereas incorrect classifications are punished by moving the CV in the opposite direction. The magnitudes of these weight adjustments are controlled by a learning rate which can be lowered over time in order to get finer movements in a later learning phase. Improved versions of LVQ1 are KOHONENs OLVQ1 with different learning rates for each CV in order to get faster convergence and LVQ2, LVQ2.1 and LVQ3.

A brief description of the most advanced training algorithm, LVQ3, will be given below. Detailed descriptions of the currently available training algorithms can be found in [152, 153].

First step: in an LVQ3 training iteration t is the determination of the two closest codebook vectors to the present training sample $x(t)$, which will be referred to as $m_i(t)$ and $m_j(t)$.

Second step: a symmetric 'window' of non-zero width around the mid-plane of m_i and m_j must be specified. A vector x is defined to lie in the 'window' if

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s$$

where s represents a constant factor, commonly chosen between 0.4 and 0.8, and d_i, d_j, are the distances of x to m_i and m_j respectively.

Third step: the LVQ3 training process then updates m_i and m_j according to the following equations

$$\begin{aligned} m_i(t+1) &= m_i(t) - \alpha(t)\left[x(t) - m_i(t)\right] \\ m_j(t+1) &= m_j(t) + \alpha(t)\left[x(t) - m_j(t)\right] \end{aligned} \tag{6.13}$$

if x falls into the 'window' and x and m_j belong to the same class, while x and m_i belong to different classes;

$$m_k(t+1) = m_k(t) + \epsilon\alpha(t)\left[x(t) - m_k(t)\right], k \in \{i, j\} \tag{6.14}$$

if x falls into the 'window' and x, m_j and m_i belong to the same class. Here $\alpha(t)$ is a scalar gain, decreasing monotonically in time. A common initial value $\alpha(0)$ is 0.03. Epsilon (ϵ)is a constant, applicable values are between 0.1 and 0.5 [153].

6.8.2 Experimental Results Based on LVQ

The basic measurement setup used to performed this experiment are an FPGA board with a SPARTAN 3 FPGA, a Tektronix TDS2012 oscilloscope with 1 Gigasample per second to record the power/electromagnet signal traces, a Tektronix CT1 current probe for measuring power consumption of FPGA, an ETS near-field probe set (model 7405) for measuring electromagnetic emission and an ETS broadband amplifier (model 7405-907b) to enhance the quality of the input signal traces (see Figure 3.10). This experiment was performed through a MATLAB R2015a toolbox and a PC configuration of Intel Core i7, 2.8 GHz and 16.00 GB RAM. For more details, refer to 3.5.

Concerning our LVQ-based analysis, the number of hidden layers plays an important part in overall neural network architecture and has a significant influence on the final output. There is no specific method or formula to determine this number, and hence this number must be carefully chosen via experiment. Using too few neurons can lead to under-fitting while too many neurons can contribute to computational complexity or over-fitting.

For this purpose, a comparison of classification accuracy, timing complexity and memory consumption between LVQ-based architectures with different numbers of hidden layers ranging from 10 to 110 is performed and the experimental results are provided in Table 6.3. As can be seen from this table, by increasing the number of hidden layers from 10 to 80, the error dropped from 0.135 to 0.060 and minimised at 0.057 with the number of hidden layers of about 90 to 100, after that the error increased by 0.013 with 110 hidden layers.

Concerning the timing complexity, the most time-consuming LVQ architectures are with the number of hidden layers between 90 and 100, with a range of $[14000, 16000]$ seconds. Other processing times increase gradually from 2303 to 12500 seconds as the number of hidden layers increases from 10 to 80.

Judging from the memory consumption information in this table, the memory consumption of all hidden layers are almost the same (in the range of [0.124, 0.129]). In Figure 6.19 the training performance of our LVQ-based classification is indicated. From this figure, the best training performance is 0.066556 at epoch 279. The efficiency of the LVQ-based analysis is verified through a confusion matrix in which the numbers of times that a particular key value was correctly classified or misclassified are presented; see Figure 6.20. The diagonal cells show the number of times that the key values were correctly classified, and the off-diagonal cells show the misclassified cases. The blue cell in the bottom right shows the total percentage of correctly classified cases (in green) and the total percentage of misclassified cases (in red). As can be seen, the results show a good classification because of the higher numbers of correct responses in the diagonal squares (143, 128, 110 and 136) compared to the low numbers of incorrect responses in the off-diagonal squares (ranging [0 23]). The lower-right blue square illustrates the overall accuracies of 86.7% correctly classified and 14.3% misclassified. Figure 6.21 illustrates Receiver Operating Characteristic (ROC) curves to check the quality of classifiers. For each class of a classifier (class of key-bit 1, key-bit 2, key-bit

TABLE 6.3: LVQ network performance comparison based on the number of hidden layers, timing complexity and memory consumption

No. Hidden layers	MSE	Time (Sec)	Mem(GB)
10	0.135	2303	0.124
20	0.112	3253	0.124
30	0.090	4932	0.124
40	0.080	5841	0.125
50	0.070	6909	0.125
60	0.065	9639	0.129
70	0.060	11247	0.128
80	0.062	12512	0.127
90	0.057	13915	0.129
100	0.057	16023	0.128
110	0.070	5985	0.126

3 and key-bit 4), ROC applies threshold values across the interval [0,1] to outputs. For each threshold, two values are calculated, the True Positive Ratio (the number of outputs greater than or equal to the threshold, divided by the number of one targets), and the False Positive Ratio (the number of outputs less than the threshold, divided by the number of zero targets). In Figure 6.21, the coloured lines represent the ROC curves which are plots of the true positive rate versus the false positive rate as the threshold is varied. A perfect test would show points in the upper-left corner. From this figure, the best classification was for key-bit 1, then for key-bits 4, 2 and 3.

6.8.3 Summary

This section is dedicated to pattern recognition and classification of side-channel information based on learning vector (LVQ) quantisation neural network. This approach is adopted because of some features;

- There are no general restrictions on the complexity of the problem domain.
- A simple and fast topology with an appropriate training algorithm.

Judging from the results of this section, LVQ architectures with between 90 and 100 hidden layers can be considered as the most accurate architectures, although they are known as the slowest ones. In addition, judging from the confusion matrices and error histogram, our results indicate an overall accuracy of 86% correctly classified, 14% misclassified and no significant over-fitting; therefore, LVQ can be considered as a promising approach of side-channel data characterisation.

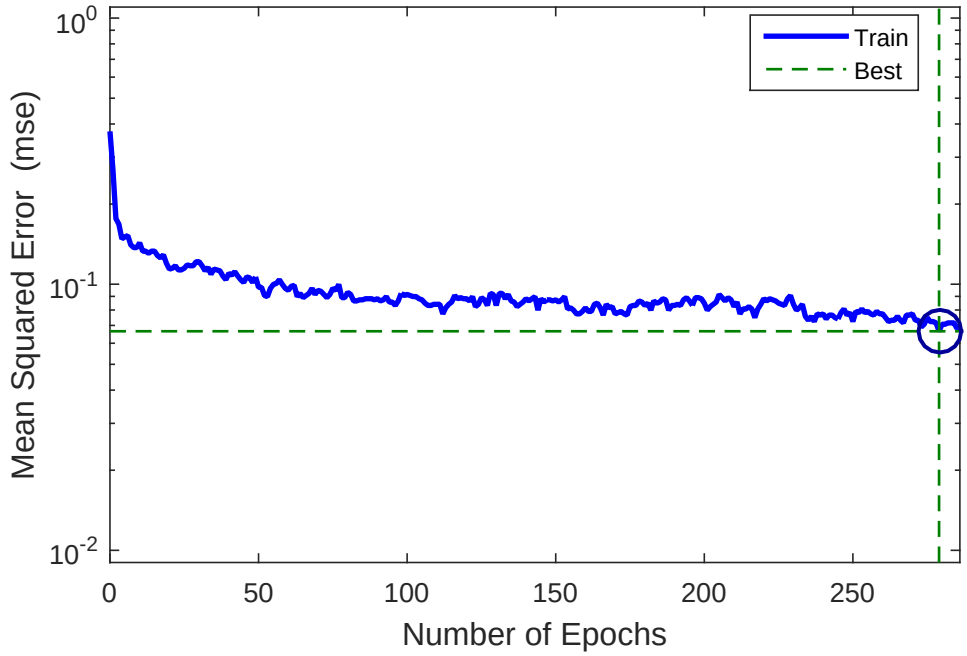


FIGURE 6.19: The training performance of our LVQ-based classification. Best Training Performance is 0.066556 at epoch 279.

