Investigating Power and Fault Analysis with Specific Application to Bilinear Pairings

Claire Whelan

B.Sc.

A Dissertation submitted in fulfilment of the

requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University

Faculty of Engineering and Computing,

School of Computing

Supervisor: Dr. Michael Scott

April, 2007

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed Claire Whelan

Student ID 53144678

Date 10th Sept. 2007

Thesis Outline

Abstract

The security of bilinear pairings against implementation attacks such as side channel and fault attacks is largely an uncharted area of research. Apart from one publication on the topic, coverage of this area is non-existent. Armed with the fact that the number of applications based on bilinear pairings is ever-increasing, the bilinear pairing algorithms themselves are constantly being enhanced and optimised such that they are commercially viable, and the fact that the current research on elliptic curve primitives is not applicable to bilinear pairings, makes this a vital topic for further investigation and analysis.

This research aims to begin to fill this void. Along with addressing some of the more subtle aspects of implementation attacks, this research presents an investigation into the security of bilinear pairings against implementation attacks. Specifically, the process of performing the data analysis phase of a Side Channel Attack (SCA) is analysed. A theoretical fault attack on the Digital Signature Algorithm (DSA) is examined and implemented in practice. A number of candidate bilinear pairing algorithms are assessed for vulnerability to the SCA, first-order power analysis, which passively monitors the power consumption of a device. Furthermore, a number of candidate bilinear pairing algorithms are assessed for vulnerability to fault analysis, which seeks to actively disrupt the normal execution of an algorithm.

Our principal results can be summarised as follows: We suggest computational improvements to the Differential Power Analysis (DPA) data analysis process, which can reduce the number of operations by up to 97%. We demonstrate how a theoretical attack on the DSA using lattice reduction can be executed in practice with the aid of a glitch attack. We propose a novel SCA technique to attack various finite field operations. This attack involves analysing the structural evolution of finite field operations and is based on Correlation Power Analysis (CPA), which is a form of first-order power analysis. We examine the Tate, Ate and η_T pairing for vulnerability to first-order power analysis and

discover that given certain parameter choice, the Tate and Ate pairing can provide options for minimising an attack, whereas the η_T pairing provides no such options and can be attacked from all parameter positions. We investigate the existence of opportunistic faults on the Weil, Tate and η pairing and discover two types of fault attacks that can be successfully applied to the Weil and η pairing to reveal the secret key. This weakness is attributed to the absence or simplicity of the final exponentiation employed, highlighting the fact that the final exponentiation is a vital operation in bilinear pairing computation and in particular adds a layer of protection to pairings. This fact is further compounded in the proof that the Tate pairing is immune to such fault attacks. Finally, we provide recommendations based on our findings for secure bilinear pairing implementation in terms of power and fault analysis.

Acknowledgements

There are many people who I would like to express my sincere thanks to for making the last three and a half years possible.

Firstly, I would like to thank my supervisor Mike Scott for getting me through these past years. You have been a tremendous source of inspiration, direction and expertise.

I would like to thank all the people that I have worked with both inside and outside of DCU. I would like to thank my co-authors David Naccache, Mike Tunstall, Phong Nguyễn, Hagai Bar El, Hamid Choukri and Rob McEvoy. In particular, I would like to acknowledge David Naccache who was instrumental in the first part of my Ph.D., thank you for your interest, enthusiasm and guidance. Also, thanks for funding a number of trips and for making it possible for me to spend time in Gemplus' Applied Research & Security Centre.

I would like to acknowledge my sponsors, the Irish Research Council for Science, Engineering and Technology (IRCSET), without whom this research and experience would not have been possible.

I would like to thank all my family and friends for their constant support and encouragement. A special thanks to Marie and Sean for always supporting and encouraging me in

all that I do.

Finally to Tom, you have been there for both the highs and lows of the past three years. Thank you for your unwavering support, love and belief in me.

List of Publications

A number of publications form the basis for this research.

The Sorcerers Apprentice Guide to Fault Attacks was co-authored by Michael Tunstall, Hamid Choukri, David Naccache and Hagai Bar-El. It was first published at the Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), which was run in association with the International Conference on Dependable Systems and Networks (DSN), in Florence, Italy, June 28 - July 1, 2004. This work was later published in the Proceedings of the IEEE, Special Issue on Cryptography and Security, Volume 96, Number 2, in February, 2006.

Computational Improvements to Differential Side Channel Analysis was co-authored by Michael Tunstall and David Naccache. It was published at NATO Security through Science Series D: Information and Communication Security, Volume 2, IOS Press, in January, 2006.

Experimenting with Faults Lattices and the DSA was co-authored by Michael Tunstall, Phong Nguyễn and David Naccache. It was published at the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC), Volume 3386 of Lecture Notes in Computer Science, in January, 2005.

Side Channel Analysis of Practical Pairings: Which Path is More Secure was co-authored by Mike Scott. It was published at the International Conference on Cryptology in Vietnam (VIETCRYPT), Volume 4341 of Lecture Notes in Computer Science, which took place in Hanoi in October 2006.

The Importance of the Final Exponentiation in Pairings when Considering Fault Attacks was co-authored by Mike Scott. It was recently published at the First International Conference on Pairing-Based Cryptography (Pairing 2007), Volume 4575 of Lecture Notes in Computer Science, which took place in Tokyo, Japan, in July 2007.