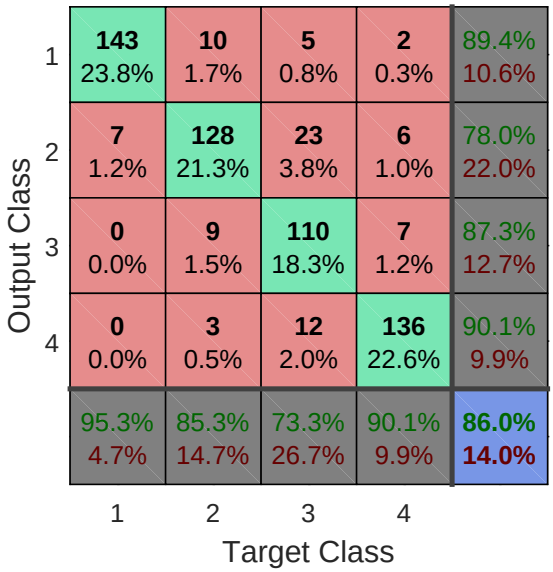


FIGURE 6.20: LVQ network Confusion matrix. The diagonal cells (green): the number of correctly classified cases, The off-diagonal cells (red): the number of misclassified cases, The blue cell: the total percentages.

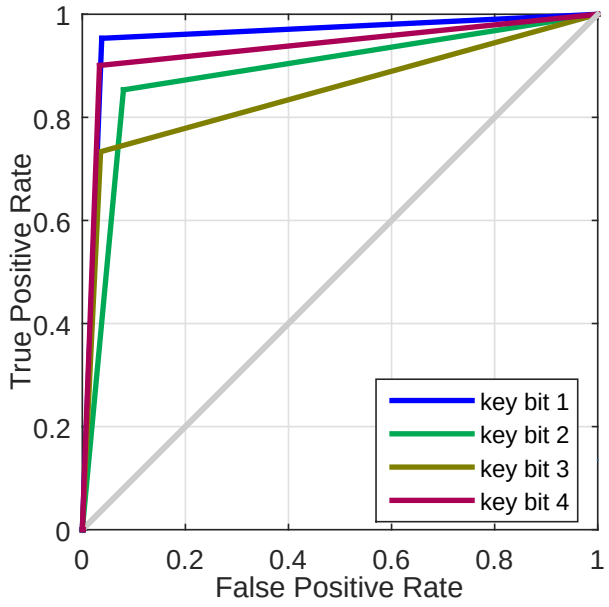


FIGURE 6.21: Receiver Operating Characteristic (ROC) curves based on LVQ analysis. The best classification was for key-bit 4, then for key bits 1, 3 and 2.

6.9 A Comparison of the Performance of Various Neural-Network Architectures in Side Channel Information Analysis

Neural networks have emerged as an important tool for the characterisation of side channel information analysis. The recent literature considers neural networks as a promising alternative to various conventional side-channel attacks.

Although significant progress has been made in different applications of classification-related areas of neural networks, a number of issues in applying neural networks in side-channel analysis still remain and have not been solved completely. In this section, some empirical-comparison issues of different architectures of neural networks in SCA are reviewed and discussed. This section aims to provide a summary and comparison of the most powerful neural networks for multi-class classification of side channel information. The strengths and weaknesses of each network as well as their performance in terms of accuracy, memory and time consumption are also discussed. Although many types of neural networks can be used for multi-class classification purposes, our focus is on feed-forward back-propagation (FFBP), Probabilistic neural network (PNN), Cascade-forward back propagation (CFBP) and Learning vector quantisation (LVQ) which are the most widely studied and used neural-network classifiers. Below is a short summary of these networks.

- **FFBP**: The feed-forward neural network was the first and simplest type of neural network in which the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There

are no cycles or loops in the network. In order to training the network, back-propagation calculates the gradient of an error function with respect to all the weights in the network. The gradient is fed to the optimisation method which in turn uses it to update the weights, in an attempt to minimise the error.

- **PNN**: A probabilistic neural network (PNN) is a feed-forward neural network, which provides a general solution to pattern classification problems by following an approach developed in statistics, called Bayesian classifiers; see [141].

- **CFBP**: The main feature of CFBP architecture is to build up the cascade architecture by adding new neurons together with their connections to all the inputs as well as to the previous hidden neurons. This configuration is not changed at the following layers [144]. The other special feature of CFBP is to use only the newly created neuron by fitting its weights so that to minimise the residual error of the network. The new neurons are added to the network while its performance increases.

- **LVQ**: learning-vector-quantisation neural networks consist of two layers. The first layer maps input vectors into clusters that are found by the network during training. The second layer merges groups of first-layer clusters into the classes defined by the target data.

Table 6.4 provides a comparison of the advantages and disadvantages of different multi-class classifiers in side-channel information analysis. This table is based on the theoretical information and experimental results discussed in this chapter. Judging from this table, FFBP is known as a simple type of network which can be easily implemented while providing an acceptable accuracy. However the results are sensitive to network parameters such as the number of hidden layers and training functions, and any small change in these parameters can drastically affect the results or cause under/over fitting.

PNN is the fastest algorithm in terms of training speed. Moreover, it is robust to the noisy samples; therefore, it can be a good choice for real-time analysis, although PNN is limited to applications involving relatively small datasets. Large datasets cause computational complexity. In addition, this could saturate the feature space with overlapping Gaussian functions that would increase the rate of misclassification.

In CFBP, no structure of a network is predefined, that is, the network is automatically built up from the training data, so a smaller number of neurons can be applied, which can lead not only to faster learning but also to low memory consumption while still being accurate. On the other hand, a disadvantage of CFBP is that the cascaded networks can be over-fitted in the presence of noisy features.

In LVQ there are no general restrictions on the complexity of the problem domain. Compared with other neural-network structures, LVQ topology is superior in transparency and speed with appropriate training algorithms. However a major disadvantage of the LVQ classifier is that it does not perform an interior scaling transformation. Therefore, the success of a classification scheme may be directly associated

TABLE 6.4: Comparison of advantages and disadvantages of neural-network classifiers

	Advantages	Disadvantages
FFBP	- Good accuracy - Easy to implement	- Sensitive to network parameters - Low training speed - Vulnerable to under-fitting - Vulnerable to over-fitting
PNN	- The highest training speed - Robust to the noise samples - Efficient for real-time tasks	- High memory consumption - High computational complexity - Not accurate
CFBP	- The most accurate - Low memory consumption - Fast learning	- Not efficient with noise - Vulnerable to over-fitting
LVQ	- Fast training - No general complexity limit - Good accuracy	- Needs appropriate preprocessing stage - The slowest network

with an appropriate data preprocessing transformation to normalise data and discard non-relevant input features. In addition it is known as the slowest network.

Table 6.5 indicates a general comparison of the experimental results of side-channel information characterisation based on different neural-network classifiers (FFBP, PNN, CFBP and LVQ). For each network different architectures and parameters (such as training functions, number of hidden layers) have been tested and verified. Having determined the most proper network architectures and parameters, their general performance is compared in terms of MSE error, timing complexity and memory consumption. As shown in this table, FFBP and CFBP are the most accurate classifiers of side channel information with an MSE-range of [0.01 0.18] and [0.01 0.27] respectively, while PNN with 0.37 has the worst accuracy. In more detail, the best accuracy is known for CFBP with 27 hidden layers and a *Bayesian Regularisation* training function, and FFBP with 29 hidden layers and the same training function.

In addition to accuracy, the computational complexity of the algorithms plays an important role in their efficiency and should be taken into consideration. Regarding the timing complexity, as shown in this table, PNN is considered to be the fastest algorithm and needs only tw seconds of processing time, whereas CFBP and LVQ need about 8600 and 13000 seconds to achieve the best accuracy, significantly greater than other algorithms. According to this table, sometimes a little improvement of the accuracy, drastically affects the network complexity. For example, the processing time for an FFBP network can be about 16 seconds (with 25 hidden layers and *Levenberg-Marquardt* training function) to have 0.06 MSE while in order to get 0.017 MSE, a network architecture is needed with 29 hidden layers and *Bayesian Regularisation* as the training function. The processing time of this architecture will be approximately 6584 seconds.