List of Figures

List of Algorithms

List of Tables

Glossary

AES	Advanced Encryption Standard
ASIC	Application Specific Integrated Circuit
BDHP	Bilinear Diffie-Hellman Problem
BKLS	Barreto Kim Lynn Scott
BKZ	Block Korkine-Zolotarev
CPA	Correlation Power Analysis
CPI	(Number of clock) Cycles Per Instruction
CPU	Central Processing Unit
CRT	Chinese Remainder Theorem
DEMA	Differential Electromagnetic Analysis
DES	Data Encryption Standard
DFA	Differential Fault Analysis
DHP	Diffie-Hellman Problem
DPA	Differential Power Analysis
DSA	Digital Signature Algorithm
DSCA	Differential Side Channel Attack
DSS	Digital Signature Standard
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Log Problem
ECDSA	Elliptic Curve Digital Signature Algorithm

EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electromagnetic
FPGA	Field Programmable Gate Array
GCD	Greatest Common Divisor
GF	Galois Field
IBE	Identity Based Encryption
I/O	Input Output
MHz	MegaHertz
NAF	Non Adjacent Form
NTL	(Shoup's) Number Theory Library
NVM	Non Volatile Memory
PCB	Printed Circuit Board
PBC	Pairing Based Cryptography
PCSC	Personal Computer to Smart Card
PKG	Private Key Generator
QNR	Quadratic Non Residue
RAM	Random Access Memory
ROM	Read-Only Memory
SCA	Side Channel Attack/Analysis
RSA	Rivest, Shamir and Adleman: Inventors of the public key algorithm
SEMA	Simple Electromagnetic Analysis
SFA	Simple Fault Analysis
SNR	Signal to Noise Ratio
SPA	Simple Power Analysis
XOR	Bitwise Exclusive Or
$\oplus$	Bitwise Exclusive Or

Chapter 1

Introduction

1.1 Introduction

Cryptanalysis is the process of attacking a cryptosystem with the ultimate goal of identifying weaknesses such that retrieval of information about the secret key is possible. The security of any cryptosystem is based on a mathematical problem that is hard to solve. Hence, traditionally, cryptanalysis focused on the mathematical structure of the cryptosystem and so mathematical attacks were used to uncover weaknesses. Cryptosystems resistant to this analysis were generally deemed to be strong cryptosystems. In principle, the cryptosystem may be mathematically sound. However, mathematical resistance to cryptanalysis does not guarantee security. When the cryptosystem is implemented in a realistic and generally hostile environment it can be open to unforeseen and unanticipated attacks, which bypass the robust mathematics on which the cryptosystem is based. An extremely powerful branch of cryptanalysis, referred to as the field of implementation attacks, covers a range of attacks that exploit features of the implementation, making ordinarily secure algorithms insecure. This branch of cryptanalysis has become increasingly relevant due to the technological advances in the devices on which the cryptosystem can be implemented, making such attacks more of a threat.

1.2 Overview of the Research Area

The purpose of this section is to give a history of the developments in the field of implementation attacks and bilinear pairings. Since both fields have until recently evolved in parallel, they will be described separately. Both fields have been well studied and so only the most notable and relevant developments will be mentioned. The Side Channel Cryptanalysis Lounge [1], any of the proceedings of Cryptographic Hardware and Embedded Systems (CHES) [60], or Fault Diagnosis and Tolerance in Cryptography (FDTC) [2], can be referred to, to access the vast body of papers in the field of implementation attacks. Similarly, Baretto's Pairing Based Crypto Lounge [12] can be referred to for results in the domain of bilinear pairings.

1.2.1 Implementation Attacks

Implementation attack is an umbrella term used to describe cryptanalytic attacks which exploit various implementation characteristics. There are a number of physical properties that can be exploited when considering a specific implementation on a specific device. One of these properties is the inescapable physical manifestations inherent in any device. Attacks which exploit observable characteristics of this form are called **Side Channel Attacks (SCA)**. Another property is the fact that all computational devices are prone to error and malfunction given certain circumstances. Attacks which force a device to work outside its normal working conditions with the objective of inducing abnormal behaviour are referred to as **Fault Attacks**.

The act of breaking a cryptosystem via implementation weaknesses has been known for decades. In 1965, the British intelligence agency MI5 eavesdropped on the Egyptian embassy in London by placing a microphone near the rotors of their Hagelin cipher machines. This attack, recounted in the novel "Spy Catcher" [139] by Wright, a chief scientist with MI5, is just one account of an implementation attack that relies on exploiting information unintentionally leaked by a device. The rapid advancement of technology

over the years and in particular the development of portable embedded devices known as smart cards [29], has heightened the relevance of implementation attacks. This is because implementation attacks can be easily launched on the smart card, since physical measurements can be taken and faults can be injected with minimal effort.

The application of SCA to attacking embedded devices began in earnest in 1996. In August 1996, Kocher [64] introduced a timing attack on a smart card implementing various public key algorithms. By measuring the time that it took to perform certain operations, one could deduce information about the private key.

Then in September 1996, in a press release Boneh, DeMillo and Lipton announced the first account of fault analysis. The effect of faults on electronic systems has been studied since the 1970s when it was noticed that radioactive particles produced by elements naturally present in packaging material [78] caused faults in chips. Specifically, Uranium-235, Uranium-238 and Thorium-230 residues present in the packaging decay to Lead-206 releasing α particles. These particles create a charge in sensitive chip areas causing bits to flip, affecting a chip's behavior. Boneh, DeMillo and Lipton realised the implications of such faults for the cryptographic world when they described a new type of cryptanalytic attack which focused on attacking specific algebraic properties of modular arithmetic. This idea, later published in Eurocrypt 1997 [21], while only theoretical at the time, sparked interest into both the search for possible opportunistic faults in existing cryptosystems and the need to find a mechanism by which to actually induce these faults. Some of the first researchers to the field were Biham and Shamir, who in 1997 presented a theoretical fault attack applicable to almost any secret key cryptosystem [19].

In 1998, Kocher, Jaffe and Jun, further signaled the weakness of smart cards. Kocher et al. demonstrated how measuring the variation in power consumption of the hardware during execution could be used to deduce information about operations being performed. In a technical report first released [65] and later published [66], Kocher et al. defined two forms of power analysis, namely Simple Power Analysis (SPA) and Differential Power

Analysis (DPA), both of which were applied to the Data Encryption Standard (DES) [96]. SPA is the most straightforward form of power analysis, and involves directly examining the power consumption acquired. SPA generally relies on the power consumption of a single execution and can reveal various characteristics of an algorithm, such as the instruction sequences being executed. The application of SPA to DES showed that the 16 individual rounds of the DES algorithm could be easily distinguished in the power consumption. DPA on the other hand, is a more complex form of power analysis, involving the acquisition of power consumption relating to various executions together with statistical analysis techniques. DPA exploits variations in power consumption that are correlated to the data values being manipulated. These variations are typically much smaller than those associated with different instruction sequences, and may be obfuscated by noise and other factors. Statistical methods are used on a collection of power consumption acquisitions in order to reduce the noise and amplify data dependant power consumption. The application of DPA to DES allowed the extraction of the secret key.

The apparent strength of these attacks sparked avid interest in the research community. This in turn has led to the development of attack variants and countermeasures, which have since shaped the development of embedded devices. In particular, aspects relating to the development of both hardware and software countermeasures, the development of other types of implementation attacks, the application of implementation attacks to different cryptosystems and algorithms, and the application of such attacks to other platforms, have received great attention from both the cryptographic and smart card community.

In 1999 a number of other cryptographic algorithms were shown also to be vulnerable to power analysis. At that time the official call for candidate algorithms for the Advanced Encryption Standard (AES), the intended successor to DES, was underway. In 1999, Chari, Jutla, Rao and Rohatgi [27] demonstrated that all AES candidate algorithms were in some way vulnerable to power analysis. The same year power analysis was also applied

to modular exponentiation [86] and elliptic curve cryptosystems [31]. Fault attacks also progressed in 1999 when Kommerling and Kuhn [69] reported that a glitch attack on the external power supply and clock supply lines were the most useful fault attacks to inject in practice.

Immediately after Kocher et al.'s [66] breakthrough, countermeasures to thwart these attacks and render the power consumption useless were investigated. One such countermeasure, known as masking, is where intermediate computations are handled under a probabilistic form to defeat statistical correlation. In 2000, masking techniques based on boolean and arithmetic functions were suggested by Coron and Goubin [32]. However, in the same year Messerges [84] developed a more powerful variant of DPA, which defeated masking techniques and so-called DPA resistant implementations. These attacks were named second or high-order attacks, since they correlated the power consumption at multiple steps during a single computation. The development of this branch of power analysis has since led to the original DPA of [66] being referred to as first-order DPA.

Another significant breakthrough in the field of SCA came in 2000 when the idea of using electromagnetic (EM) emanations as the side channel was proposed by Quisquater and Samyde [111]. The attacks Simple Electromagnetic Analysis (SEMA) and Differential Electromagnetic Analysis (DEMA) were also proposed as the EM analog to SPA and DPA. Concrete results of these attacks were later demonstrated by Gandolfi, Mourtel and Olivier [42]. The usefulness of EM emanations was postulated for years, but this was the first time that concrete results were documented.

In 2002, Chari, Rao and Rohatgi [28] developed another variant of the original DPA, referred to as a template attack. Template attacks were developed for scenarios where there is access to only a single power consumption acquisition. In addition to availing of the power consumption information that leaks naturally from a device, a template attack allows the adversary access to a device similar to the target. This allows the adversary to learn about various characteristics of the target device. In particular, the power con-

sumption characteristics of instructions and noise characteristics of the device can be constructed. This is usually achieved by determining the probability distribution of the power consumption of certain instructions. Also in 2002, a new class of fault induction attacks was revealed by Skorobogatov and Anderson [124], which was a form of semi-invasive optical fault attack which allowed control of specific registers.

In 2004, Asonov and Agrawal [8] developed acoustic cryptanalysis, which uses the sound produced during the computation as the side channel. The authors of [8] announced that computer keystrokes and keypads used on telephones and automatic teller machines (ATMs) are vulnerable to attacks based on differentiating the sound produced by different keystrokes. By analysing recorded sounds, they were able to determine the text of data being entered. In the same year, Shamir and Tromer [122] further demonstrated the usefulness of acoustic cryptanalysis when they demonstrated that it was possible to conduct a timing attack against a CPU performing cryptographic operations. This attack was based on analysing variations in the CPU's humming noise.

Also in 2004, Brier, Clavier and Olivier [24] defined another variant of first-order DPA, which they called Correlation Power Analysis (CPA). The authors of [24] highlighted that there were some shortcomings with first-order DPA [66]. These shortcomings, which were manifested as inaccuracies in the results witnessed, were on account of the fact that the assumptions made about power consumption were incomplete. Brier et al. proposed a new model which takes more factors about what constitutes power consumption into account.

In 2005, Percival [109] and Bernstein [17] described a type of timing attack based on monitoring the number of cache misses over a period of time. Then in 2006 Aciicmez, Koç and Seifert [3] developed another form of implementation attack, known as branch prediction analysis, that was shown to be successful against an openSSL RSA implementation executing on a popular commodity PC. Branch prediction analysis employs a spy-process to monitor the different branches that the CPU's branch predictor takes,

which can be analysed to reveal the secret.

Throughout the years, since 1996, the field of implementation attacks has exploded. A vast body of papers relating to the attack [132, 135] and defence [133, 59] of modular exponentiation, the attack [102] and defence [6, 129, 43] of AES, the attack [9] and defense [134] of ECC, the application of SCA to an FPGA [101] and ASIC [102], the advancement of second-order power analysis [131, 57], template attacks [112, 5, 103] and CPA [72], and the development of fault injection techniques [11] have appeared.

To this day, the research and race for stronger attacks, more complete countermeasures and attacks to overcome these countermeasures continues. One particularly important topic in this field is the application of such attacks to novel cryptosystems. A relatively new phenomena in cryptography is the construction of cryptosystems based on bilinear pairings.

1.2.2 Bilinear Pairings

In 1985, Koblitz and Miller independently discovered the usefulness of elliptic curves over finite fields for cryptography. They realised that discrete logarithm based cryptosystems might provide better security when based on the group of points on an elliptic curve rather than the conventional multiplicative group of a finite field. This realisation had the added benefit that elliptic curve cryptosystems could provide similar security levels for much shorter key length. Since then, there has been a vast amount of research into the development of cryptosystems based on elliptic curves.