Judging from the memory consumption information in this table, FFBP (with 29

Table 6.5: Performance comparison of neural-network classifiers

Network Properties			**MSE**		**Time (Sec)**		**Mem (GB)**	
	Training Function	No. H.L		Range		Range		Range
FFBP	Bayesian Regularisation	29	0.0170	[0.01 0.18]	6584	[4 6584]	0.019	[0.01 1.77]
	Levenberg-Marquardt	25	0.059		16.23		1.155	
	BFGS quasi-Newton	32	0.084		335		0.015	
PNN	Bayesian Regularisation	-	0.37	[0.3 0.4]	2	[0.2 3]	1.55	[1.55 1.66]
CFBP	Levenberg-Marquardt	35	0.067	[0.01 0.27]	46.23	[1 8600]	1.184	[0.02 1.185]
	Bayesian Regularisation	27	0.014		8600		0.02	
LVQ	Based on Learning Vectr Quantisation	90	0.057	[0.05 0.135]	13915	[2303 16023]	0.129	[0.124 0.129]

TABLE 6.6: Accuracy comparison of key-bit classification

	Correctly Classified %					**Misclassified** %					**Outliers**
	k.b1	k.b2	k.b3	k.b4	avg	k.b1	k.b2	k.b3	k.b4	avg	
FFBP	89.7	82.2	88.0	91.9	88.0	10.3	17.8	12.0	8.1	12.0	NO
PNN	73.2	70.0	69.8	75.0	72.0	26.8	30	30.2	25	28.0	NO
CFBP	75.7	62.0	76.9	89.1	75.7	24.3	38.0	23.1	10.9	24.3	NO
LVQ	89.4	78.0	87.3	90.1	86.0	10.6	22.0	12.7	9.9	14.0	NO

hidden layers and *Bayesian Regularisation* training function) and CFBP (with 27 hidden layers and *Bayesian Regularisation* training function) consume the least amount of memory at approximately 0.02 GB, while all other algorithms need almost the same size of memory ranging from 0.1 to 1.5 GB.

In order to compare the classification accuracy of our experimental results, Table 6.6 is provided to indicate the percentages of correctly classified and misclassified samples along with the best validation and outlier status. The classification results are also presented for each class separately (class of key-bit 1, key-bit 2, key-bit 3 and key-bit 4) to show the efficiency of classification for each class. As can be seen from this table, FFBP and LVQ have a good performance, correctly classifying 88% and 86% respectively, whiles the corresponding percentages for CFBP and PNN are 75.7% and 72%. The outliers may occur in the results due to variability in measurement; hence the outliers of our result are checked to determine if the data is bad, or if those data points are different from the rest of the dataset. If the outliers are valid data points, but are unlike the rest of the data, then the network extrapolates for these points; therefore, more data should be collected that looks like the outlier points, and the network should be retrained. In our experiment outliers only correspond to analysis based on a few instances and after training with enough samples the results would be reliable.

6.9.1 Summary

In this chapter, characterisation of side-channel information based on the most powerful neural networks in multi-class classification is investigated. For this purpose, we applied feed-forward back-propagation (FFBP), Probabilistic neural network (PNN), Cascade-forward back propagation (CFBP) and Learning-vector quantisation (LVQ) which are the most widely studied and used neural network classifiers. Regarding our experimental results, the performance of these classifiers is compared through some comparison tables which provide useful information for cryptosystem designers. The strengths and weaknesses of each classifier as well as their efficiency in terms of accuracy and computational complexity are presented. From our results, it can be inferred that FFBP with 25 hidden layers and the training function of *Levenberg-Marquardt* is the most efficient neural network architecture for characterisation of side-channel information. Although FFBP or CFBP with the *Bayesian Regularisation* training

function are more accurate than FFBP with *Levenberg-Marquardt*, their processing time is significantly long; they, therefore, cannot be considered as an efficient approach of side-channel analysis especially in real-time attacks.

7

Thesis Conclusion and Recommendations for Future Work

7.1 Thesis Conclution

Cryptosystems have been widely used in different applications which play an important role in our modern life. In this thesis, elliptic-curve cryptography (ECC) is introduced as a promising public-key cryptosystems. In addition, its physical vulnerabilities and implementation considerations are discussed. ECC offers two major benefits over conventional cryptographic algorithms; it has more security per bit and a suitable key size for hardware and modern communications.

Cryptosystems, even after recent improvements in mathematical cryptography algorithms, are still vulnerable to Side-Channel Attack (SCA) which has been found to be a powerful class of attack against all implementations of cryptographic algorithms. In this circumstance, both attacks and countermeasures interact strongly, as countermeasures get broken by improved attacks and new countermeasures are developed to thwart ever more advanced attacks. Although several countermeasures against conventional attacks have been proposed, some inherent leakages during single executions in a cryptography algorithm cannot be prevented in many cases. Another challenging issue is that side-channel information analysis involves a huge input dataset with high signal-noise ratio, because a high sampling rate is usually mandatory in order to retain the frequency content of the side channel. This leads to inaccuracy, excessive computational loads and a prohibitively large memory usage.

This thesis proposes a powerful and promising method of SCA based on machine-learning techniques in the forms of Neural Networks (NN), Support Vector Machines (SVM) and Principal Component Analysis (PCA). Regarding our experimental results based on an FPGA implementation of elliptic curve cryptography (ECC), PCA can be

used as a strong preprocessing stage to reduce the signal-noise ratio, data-set dimension and algorithm complexity. In addition, the most efficient machine-learning techniques for side-channel information characterisation are LVQ neural network (with a number of hidden layers between 90 and 100), and SVM with Gaussian RBF kernel function with parameter p value of 5 and 50 for **CS** and **M-SVM**^{2} SVM models respectively with about 80 to 85 % accuracy.

7.2 Future Work Directions

This research has laid significant groundwork for further investigation in the application of machine learning techniques in side channel data characterisation. Deep learning refers to artificial neural networks that are composed of many layers. one of the promising approach for future work can be deep learning which is a growing trend in machine learning due to some favourable results in applications where the target function is very complex and the datasets are large. This algorithm is based on a set of algorithms that attempt to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations.

A further contribution can be in preprocessing stage and investigating the performance of other approaches such as Linear discriminant analysis (LDA) and Fisher discriminant analysis (FDA) in comparison with PCA. LDA is a generalisation of Fisher's linear discriminant, a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or event.

In addition, in terms of measurement set-up, micro probing technique is recommended as a more precisely method for measuring the side channel information rather than conventional measurement using probe.

A Measurement Probes

Different types of current and electromagnetic probes used in this project are:

A.1 Current Probe

Tektronix CT1: for measuring the power consumption of our FPGA board. The CT1 Current Probe (Figure A.1) is designed for permanent or semi-permanent in-circuit installation. It consists of a current transformer and an interconnecting cable. The current transformer has a small hole through which a current-carrying conductor is passed during circuit assembly. The FPGA board uses three discrete regulators to generate the necessary voltages. A 1.2V regulator supplies power to the FPGA's V_{CCINT} voltage inputs, which power the FPGA's core logic. In order to measure the power consumption of the FPGA, a CT1 probe is installed between the 1.2 V regulator and the FPGA IC. Figure A.2 illustrates how CT1 is used to measure the power consumption of the FPGA IC alone.

FIGURE A.1: Tektronix CT1 current probe

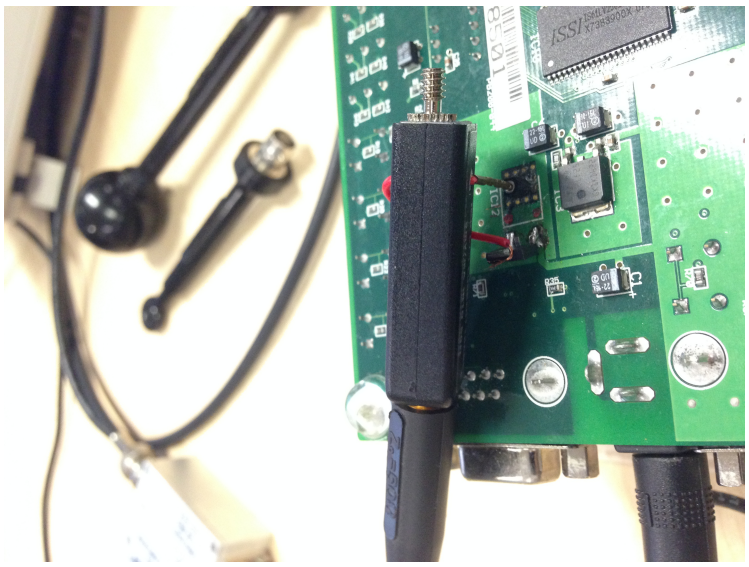


FIGURE A.2: CT1 connection to measure the FPGA power consumption

A.2 Electromagnetic probes

Various commercial and hand-crafted probes are used and tested in this project.

A.2.1 Hand-crafted probes

Hand-crafted probes have been used widely in side-channel measurement [7], Figure A.3.

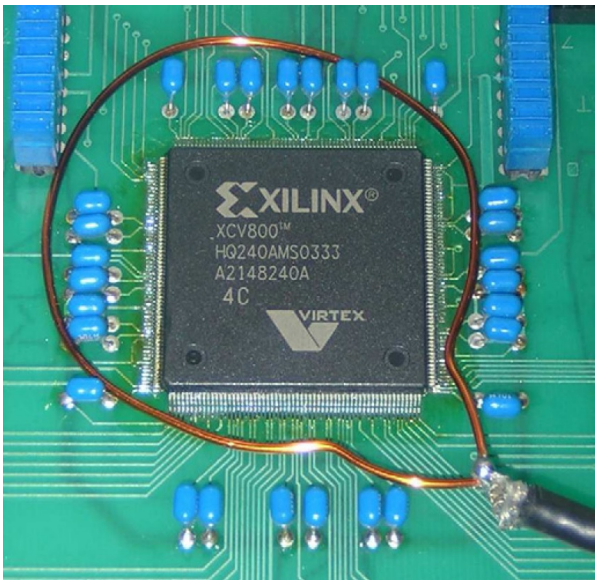


FIGURE A.3: The use of hand-crafted electromagnetic probe in literature. Taken from [7]

Magnetic probe works based on the fact that magnetic field passing through the probe loop generates a voltage according to Faradays law, which states that the induced voltage is proportional to the rate of change of magnetic flux through a circuit loop. At very low frequencies a voltage would be induced directly in the internal loop conductor, but the copper sheath is quite a good shield to magnetic fields at frequencies exceeding the low KHz range. So at high frequency, a voltage is then induced preferentially in the outer sheath loop, and this appears across the sheath gap [154]. The metal sheath thickness is several skin depths, so this prevents direct interaction between currents on the external surface and internal surfaces of the shield.

The magnetic field probes are made in the form of a loop with an inherent electrostatic shield, generally from 50 Ohm semi-rigid coaxial cable. The loop is formed by making a circle or square from semi-rigid coax with a gap placed symmetrically in the middle of the loop. The position of the gap is very important to the performance of the shield. If the gap is not in the middle of the loop, shielding effectiveness is compromised. Square loops are useful for making measurements of circuit voltage and current while either square or round loops are suitable for measuring magnetic fields in free space. They vary slightly in configuration and in characteristics, but essentially they are electrically small shielded loop antennas derived from the antennas used for radio communication and direction finding. Figure A.4 shows our hand-crafted electromagnetic probes.

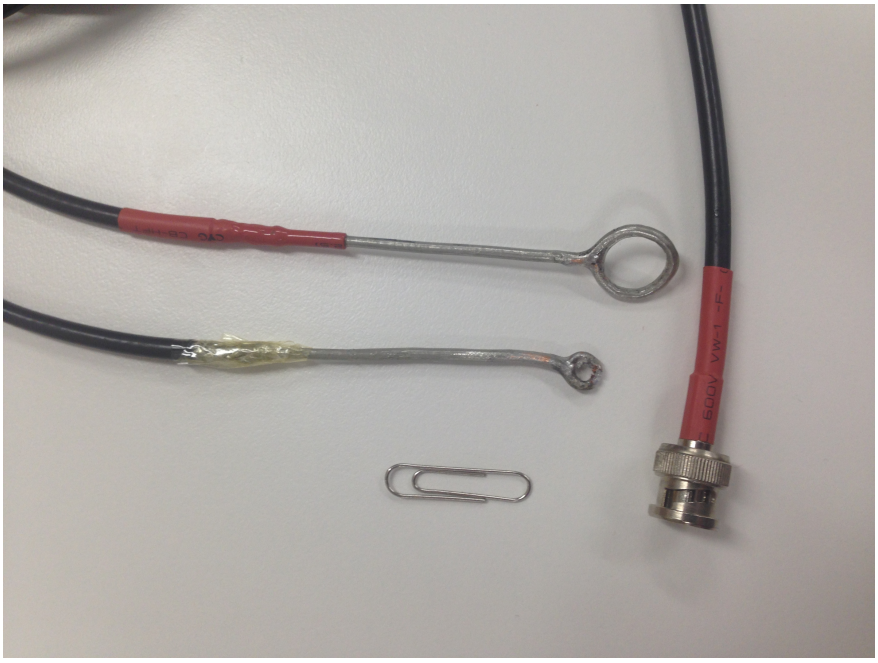


Figure A.4: The hand-crafted electromagnetic probes verified in this project

A.2.2 Commercial probes

ETS near-field probe: An ETS near-field probe set (model 7405) (Figure A.5) is used for measuring the electromagnetic emission of our FPGA board. This set consists of three loop probes, one stub and one ball probe, an extension handle and an optional battery-powered pre-amplifier. The handle of each probe terminates in a BNC connector. These probes are designed to be used with a signal-analysing device such as an oscilloscope or a spectrum analyser.

FIGURE A.5: ETS near-field probe set (model 7405)[2]

- Ball probe (Figure A.6): is a sensitive one. The larger sensing element does not offer the highly-refined definition of the source location which the stub probe allows, but it is capable of tracing much weaker signals. The impedance of the stub probe is essentially the same as that of a non-terminated length of 50 ohm coaxial cable.

FIGURE A.6: ETS near-field ball probe [2]

- Surface probe (Figure A.7): The surfaces of bus structures, large components or supply structures emit E-fields that can cause EMI. The bottom of the surface probe detects these fields on an area measuring approx.

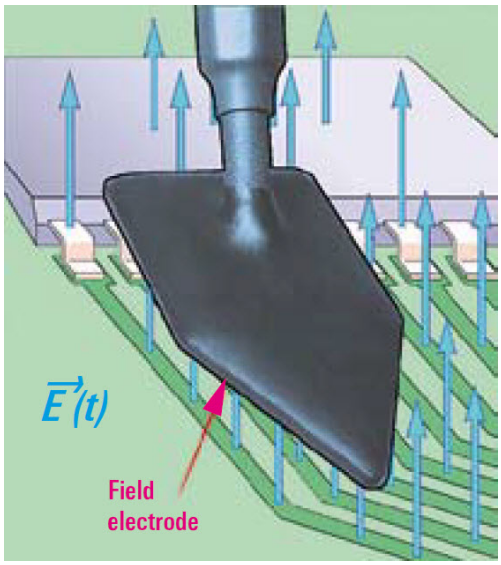


Figure A.7: Surface probe (model RSE02) [8]

- Narrow probe (Figure A.8): The narrow electrode of the this probe can select a single conductor track from a bundle of conductor tracks 0.2 mm in width. The light colour of the probe tip stands out in sharp contrast to the dark green of the printed circuit board.