A bilinear pairing is a function that takes two elliptic curve points and outputs an element in some multiplicative finite field. Furthermore, a pairing satisfies some special properties, the most important being bilinearity. The two founding pairings are the Weil and Tate Pairing. Bilinear pairings made their first appearance into the realm of cryptography, when it was discovered that they could be used for cryptanalytic purposes. In 1993, Menezes, Okamoto and Vanstone [81] discovered that the Weil pairing could be

used to attack discrete logarithm based systems on a certain class of elliptic curves. This attack is known as the MOV reduction. One year later, Frey and Ruck [40] defined a similar attack, the FR-reduction, which used the Tate pairing. This cryptanalytic use was the only known application of pairings until 2000, when Joux [55] and Sakai, Ohgishi and Kasahara [113], showed how bilinear pairings could be used constructively to build cryptographic protocols with unique properties. Since then, there has been avid interest in the field of bilinear pairings, leading to the fruitful development of many new applications based on pairings [12]. The field of pairing based cryptography has evolved in two main directions. Namely, the development of pairing based protocols and the development of algorithms to efficiently compute a bilinear pairing.

Undoubtedly the most striking application of pairings is the realisation of Identity Based Cryptography (also known as Identity Based Encryption (IBE)). IBE is a variant of public key cryptography originally proposed by Shamir in 1984 [121], where the user's public key is derived from their identity. The first concrete implementation of IBE however, did not appear until 2001 when Boneh and Franklin [22] proposed an IBE scheme based on bilinear pairings. The breakthrough of this seminal work opened the floodgates for pairings to be used in many applications. For example, a myriad of key agreement schemes, short signature schemes and encryption schemes have been proposed. A survey of the many schemes can be found in [36].

Alongside the rapid development of pairing based protocols, algorithms to efficiently compute bilinear pairings have been continually developed and advanced. Since the bilinear pairing lies at the heart of these schemes, and is generally the most expensive computation in the scheme, the efficiency of the pairing affects both whether such schemes will be adopted and where it will be used. Two methods to compute a bilinear pairing have been put forward. The first method was proposed in an unpublished manuscript by Miller [88] in 1986. This method is based on divisor theory [136] and to date has been the foundation from which all pairing algorithms have been constructed. In 2006, an alternative

technique was proposed by Stange [126]. Stange's method is based on elliptic nets, and will not be addressed in this thesis. Numerous optimisations to Miller's algorithm have been made, and in particular in the context of the Tate pairing, the time to compute cryptographically secure bilinear pairings has dramatically decreased from several minutes [79] to only a few milliseconds [14, 118]. In addition, a number of variants of the Tate pairing, namely the η, η_T and the Ate pairing, have been developed. These bilinear pairings are becoming increasingly fast and efficient, yielding pairing based protocols that are viable competitors to more established public key cryptosystems like RSA.

1.3 Motivation for this Research

Due to the threat of implementation attacks, analysis of all cryptographic algorithms for vulnerability to implementation attacks is necessary. The smart card is the most notorious device for implementation attacks. Hence, analysis of cryptographic algorithms which may potentially be implemented on a smart card, is an urgent priority.

As already mentioned, the popularity of bilinear pairings has seen the development of a plethora of pairing based protocols. One barrier to the adoption of pairings was their inability to compete with alternative cryptographic algorithms on resource constrained devices. However, the development of increasingly efficient pairing implementations has eliminated this barrier. In 2006, Scott, Costigan and Abdulwahab [120] demonstrated that three bilinear pairing algorithms were efficiently computable on a smart card. These bilinear pairings were namely the BKLS algorithm for the Tate pairing [14], the Ate pairing [50], and the BGOhES algorithm for a truncated version of the η pairing, the η_T pairing [13]. An excerpt from [120] presenting computation times and comparisons with RSA decryption is given at the end of this chapter in Table 1.1. Listed in this table are results for the number of clock cycles required to compute three types of pairings, the number of clock cycles per instruction (CPI), and the time in seconds. This development,

while advancing the field of pairing based cryptography, raises some alarming open issues about the security of pairings on such devices. Hence, there are compelling reasons to investigate bilinear pairings for vulnerability to implementation attacks.

The majority of implementation attacks on elliptic curve cryptosystems have studied the algorithm for point scalar multiplication [9, 30], where the scalar is the secret parameter of interest. For example, the Elliptic Curve Digital Signature Algorithm (ECDSA) requires the computation of $[s]P$, where P is a publicly known elliptic curve point and s is the secret scalar of interest. In cryptosystems based on pairings, however, the secret is not typically a scalar but an elliptic curve point. Hence, previous attacks defined are not applicable to bilinear pairings, requiring attack strategies and consequently defensive strategies to be rethought and re-evaluated.

To date, one publication addressing power and fault analysis of bilinear pairings exists. In 2004, Page and Vercauteren released the preprint [105], later published in [107], which assessed the general frame of bilinear pairings for vulnerability to DPA. They also assessed the Duursma-Lee algorithm [37] for the Tate pairing and Kwon-BGOS algorithm [71] for the η pairing for vulnerability to one type of fault, namely a fault which alters the number iterations in a loop. However, multiple classes of implementation attacks and various algorithms for pairings exist. Implementation attacks are by their nature implementation specific, exploiting features of the implementation. Therefore, assessing the general frame of a pairing while necessary and useful, is incomplete. Specific implementations of pairings must be assessed, and in particular, efficient implementations that may be implemented on smart cards must be assessed so that potential vulnerabilities can be anticipated and appropriate countermeasures deployed.