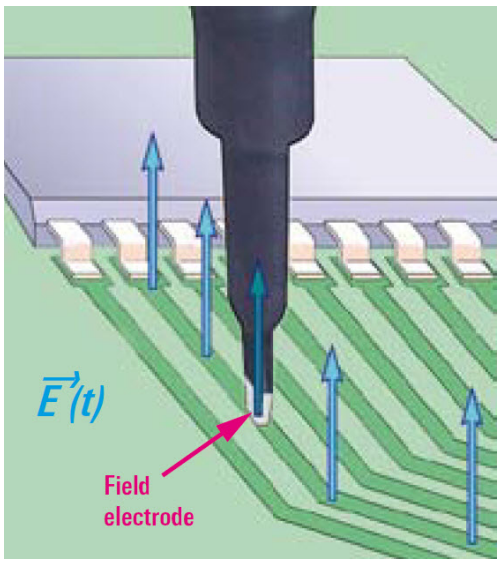


Figure A.8: Narrow probe (model RSE10) [8]

- STUB probe (Figure A.9): This probe can be used to selectively detect the current spectrum in conductor tracks and component leads such as on capacitors

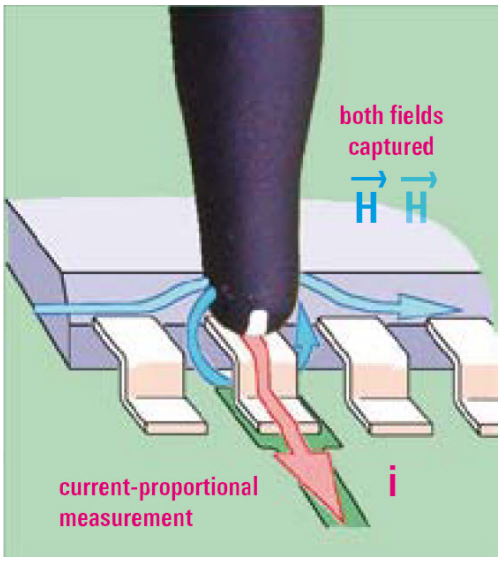


FIGURE A.9: STUB probe (model RSH50-1) [8]

or ICs. The probe tip has a magnetically active groove of approx. 0.5 mm in width.

B

Software Considerations

The analytical stage of this project was conducted through a MATLAB toolbox, and the multi-class classification procedure is performed through MSVMpack 1.5 [127], which is an open-source package dedicated to multi-class support-vector machines and can handle classification problems with more than two classes, calculating the time complexity. Following is a list of the functions used:

- **Timing Complexity**: *"etime"* and *"clock"* command.
  e = etime(t2,t1) returns the number of seconds between two date vectors or matrices of date vectors, t1 and t2.
  c = clock returns a six-element date vector containing the current date and time in decimal form.

  c = clock;
  Algorithm
  e = etime(clock,c);

- **Memory Consumption**: *"Memory"*
  The *"Memory"* function displays information showing how much memory is available and how much the MATLAB software is currently using.
  ≫ Memory
  `Maximum possible array: 14253 MB (1.495e+10 bytes)`
  `Memory available for all arrays: 14253 MB (1.495e+10 bytes)`
  `Memory used by MATLAB: 747 MB (7.833e+08 bytes) Physical Memory (RAM): 12279 MB (1.288e+10 bytes)`

- **Cross-Validation**: *"crossvalind"*
  [Train, Test] = crossvalind("Holdout", N, P) returns logical index vectors for

cross-validation of N observations by randomly selecting $P * N$ (approximately) observations to use for the evaluation set. P must be a scalar between 0 and 1. P defaults to 0.5 when omitted, corresponding to holding 50% out. Using holdout cross-validation within a loop is similar to K-fold cross-validation one time outside the loop, except that non-disjointed subsets are assigned to each evaluation.

- **Principal component analysis**: *"pca"*
  [coeff,score,latent] = pca(X) returns the principal component coefficients, also known as loadings, for the n-by-p data matrix X. The rows of X correspond to observations and the columns correspond to variables. The coefficient matrix is p-by-p. Each column of "coeff" contains coefficients for one principal component, and the columns are in descending order of component variance. This function also returns the principal component scores in "score" and the principal component variances in "latent". You can use any of the input arguments in the previous syntaxes.

- **Support-vector machine**: *"svmclassify"*
  Group = svmclassify(SVMStruct,Sample) classifies each row of the data in *Sample*, a matrix of data, using the information in a support-vector machine classifier structure "SVMStruct", created using the svm train function. Like the training data used to create "SVMStruct", *Sample* is a matrix where each row corresponds to an observation or replicate, and each column corresponds to a feature or variable. Therefore, *Sample* must have the same number of columns as the training data. This is because the number of columns defines the number of features. Group indicates the group to which each row of *Sample* has been assigned.

- **Neural-network functions**:
  Feed-forward back-propagation networks: *feedforwardnet(hiddenSizes,trainFcn)*
  Cascade back-propagation network: *cascadeforwardnet(hiddenSizes,trainFcn)*
  Probabilistic neural network: *newpnn(P,T,spread)*
  Learning-vector quantisation neural network: *lvqnet(hiddenSize,lvqLR,lvqLF)*

List of Acronyms/Abbreviations

ALU	Arithmetic Logic Unit
CFBP	Cascade and Forward Back-Propagation Neural Network
CS	Crammer and Singer
CV	Code Vector
DAG	Directed Acyclic Graph
DEMA	Differential Electromagnetic Analysis
DH	Diffie-Hellman
DPA	Differential Power Analysis
DSA	Digital Signature Algorithm)
ECC	Elliptic Curve Cryptography
ECP	ECC Processor
ECPA	Elliptic-Curve Point Addition
ECPD	Elliptic-Curve Point Doubling
ECPM	Elliptic-Curve Point Multiplication
ECSM	Elliptic-Curve Scalar Multiplication
EM	Electromagnetic
FFBP	Feed-Forward Back-Propagation
FFMA	Finite-Field Modular Arithmetic
LLW	Lee, Line and Wahba
LVQ	Learning-Vector-Quantisation Neural Network
PCA	Principal Component Analysis
PNN	Probabilistic Neural Network

MSE	Mean Squared Error
NAF	Non-Adjacent Form
NN	Neural Networks
RPA	Refined Power Analysis
ROC	Receiver Operating Characteristic
RTL	Register Transfer Level
SCA	Side-Channel Attacks
SEMA	Simple Electromagnetic Analysis
SPA	Simple Power Analysis
SVM	Support Vector Machines
WW	Weston and Watkins
ZPA	Zero-Value Point Attack

Bibliography

[1] *Tektronix ct1 current probe*. Url date: Jun 2016, URL http://www.tek.com/datasheet/current-probe.

[2] *Ets near-field probe set (model 7405)*. Url date: Jun 2016, URL http:http://www.ets-lindgren.com/7405.

[3] *Ets broadband amplifier (model 7405-907b)*. Url date: Jun 2016, URL http:http://www.ets-lindgren.com/7405.

[4] C. J. Burges. *A tutorial on support vector machines for pattern recognition*. Data mining and knowledge discovery **2**(2), 121 (1998).

[5] W. Xie, D. Hou, and Q. Song. *Bullet-hole image classification with support vector machines*. In *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, vol. 1, pp. 318–327 (IEEE, 2000).

[6] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. *Large margin dags for multiclass classification.* In *nips*, vol. 12, pp. 547–553 (1999).

[7] E. D. Mulder, S. rs, B. Preneel, and I. Verbauwhede. *Differential power and electromagnetic attacks on a fpga implementation of elliptic curve cryptosystems*. Computers & Electrical Engineering **33**, 367 (2007).

[8] *Probe set for e and h near-field measurements*. Url date: Jun 2016, URL http:https://www.rohde-schwarz.com/au/product/hz15-productstartpage_63493-8985.html.

[9] *National Institute of Standards and Technology, Digital Signature Standard, FIPS Publication 186-2* (. Gaithersburg, MD, USA: NIST, 2000).

[10] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography* (Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003).

[11] D. Hankerson, J. L. Hernandez, and A. Menezes. *Software implementation of elliptic curve cryptography over binary fields*. In *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '00, pp. 1–24 (Springer-Verlag, London, UK, UK, 2000). URL http://dl.acm.org/citation.cfm?id=648253.752410.

[12] K.-H. Yeh. *A lightweight authentication scheme with user untraceability*. Frontiers of Information Technology & Electronic Engineering **16**, 259 (2015).