In this thesis, implementation attacks, and in particular first-order power and fault attacks, are investigated. The implications of first-order power and fault attacks with respect to bilinear pairings are examined. Specifically, the bilinear pairings algorithms that Scott et al. [120] proved to be efficiently computable on a smart card are examined for

vulnerability to first-order power analysis. The bilinear pairings, the BKLS algorithm for the Tate pairing [14], the Ate pairing [50], and the BGOhES algorithm for the η_T pairing [13], are three of the fastest implementations at present, and so are the most viable candidates for smart card adoption. This consequently makes them the most notable targets for implementation attacks. An investigation into the vulnerability of each of these algorithms to first-order power analysis is presented. The analysis is performed in theory, using empirical knowledge about first-order power analysis. The bilinear pairings, the η pairing of Galbraith et al. [41], the Weil pairing [117] and the BKLS algorithm for the Tate pairing [14], are investigated for vulnerability to fault attacks. Two of these bilinear pairings are not the most efficient algorithms but are chosen based on their characteristics. In particular, the three algorithms are chosen based on the varying complexity of their final exponentiation, since as will be demonstrated, this is a vital feature in either allowing or disallowing fault attacks. From our analysis, it will be demonstrated that there are certain characteristics that can either weaken or strengthen a bilinear pairing against power and fault attacks. Recommendations based on our findings will be given, which describe features for secure bilinear pairing implementation in the context of implementation attacks.

The structure of this thesis is as follows. Chapter 2 provides an overview of power and fault analysis. In this chapter, the necessary background, terminology and notation for the thesis are provided. The process of performing a power analysis attack and some of the various types of attacks mentioned in Section 1.2.1 will also be described. Chapter 3 provides an overview of bilinear pairings. In this chapter, the necessary concepts and definitions relating to finite field arithmetic, elliptic curve cryptography and bilinear pairings are given. The bilinear pairing algorithms that are required for the investigations described in the thesis are also provided. Chapters 2 and 3 also serve as a literature review, detailing the most notable advances in each field. The research contribution of this thesis is split into three chapters. Chapter 4 examines power and fault analysis in greater detail.

Specifically, two aspects are presented. Namely, a fault attack on the Digital Signature Algorithm (DSA) is described and computational improvements to the data analysis phase of DPA are proposed. Chapter 5 explores bilinear pairings for vulnerability to power analysis. It is shown how the core operations in pairing computation are vulnerable to power analysis based on analysing the structural evolution of various finite field operations. Analysis of these operations is carried out based on empirical knowledge about power analysis. The consequences of these findings are subsequently realised when it is shown how three pairing algorithms, namely the Tate, η_T and Ate pairing, can be attacked with power analysis. Chapter 6 explores bilinear pairings for vulnerability to fault analysis. It is shown how certain characteristics of bilinear pairings weaken the pairing and make it more susceptible to a fault attack. Specifically, fault attacks are demonstrated on the Weil and η pairing. Finally, the thesis is concluded in Chapter 7. The contributions of this thesis are summarised, recommendations for secure bilinear pairing implementation are given and areas for further exploration are discussed in this final chapter.

	9 MHz		
	Clock Cycles Required	CPI	Time in Seconds
$E(\mathbb{F}_p)$ Tate Pairing	9104450	1.17	1.01
$E(\mathbb{F}_{2^{379}})$ η_T Pairing	4311454	1.16	0.48
$E(\mathbb{F}_p)$ Ate Pairing	10860479	1.33	1.21
RSA Decryption	4740271	1.08	0.53
	20.57 MHz		
	Clock Cycles Required	CPI	Time in Seconds
$E(\mathbb{F}_p)$ Tate Pairing	9755457	1.26	0.47
$E(\mathbb{F}_{2^{379}})$ η_T Pairing	4590712	1.24	0.22
$E(\mathbb{F}_p)$ Ate Pairing	12207440	1.50	0.59
RSA Decryption	4880323	1.12	0.24
	36 MHz		
	Clock Cycles Required	CPI	Time in Seconds
$E(\mathbb{F}_p)$ Tate Pairing	10467010	1.35	0.29
$E(\mathbb{F}_{2^{379}})$ η_T Pairing	4891054	1.32	0.14
$E(\mathbb{F}_p)$ Ate Pairing	13621597	1.67	0.38
RSA Decryption	5072415	1.16	0.14

Table 1.1: Timings for pairings on a MIPS32 smart card [120], performed using an FPGA based emulator. Note the p is a 512 bit prime for the Tate pairing and a 256 bit prime for the Ate pairing.

Chapter 2

Power and Fault Analysis

2.1 Introduction

Implementation attacks can be classified into various categories. An implementation attack can be invasive or non-invasive, where the physical body of the target device can be tampered with, or go untouched. They can also be passive or active, in that the physical emanations that naturally leak are simply measured or certain reactions are purposely provoked. Specifically, this dissertation will consider two types of implementation attacks. Firstly, attacks which exploit the power consumption of a device and so are considered a type of passive, non-invasive SCA. Secondly, both invasive and non-invasive fault attacks which seek to disrupt the normal execution of a device are considered. The purpose of this chapter is to introduce and describe both of these types of implementation attacks.

Firstly, power analysis is examined. In particular, the stages of performing a power analysis attack, the different techniques of power analysis and common countermeasures developed to thwart power analysis are described. For demonstrative purposes, an example of Correlation Power Analysis (CPA) attacking the Advanced Encryption Standard (AES) is presented.

Secondly, fault analysis is examined. In particular, the process of performing a fault

analysis attack is discussed. This involves looking at the mechanisms for fault creation and consequently the effects of such faults. Common countermeasures developed to thwart fault analysis are also described. For demonstrative purposes, an example of a fault attack on RSA using the Chinese Remainder Theorem (CRT) is given.

Section 2.3 of this chapter is an excerpt from the paper "The Sorcerer's Apprentice Guide to Fault Attacks". This paper was performed in collaboration with Hagai Bar El, Hamid Choukri, Mike Tunstall and David Naccache. It was published originally in the first workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC) [10] in 2004, and later in a special proceedings of the IEEE on Security and Cryptography in 2006 [11].

2.2 Power Analysis

Power analysis is a type of SCA which exploits the power consumption. In this section the process of performing power analysis, and the types of power analysis attacks that have been developed will be described. For demonstrative purposes, a worked example of a power analysis attack will be presented. Common countermeasures that have been developed to deter power analysis will then be described.

2.2.1 Power Analysis Attack Methodology

Two main stages are required to perform a power analysis attack, or any other SCA. The first stage consists of an interaction or acquisition stage, where the data of interest is captured. The second stage consists of an exploitation or analysis phase where the acquired data is analysed to reveal secret data. These stages are considered below.