[13] C.-T. Li and C.-C. Lee. *A robust remote user authentication scheme using smart card*. Information Technology and Control **40**(3), 236 (2011).

[14] D. Wang, C.-G. Ma, Q.-M. Zhang, and S. Zhao. *Secure password-based remote user authentication scheme against smart card security breach*. Journal of Networks **8**(1), 148 (2013).

[15] C.-G. Ma, D. Wang, and Q.-M. Zhang. *Cryptanalysis and improvement of sood et al.s dynamic id-based authentication scheme*. In *Distributed Computing and Internet Technology*, pp. 141–152 (Springer, 2012).

[16] C.-G. Ma, D. Wang, and S.-D. Zhao. *Security flaws in two improved remote user authentication schemes using smart cards*. International Journal of Communication Systems **27**(10), 2215 (2014).

[17] P. C. Kocher. *Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems*. In *Advances in Cryptology - CRYPTO96*, pp. 104–113 (Springer, 1996).

[18] J.-S. Coron. *Resistance against differential power analysis for elliptic curve cryptosystems*. In *Cryptographic Hardware and Embedded Systems*, pp. 292–302 (Springer, 1999).

[19] C. D. Walter. *Simple power analysis of unified code for ecc double and add*. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pp. 191–204 (Springer, 2004).

[20] D. Stebila and N. Thériault. *Unified point addition formulæ and side-channel attacks* (Springer, 2006).

[21] P. Kocher, J. Jaffe, and B. Jun. *Differential power analysis*. In M. Wiener, ed., *Advances in Cryptology CRYPTO 99*, vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397 (Springer Berlin Heidelberg, 1999). URL http://dx.doi.org/10.1007/3-540-48405-1_25.

[22] E. De Mulder, P. Buysschaert, S. Ors, P. Delmotte, B. Preneel, G. Vandenbosch, and I. Verbauwhede. *Electromagnetic analysis attack on an fpga implementation of an elliptic curve cryptosystem*. In *Computer as a Tool, 2005. EUROCON 2005. The International Conference on*, vol. 2, pp. 1879–1882 (IEEE, 2005).

[23] R. R. Goundar, M. Joye, and A. Miyaji. *Co-z addition formulæ and binary ladders on elliptic curves*. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, pp. 65–79 (Springer, 2010).

[24] T. Akishita and T. Takagi. *Zero-value point attacks on elliptic curve cryptosystem* (Springer, 2003).

[25] I. Biehl, B. Meyer, and V. Müller. *Differential fault attacks on elliptic curve cryptosystems*. In *Advances in CryptologyCRYPTO 2000*, pp. 131–146 (Springer, 2000).

[26] J. Rolt, A. Das, G. Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede. *A new scan attack on rsa in presence of industrial countermeasures*. In W. Schindler and S. A. Huss, eds., *Constructive Side-Channel Analysis and Secure Design*, vol. 7275 of *Lecture Notes in Computer Science*, pp. 89–104 (Springer Berlin Heidelberg, 2012). URL http://dx.doi.org/10.1007/978-3-642-29912-4_8.

[27] S. Chari, J. R. Rao, and P. Rohatgi. *Template attacks*. In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pp. 13–28 (Springer, 2003).

[28] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater. *Template attacks in principal subspaces*. In *Cryptographic Hardware and Embedded Systems-CHES 2006*, pp. 1–14 (Springer, 2006).

[29] M. Medwed and E. Oswald. *Template attacks on ecdsa*. In *Information Security Applications*, pp. 14–27 (Springer, 2008).

[30] E. Saeedi and Y. Kong. *Side channel information analysis based on machine learning*. In *Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on*, pp. 1–7 (IEEE, 2014).

[31] T. Bartkewitz and K. Lemke-Rust. *Efficient template attacks based on probabilistic multi-class support vector machines* (Springer, 2013).

[32] E. Saeedi, M. S. Hossain, and Y. Kong. *Multi-class svms analysis of side-channel information of elliptic curve cryptosystem*. In *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2015 International Symposium on*, pp. 1–6 (IEEE, 2015).

[33] A. Heuser and M. Zohner. *Intelligent machine homicide*. In *Constructive Side-Channel Analysis and Secure Design*, pp. 249–264 (Springer, 2012).

[34] S. S and Haykin. *Neural networks and learning machines*, vol. 3 (Pearson Education Upper Saddle River, 2009).

[35] G. Cybenko. *Approximation by superpositions of a sigmoidal function*. Mathematics of control, signals and systems **2**(4), 303 (1989).

[36] S. Tillich and C. Herbst. *Attacking state-of-the-art software countermeasuresa case study for aes*. In *Cryptographic Hardware and Embedded Systems–CHES 2008*, pp. 228–243 (Springer, 2008).

[37] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks: Revealing the secrets of smart cards*, vol. 31 (Springer Science & Business Media, 2008).

[38] M. Joye and C. Tymen. *Protections against differential analysis for elliptic curve cryptographyan algebraic approach.* In *Cryptographic Hardware and Embedded SystemsCHES 2001*, pp. 377–390 (Springer, 2001).

[39] E. Brier and M. Joye. *Weierstraß elliptic curves and side-channel attacks.* In *Public Key Cryptography*, pp. 335–345 (Springer, 2002).

[40] B. Chevallier-Mames, M. Ciet, and M. Joye. *Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity.* Computers, IEEE Transactions on **53**(6), 760 (2004).

[41] C. Giraud and V. Verneuil. *Atomicity improvement for elliptic curve scalar multiplication.* In *Smart Card Research and Advanced Application*, pp. 80–101 (Springer, 2010).

[42] G. Hachez and J.-J. Quisquater. *Montgomery exponentiation with no final subtractions: Improved results.* In *Cryptographic Hardware and Embedded SystemsCHES 2000*, pp. 293–301 (Springer, 2000).

[43] C. D. Walter. *Montgomerys multiplication technique: How to make it smaller and faster.* In *Cryptographic Hardware and Embedded Systems*, pp. 80–93 (Springer, 1999).

[44] H. Sato, D. Schepers, and T. Takagi. *Exact analysis of montgomery multiplication.* In *Progress in Cryptology-INDOCRYPT 2004*, pp. 290–304 (Springer, 2004).

[45] J. Heyszl, S. Mangard, B. Heinz, F. Stumpf, and G. Sigl. *Localized electromagnetic analysis of cryptographic implementations.* In *Topics in Cryptology–CT-RSA 2012*, pp. 231–244 (Springer, 2012).

[46] K. Itoh, T. Izu, and M. Takenaka. *Address-bit differential power analysis of cryptographic schemes ok-ecdh and ok-ecdsa.* In *Cryptographic hardware and embedded systems-CHES 2002*, pp. 129–143 (Springer, 2003).

[47] E. Prouff. *Constructive Side-channel Analysis and Secure Design* (Springer, 2014).

[48] B. Kopf and M. Durmuth. *A provably secure and efficient countermeasure against timing attacks.* In *Computer Security Foundations Symposium, 2009. CSF'09. 22nd IEEE*, pp. 324–335 (IEEE, 2009).

[49] H. K. Kalita and A. Kar. *Wireless sensor network security analysis.* International Journal of Next-Generation Networks (IJNGN) **1**(1), 1 (2009).

[50] Z. Gou, S. Yamaguchi, and B. Gupta. *Analysis of various security issues and challenges in cloud computing environment: A survey.* In *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security*, pp. 393–419 (IGI Global, 2016).

[51] Y. Ren, L. Wu, H. Li, X. Li, X. Zhang, A. Wang, and H. Chen. *Key recovery against 3des in cpu smart card based on improved correlation power analysis.* Tsinghua Science and Technology **21**(2), 210 (2016).

[52] R. N. Akram, K. Markantonakis, and D. Sauveron. *Recovering from a lost digital wallet: A smart cards perspective extended abstract.* Pervasive and Mobile Computing **29**, 113 (2016).

[53] E. Kim and J. Moon. *A new approach to deploying private mobile network exploits.* The Journal of Supercomputing **72**(1), 46 (2016).

[54] J. Hernandez-Castro and G. Avoine. *Cryptanalysis of ubiquitous computing systems.* In *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pp. 1–4 (IEEE, 2016).

[55] J. Buchmann. *Introduction to cryptography* (Springer Science & Business Media, 2013).

[56] I. F. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*, vol. 265 (Cambridge University Press, 1999).

[57] A. Salomaa. *Public-key cryptography* (Springer Science & Business Media, 2013).

[58] V. Carlier, H. Chabanne, E. Dottax, and H. Pelletier. *Generalizing square attack using side-channels of an aes implementation on an fpga.* In *Field Programmable Logic and Applications, 2005. International Conference on*, pp. 433–437 (IEEE, 2005).

[59] J.-S. Coron, D. Naccache, and P. Kocher. *Statistics and secret leakage.* ACM Transactions on Embedded Computing Systems (TECS) **3**(3), 492 (2004).

[60] T. Messerges. *Using second-order power analysis to attack dpa resistant software.* In *Cryptographic Hardware and Embedded SystemsCHES 2000*, pp. 27–78 (Springer, 2000).

[61] J. Waddle and D. Wagner. *Towards efficient second-order power analysis.* In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pp. 1–15 (Springer, 2004).

[62] E. Brier, C. Clavier, and F. Olivier. *Correlation power analysis with a leakage model.* In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pp. 16–29 (Springer, 2004).

[63] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. *Examining smart-card security under the threat of power analysis attacks.* Computers, IEEE Transactions on **51**(5), 541 (2002).

[64] S. B. Örs, E. Oswald, and B. Preneel. *Power-analysis attacks on an fpga–first experimental results.* In *Cryptographic Hardware and Embedded Systems-CHES 2003*, pp. 35–50 (Springer, 2003).

[65] K. Gandolfi, C. Mourtel, and F. Olivier. *Electromagnetic analysis: Concrete results*. In *Cryptographic Hardware and Embedded SystemsCHES 2001*, pp. 251–261 (Springer, 2001).

[66] J.-J. Quisquater and D. Samyde. *Electromagnetic analysis (ema): Measures and counter-measures for smart cards*. In *Smart Card Programming and Security*, pp. 200–210 (Springer, 2001).

[67] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic. *Digital integrated circuits*, vol. 2 (Prentice hall Englewood Cliffs, 2002).

[68] H. Handschuh, P. Paillier, and J. Stern. *Probing attacks on tamper-resistant devices*. In *Cryptographic Hardware and Embedded Systems*, pp. 303–315 (Springer, 1999).

[69] D. Samyde, S. Skorobogatov, R. Anderson, and J.-J. Quisquater. *On a new way to read data from memory*. In *Security in Storage Workshop, 2002. Proceedings. First International IEEE*, pp. 65–69 (IEEE, 2002).

[70] S. P. Skorobogatov and R. J. Anderson. *Optical fault induction attacks*. In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pp. 2–12 (Springer, 2002).

[71] R. Anderson and M. Kuhn. *Low cost attacks on tamper resistant devices*. In *Security Protocols*, pp. 125–136 (Springer, 1997).

[72] N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Shamir. *Collision-based power analysis of modular exponentiation using chosen-message pairs*. In *Cryptographic Hardware and Embedded Systems–CHES 2008*, pp. 15–29 (Springer, 2008).

[73] P.-A. Fouque and F. Valette. *The doubling attack–why upwards is better than downwards*. In *Cryptographic Hardware and Embedded Systems-CHES 2003*, pp. 269–280 (Springer, 2003).

[74] L. Goubin. *A refined power-analysis attack on elliptic curve cryptosystems*. In *Public key cryptographyPKC 2003*, pp. 199–211 (Springer, 2003).

[75] P.-Y. Liardet and N. P. Smart. *Preventing spa/dpa in ecc systems using the jacobi form*. In *Cryptographic Hardware and Embedded SystemsCHES 2001*, pp. 391–401 (Springer, 2001).

[76] P.-A. Fouque, D. Réal, F. Valette, and M. Drissi. *The carry leakage on the randomized exponent countermeasure*. In *Cryptographic Hardware and Embedded Systems–CHES 2008*, pp. 198–213 (Springer, 2008).

[77] T. Hastie, R. Tibshirani, and J. Friedman. *Unsupervised learning* (Springer, 2009).

[78] M. Ciet and M. Joye. *(virtually) free randomization techniques for elliptic curve cryptography.* In *Information and Communications Security*, pp. 348–359 (Springer, 2003).

[79] M. Joye and S.-M. Yen. *The montgomery powering ladder.* In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pp. 291–302 (Springer, 2002).

[80] *IEEE standard specifications for public-key cryptography.* IEEE Std 1363-2000 pp. 1–228 (2000).

[81] V. S. Miller. *Use of elliptic curves in cryptography.* In H. Williams, ed., *Advances in Cryptology - CRYPTO 85 Proceedings*, vol. 218 of *Lecture Notes in Computer Science*, pp. 417–426 (Springer Berlin Heidelberg, 1986). URL http://dx.doi.org/10.1007/3-540-39799-X_31.

[82] N. Koblitz. *Elliptic curve cryptosystems.* Mathematics of computation **48**(177), 203 (1987).

[83] C. McIvor, M. McLoone, and J. McCanny. *Hardware elliptic curve cryptographic processor over* gf(p). Circuits and Systems I: Regular Papers, IEEE Transactions on **53**(9), 1946 (2006).

[84] P. Longa and A. Miri. *Fast and flexible elliptic curve point arithmetic over prime fields.* Computers, IEEE Transactions on **57**(3), 289 (2008).

[85] N. Koblitz. *Elliptic curve cryptosystems.* In *Proc. Math. Computation*, vol. 48, pp. 203–209 (American Mathematical Society., vol. 48, pp. 203-209, 1987). URL http://dl.acm.org/citation.cfm?id=18262.25413.

[86] H. Marzouqi, M. Al-Qutayri, and K. Salah. *An fpga implementation of nist 256 prime field ecc processor.* In *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*, pp. 493–496 (2013).

[87] N. Koblitz, A. Menezes, and S. Vanstone. *The state of elliptic curve cryptography.* Des. Codes Cryptography **19**(2-3), 173 (2000). URL http://dx.doi.org/10.1023/A:1008354106356.

[88] G. Sutter, J. Deschamps, and J. Imana. *Efficient elliptic curve point multiplication using digit-serial binary field operations.* IEEE Transactions on Industrial Electronics **60**(1), 217 (2013).

[89] A. G. Karegowda, M. Jayaram, and A. Manjunath. *Feature subset selection problem using wrapper approach in supervised learning.* International journal of Computer applications **1**(7), 13 (2010).

[90] P. A. Devijver and J. Kittler. *Pattern recognition: A statistical approach*, vol. 761 (Prentice-Hall London, 1982).

[91] K. Fukunaga. *Introduction to statistical pattern recognition* (Academic press, 2013).

[92] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*, vol. 1 (Basic Books, 1991).

[93] E. Oja. *Neural networks, principal components, and subspaces.* International journal of neural systems **1**(01), 61 (1989).

[94] T. D. Sanger. *Optimal unsupervised learning in a single-layer linear feedforward neural network.* Neural networks **2**(6), 459 (1989).

[95] G. Hospodar and V. I. V. J. Mulder, Benedikt. *Least squares support vector machines for side-channel analysis.* Center for Advanced Security Research Darmstadt pp. 99–104 (2011).

[96] C. M. Bishop *et al. Pattern recognition and machine learning*, vol. 1 (springer New York, 2006).

[97] J. Karhunen and J. Joutsensalo. *Generalizations of principal component analysis, optimization problems, and neural networks.* Neural Networks **8**(4), 549 (1995).

[98] R. Battiti. *Using mutual information for selecting features in supervised neural net learning.* Neural Networks, IEEE Transactions on **5**(4), 537 (1994).

[99] I. Jolliffe. *Principal component analysis* (Wiley Online Library, 2005).

[100] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo. *Supervised feature selection via dependence estimation.* In *Proceedings of the 24th international conference on Machine learning*, pp. 823–830 (ACM, 2007).

[101] L. I. Smith. *A tutorial on principal components analysis.* Cornell University, USA **51**, 52 (2002).

[102] Y. Li, M. Dong, J. Hua, *et al. A gaussian mixture model to detect clusters embedded in feature subspace.* Communications in Information & Systems **7**(4), 337 (2007).