2.2.1.1 Data Acquisition

Capturing the power consumption requires data acquisition equipment. This must be configured and calibrated to the target device, so that the optimal signal is acquired. The

most common device attacked using power analysis is the smart card [86, 87], although other devices such as an ASIC [102] and FPGA [101] have more recently come under the scrutiny of power analysis. The experimental apparatus in which a smart card is the target will be described.

Smart cards are currently infiltrating many aspects of our everyday life and so can be found in most peoples' possession. A smart card is a plastic card in which a microprocessor is embedded. It generally consists of an 8, 16 or 32 bit processor, with ROM, EEPROM, and RAM, so is capable of performing computations. The smart card processor is embedded in a gold or silver chip, as depicted in Figure 2.1, connecting it to the outside world through eight pins. Each pin has a different purpose and the locations, dimensions and functional assignments of the contact points are specified in the ISO 7816 standard [53]. The main pins of interest for power analysis are the power supply - pin C1, the ground - pin C5 and the I/O - pin C7.

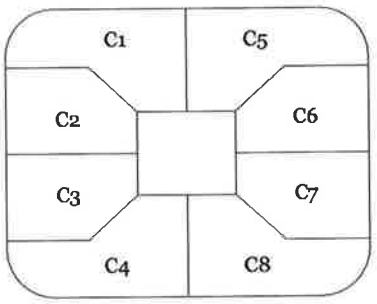

Figure 2.1: Smart card contact points.

A smart card is an easy target for SCA (and fault attacks). Firstly, they are used in many cryptographic applications. The presence of a processor and memory on the smart card allows cryptographic operations to be executed, which in turn requires storage and usage of confidential information such as a secret key. Secondly, a smart card is a small, portable device which can easily be stolen. Thirdly, it is relatively straightforward to obtain the power consumption of a smart card. In addition, due to obvious space

restrictions, typical hardware components to reduce or shield emanations cannot be used.

A typical experimental apparatus where a smart card is being attacked using power analysis is depicted in Figure 2.2. This illustrates the attack configuration where, as the host instructs the smart card to perform a cryptographic operation, the power consumption is acquired using a digital oscilloscope.

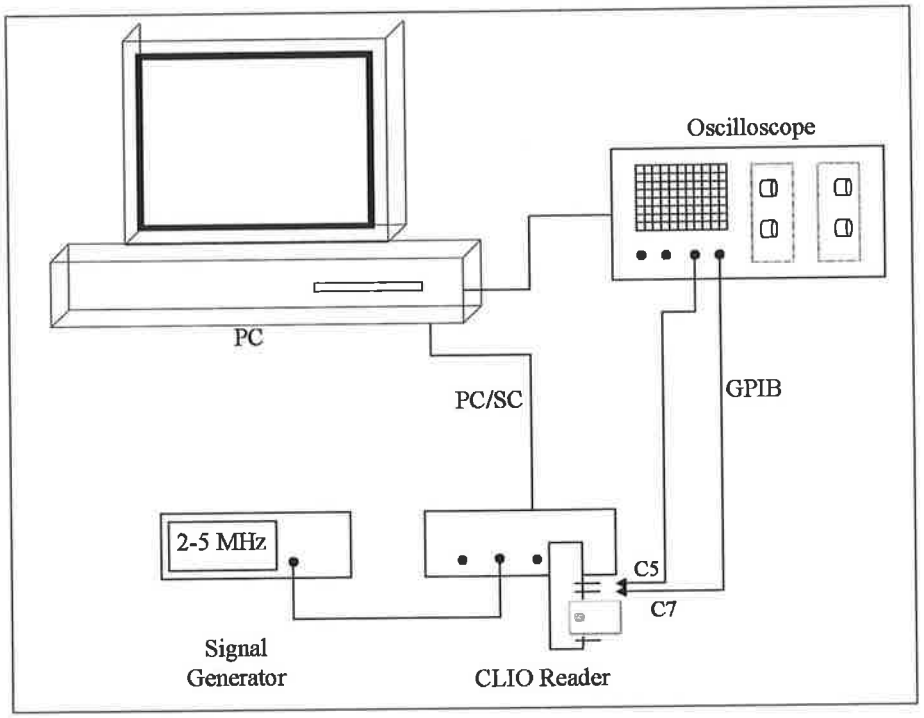

Figure 2.2: Experimental apparatus for power analysis attack.

Acquiring the power consumption is facilitated by a smart card extension (which is a customised printed circuit board (PCB), photographed in Figure 2.3 and 2.4), which allows the pin contacts on the smart card to be accessed externally. By placing a resistor in series between the smart card's ground (C5 pin) and the true ground (in the reader), a probe can sample the voltage difference across the resistor, thus giving a measure of power consumption.

Generally as many acquisitions as possible will be captured, and then sent on to the data analysis phase. The efficiency and effectiveness of data analysis will depend on the

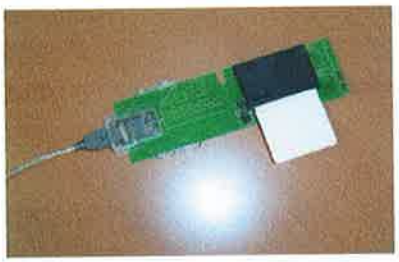

Figure 2.3: Smart card reader, PCB and card support.