[103] J. Weston, C. Watkins, *et al. Support vector machines for multi-class pattern recognition.* In *ESANN*, vol. 99, pp. 219–224 (1999).

[104] K. Crammer and Y. Singer. *On the algorithmic implementation of multiclass kernel-based vector machines.* The Journal of Machine Learning Research **2**, 265 (2002).

[105] Y. Lee, Y. Lin, and G. Wahba. *Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data.* Journal of the American Statistical Association **99**(465), 67 (2004).

[106] E. Monfrini and Y. Guermeur. *A quadratic loss multi-class svm.* arXiv preprint arXiv:0804.4898 (2008).

[107] C. Cortes and V. Vapnik. *Support-vector networks.* Machine learning **20**(3), 273 (1995).

[108] K. Jonsson, J. Kittler, Y. Li, and J. Matas. *Support vector machines for face authentication.* Image and Vision Computing **20**(5), 369 (2002).

[109] J. Lu, K. Plataniotis, and A. Venetsanopoulos. *Face recognition using feature optimization and ν-support vector learning.* In *Neural Networks for Signal Processing XI, 2001. Proceedings of the 2001 IEEE Signal Processing Society Workshop*, pp. 373–382 (IEEE, 2001).

[110] N. Bassiou, C. Kotropoulos, T. Kosmidis, and I. Pitas. *Frontal face detection using support vector machines and back-propagation neural networks.* In *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 1, pp. 1026–1029 (IEEE, 2001).

[111] E. M. dos Santos and H. M. Gomes. *Appearance-based object recognition using support vector machines.* In *sibgrapi*, p. 399 (IEEE, 2001).

[112] Z. Li, Z. Weida, and J. Licheng. *Radar target recognition based on support vector machine.* In *Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on*, vol. 3, pp. 1453–1456 (IEEE, 2000).

[113] C. Nakajima, M. Pontil, and T. Poggio. *People recognition and pose estimation in image sequences.* In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 4, pp. 189–194 (IEEE, 2000).

[114] J. Platt *et al. Fast training of support vector machines using sequential minimal optimization.* Advances in kernel methodssupport vector learning **3** (1999).

[115] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. *Improvements to platt's smo algorithm for svm classifier design.* Neural Computation **13**(3), 637 (2001).

[116] B. Gutschoven and P. Verlinde. *Multi-modal identity verification using support vector machines (svm).* In *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, vol. 2, pp. THB3–3 (IEEE, 2000).

[117] F. E. Tay and L. Cao. *Modified support vector machines in financial time series forecasting.* Neurocomputing **48**(1), 847 (2002).

[118] W. Wang, Z. Xu, W. Lu, and X. Zhang. *Determination of the spread parameter in the gaussian kernel for classification and regression.* Neurocomputing **55**(3), 643 (2003).

[119] Z. Xu, M. Dai, and D. Meng. *Fast and efficient strategies for model selection of gaussian support vector machine.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **39**(5), 1292 (2009).

[120] S.-F. Yuan and F.-L. Chu. *Support vector machines-based fault diagnosis for turbo-pump rotor.* Mechanical Systems and Signal Processing **20**(4), 939 (2006).

[121] P. Konar and P. Chattopadhyay. *Bearing fault detection of induction motor using wavelet and support vector machines (svms).* Applied Soft Computing **11**(6), 4203 (2011).

[122] C. Campbell. *An introduction to kernel methods.* Studies in Fuzziness and Soft Computing **66**, 155 (2001).

[123] V. Vapnik. *The nature of statistical learning theory* (springer, 2000).

[124] B. Zhao, Y. Liu, and S.-W. Xia. *Support vector machine and its application in handwritten numeral recognition.* In *icpr*, p. 2720 (IEEE, 2000).

[125] U. H.-G. Kreßel. *Pairwise classification and support vector machines.* In *Advances in kernel methods*, pp. 255–268 (MIT Press, 1999).

[126] B. Heisele, P. Ho, and T. Poggio. *Face recognition with support vector machines: Global versus component-based approach.* In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, pp. 688–694 (IEEE, 2001).

[127] F. Lauer and Y. Guermeur. *MSVMpack: a multi-class support vector machine package.* Journal of Machine Learning Research **12**, 2269 (2011). http://www.loria.fr/~lauer/MSVMpack.

[128] K. Hornik, M. Stinchcombe, and H. White. *Multilayer feedforward networks are universal approximators.* Neural networks **2**(5), 359 (1989).

[129] M. D. Richard and R. P. Lippmann. *Neural network classifiers estimate bayesian a posteriori probabilities.* Neural computation **3**(4), 461 (1991).

[130] D. W. Marquardt. *An algorithm for least-squares estimation of nonlinear parameters.* Journal of the Society for Industrial & Applied Mathematics **11**(2), 431 (1963).

[131] P. M. Williams. *Bayesian regularization and pruning using a laplace prior.* Neural computation **7**(1), 117 (1995).

[132] J. E. Dennis Jr and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, vol. 16 (Siam, 1996).

[133] M. Riedmiller and H. Braun. *A direct adaptive method for faster backpropagation learning: The rprop algorithm.* In *Neural Networks, 1993., IEEE International Conference on*, pp. 586–591 (IEEE, 1993).

[134] M. F. Møller. *A scaled conjugate gradient algorithm for fast supervised learning.* Neural networks **6**(4), 525 (1993).

[135] M. J. Powell. *Restart procedures for the conjugate gradient method.* Mathematical programming **12**(1), 241 (1977).

[136] M. T. Hagan, H. B. Demuth, M. H. Beale, *et al. Neural network design* (Pws Pub. Boston, 1996).

[137] R. Battiti. *First-and second-order methods for learning: between steepest descent and newton's method.* Neural computation **4**(2), 141 (1992).

[138] M. Bowling and M. Veloso. *Multiagent learning using a variable learning rate.* Artificial Intelligence **136**(2), 215 (2002).

[139] C. M. Bishop. *Pattern recognition and machine learning* (springer, 2006).

[140] D. Svozil, V. Kvasnicka, and J. Pospichal. *Introduction to multi-layer feed-forward neural networks.* Chemometrics and intelligent laboratory systems **39**(1), 43 (1997).

[141] D. F. Specht. *Probabilistic neural networks.* Neural networks **3**(1), 109 (1990).

[142] P. Raghu and B. Yegnanarayana. *Supervised texture classification using a probabilistic neural network and constraint satisfaction model.* Neural Networks, IEEE Transactions on **9**(3), 516 (1998).

[143] A. Zaknich. *A vector quantisation reduction method for the probabilistic neural network.* In *Neural Networks, 1997., International Conference on*, vol. 2, pp. 1117–1120 (IEEE, 1997).

[144] D. Badde, A. Gupta, and V. K. Patki. *Cascade and feed forward back propagation artificial neural network models for prediction of compressive strength of ready mix concrete.* IOSR Journal of Mechanical and Civil Engineering **3**, 1 (2013).

[145] V. Schetinin. *An evolving cascade neural network technique for cleaning sleep electroencephalograms.* arXiv preprint cs/0504067 (2005).

[146] H. Demuth and M. Beale. *Neural network toolbox users guide* (2000).

[147] A. Gersho. *Asymptotically optimal block quantization.* Information Theory, IEEE Transactions on **25**(4), 373 (1979).

[148] P. L. Zador. *Asymptotic quantization error of continuous signals and the quantization dimension.* Information Theory, IEEE Transactions on **28**(2), 139 (1982).

[149] T. Kohonen. *An introduction to neural computing.* Neural networks **1**(1), 3 (1988).

[150] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification* (John Wiley & Sons, 2012).

[151] M. L. Minsky and S. A. Papert. *Perceptrons - Expanded Edition: An Introduction to Computational Geometry* (MIT press Boston, MA:, 1987).

[152] T. Kohonen. *Statistical pattern recognition revisited* (1990).

[153] T. Kohonen. *Improved versions of learning vector quantization.* In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pp. 545–550 (IEEE, 1990).

[154] C. F. Carobbi, L. M. Millanta, and L. Chiosi. *The high-frequency behavior of the shield in the magnetic-field probes.* In *Electromagnetic Compatibility, 2000. IEEE International Symposium on*, vol. 1, pp. 35–40 (IEEE, 2000).