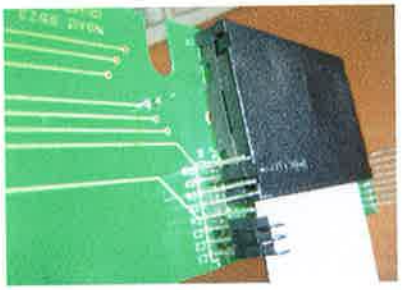

Figure 2.4: Close up of PCB, card support, pins and jumpers.

quality of the signals captured and so measures are taken in the data acquisition phase to improve signal quality. Two influences that affect signal quality are noise and desynchronisation. To minimise noise an array of filters and signal processing techniques are used. One of the most common practices is to capture the same acquisition a number of times (where the same data is repeatedly used) and then compute an average. To ensure that the power traces are synchronised techniques such as synchronised sampling are used, where the sampling equipment and smart card's internal clock are coordinated. This is where the modified reader (CLIO reader) and signal generator comes into play in Figure 2.2. A CLIO reader is a specialised high precision reader that has added functionality to control certain aspects of the smart card while in operation, such as control of the clock frequency which the smart card runs at. The ISO 7816 standard for smart card external clocks is 3.57 MHz [53], although more recent smart cards may run up to 30 MHz. Depending on the chip in the smart card, an internal clock may or may not be present. In the case where the internal clock is absent, the smart card will run at the reader's clock frequency. This is generally either 3.57 MHz and 3.68 MHz. In the case where the internal clock is present, the smart card will communicate to the reader at the reader's frequency, and then revert back to its internal clock frequency for carrying out instructions. The additional functionality that special purpose readers provide is to control the clock frequency. This allows the reader to dictate to the smart card what frequency to work at and so overrides the

smart card's internal clock. The most utilised clock frequencies of the CLIO reader are 1 MHz, 2 MHz and 4 Mhz. The advantage of these frequencies is that they give a constant number of samples per clock cycle. Another common practice to improve the quality of the signal is to attach a probe to the I/O pin on the smart card (C7 pin), and trigger at the end of communication between the smart card and terminal. When communication between the smart card and terminal has terminated, the smart card will begin executing the operation which it has been assigned. This will be a common starting point in all executions and so in all acquisitions. By triggering at this point all acquisitions will be synchronised at this same point in time.

The data acquisition phase will when possible, capture many acquisitions of the target while in operation. This may consist of inputting repeated, random or chosen messages to the target device and consequently, the target cryptographic algorithm. Once the required number of acquisitions have been captured they, along with the input data values, will be passed on to the data analysis phase of the attack.

2.2.1.2 Data Analysis

During a device's operation and cryptographic algorithm's execution, power will be consumed and hence dissipated. As described above, this can be captured using data acquisition equipment. This data captured, which represents the power consumption and is seen on the data acquisition equipment as a signal, will be referred to as a **power trace** and will be denoted by t for the remainder of this thesis. The power consumption and so power trace captured is representative of the execution of the cryptographic algorithm and so will represent different stages of the algorithm from start to finish. Locations in the power trace will relate to different parts in the algorithm where certain operations are carried out on certain data.

For example, Figure 2.5 depicts the power consumption of an AES execution. From this it can be seen that the there is a distinct repetitive pattern to this algorithm. AES

consists of a number of rounds (determined by the bit length of the secret key) and so the power trace mirrors what is happening in the algorithm. Figure 2.6 illustrates an

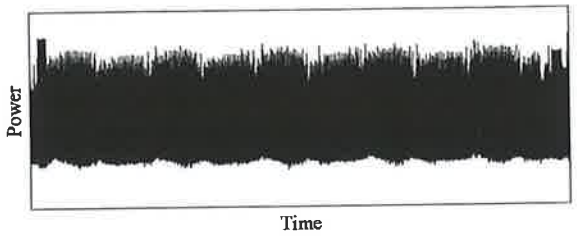

Figure 2.5: Power trace taken from a device executing AES.

amplified view of a power trace. This power trace was acquired during the execution of DES. As can be seen from the labels on the image, specific features of the algorithm can be identified[1].

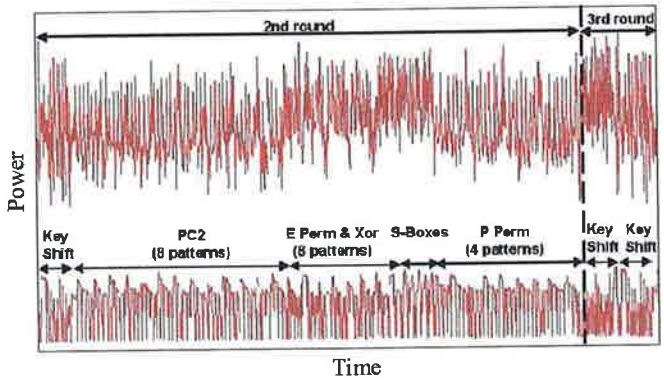

Figure 2.6: Power trace taken from a device executing DES.

Power consumption is contributed to by many factors [85]. One of these factors is the operation and operands being processed. Given that at some point in the cryptographic algorithm the secret will be involved in an operation, power consumption relating to that secret (or operations that the secret is involved in) will be contained in the power trace.

In older technologies, where SCAs were not yet a concern, this relationship was worryingly noticeable to the extent that secret bits could be visibly read from a power trace.

[1]Note, images 2.6 and 2.5 are courtesy of Gemplus Research and Development Laboratories [52].

For example, the well known attack, Simple Power Analysis (SPA) of the square and multiply algorithm for modular exponentiation simply reads the private key bit by bit, as the power consumption for a square and a multiply operation was differentiable in the power trace.

In more recent technologies, where SCA has influenced hardware design and algorithm implementation, such straightforward attacks are less of a concern. However information in the power trace relating to the secret is, while hidden, inescapably present. By taking many acquisitions and using statistical analysis techniques, data dependant power consumption can still often be identified.

Two main models have been put forward to describe how the data being operated on effects the power consumption, they are namely the Hamming weight [66] and the Hamming distance model [24]. These models are central to understanding how to retrieve the information of interest hidden in power traces.

In an w-bit microprocessor, binary data is coded $D = \sum_{j=0}^{w-1} d_j 2^j$, with bit values $d_j = \{0, 1\}$. The Hamming weight model is the simplest model and was initially proposed in [66, 85]. It is based on the assumption that the power consumption is proportional to the Hamming weight of the data being handled at any point in time, that is the number of bits set to 1. It involves an affine relationship between the power consumption and the Hamming weight of the data being manipulated at a given point in time. This can be expressed as,

$$P = aH(D) + b \quad \text{where} \quad H(D) = \sum_{j=0}^{w-1} d_j$$

and P is the power consumption, D is the value of the data being manipulated, H is a function that calculates the Hamming weight, a is the scalar gain between the power consumption and the Hamming weight and the variable b is all other factors, such as acquisition noise, offsets and time dependent components contributing to the power con-

sumption.

The Hamming distance model, proposed by [24], generalises the Hamming weight model. The Hamming distance model is based on the assumption that the power consumption is proportional to the number of bits that flip from one state to the next and so involves the addition of another variable, R. This model can be expressed as,

$$P = aH(D \oplus R) + b$$

where the notation is the same as described in the previous model, and the value R represents some previous state.

One of the most power consuming components on a smart card microprocessor is the bus. The bus is responsible for transferring data between components, and in particular for fetching data and instructions to be executed by the microprocessor. As all the information being processed by the chip is carried across the bus, this relationship can be exploited when secret information is sent across the bus. The Hamming weight model is most appropriate when between each value being sent across the bus, it is set to zero. If it is not set to zero between each value, then there is a transition from state R to state D. Generally a command fetched by the CPU will consist of several opcodes, which is usually an instruction followed by the data to be manipulated. This change in value between R and D can be modeled by an exclusive or (XOR) between the two values. In this case, where typically D is the data being manipulated and R is the instruction opcode, the Hamming distance model is the most appropriate model to model the power consumption.

These models can be used to estimate the power consumption. Given that the secret data will be used at certain points in the computation, it is reasonable to try and estimate the power consumption of data which is guessed to be the secret. The estimated power of the guessed secret, calculated using the chosen model, can then be correlated to the actual power to determine whether the guess for the secret was correct or not. This is the general

idea behind using the power consumption to extract secret data, however just estimating the power consumption for the secret is insufficient since it will not be known where in the power trace to correlate to. Furthermore, it will not be known whether the correlation is being calculated to the secret or some other value. The usage of an operation in the algorithm involving some part of the secret and data known to the adversary, allows insight into the target locations where the secret is involved. In the literature, this operation is called a **Selection Function.** A selection function, denoted by $S()$, is an intermediate operation in the computation which involves some data that is known or computable by the adversary and some data which is in some way related to the secret key. Data which is known by the adversary will be denoted by α. Data which is related to the key will be denoted by k. On accepting input α and k, the selection function will produce an unknown intermediate output value denoted by β. Therefore, $\beta = S(\alpha, k)$. Generally k will relate to some n-bit portion of the key, and α will relate to some n-bit portion of the plaintext P.

Since it will be known that n bits relating to the secret key are entered into the selection function $S()$ with known data α, the hypothetical output of the selection function can be calculated given guesses for k. Let k_{guess} denote one such guess, then the hypothetical output $\beta = S(\alpha, k_{guess})$ can be calculated. As there are will be 2^n possible values that k_{guess} can be, the hypothetical output of the selection function for each guess can also be calculated. This process in the literature is referred to as making **key hypotheses**.

Differential side channel attacks are based on the adversary being able to capture many acquisitions of the target. This may involve executing the cryptographic algorithm with different input. Let N denote the number of times the cryptographic algorithm is executed and P_i denote the input for each execution, where $1 \leq i \leq N$. Then the hypothetical output $\beta_i = S(\alpha_i, k_{guess})$ can be calculated, where α_i is derived from the input P_i.

The hypothetical output produced from the selection function, can be used along with

the power traces acquired, to identify which of the guesses for k is correct. For each guess for k, the power consumption is estimated for the various input P_i, based on the power model selected. Hence, β_i is converted to its estimated power consumption value. The estimated power consumption of β_i is then correlated to the actual power traces. This is repeated for all possible values of k_{guess} and input P_i. The key hypothesis with the highest correlation is identified as the correct value for the portion k of the secret. The way in which the correlation is performed and the correctness of a key hypothesis is established, depends on the type of attack and power model adopted. These will be discussed next.

2.2.2 Types of Power Analysis

The first attack combining many acquisitions with statistical analysis techniques to reveal secret information, was Differential Power Analysis (DPA) [66]. Since the discovery of DPA, various types of power analysis attacks have been defined. An overview of these attacks will be presented in this section. In each description it is assumed that the target device is invoked N times with N random input values P_i, where $1 \leq i \leq N$. The cryptographic transformation of P_i under the secret key K, $T_K(P_i)$, to produce the corresponding output C_i, will result in N power traces t_i. The most familiar scenario is where $C_i = E_K(P_i)$ is calculated, and the cryptographic transform is an encryption algorithm E, which encrypts the plaintext P_i to produce ciphertext C_i.

2.2.2.1 Differential Power Analysis

Given that a suitable selection function $S()$ is identified, the correctness of a key hypothesis is established as follows. For each key hypothesis k_{guess}, where $0 \leq k_{guess} < 2^n$, the hypothetical output of the selection function $\beta_i = S(\alpha_i, k_{guess})$ is calculated, where α_i is related to the known plaintext P_i. DPA categorises the power traces t_i into two sets depending on one bit b of the hypothetical output β_i. The first set S_0 will contain all the

traces where b is equal to zero, and the second set S_1 will contain all the remaining traces, i.e. where the output bit b is equal to one. A differential trace $\Delta_{k_{guess}}$ for each hypothesis is calculated by finding the average of each set and then subtracting the resulting values from each other, i.e.

$$\Delta_{k_{guess}} = \frac{\sum_{t_i \in S_0} t_i}{|S_0|} - \frac{\sum_{t_i \in S_1} t_i}{|S_1|} \tag{2.1}$$

The differential trace with the highest peak will validate a hypothesis for k_{guess}, that is k_{guess} corresponds to the $\Delta_{k_{guess}}$ featuring a maximum amplitude.

DPA is based on the Hamming weight model and assumes that the power consumption for a logical 1 is different than a logical 0. By taking many acquisitions, when the correct secret is used to partition the traces, this difference is amplified and so presents itself in the form of a spike as depicted in Figure 2.7.

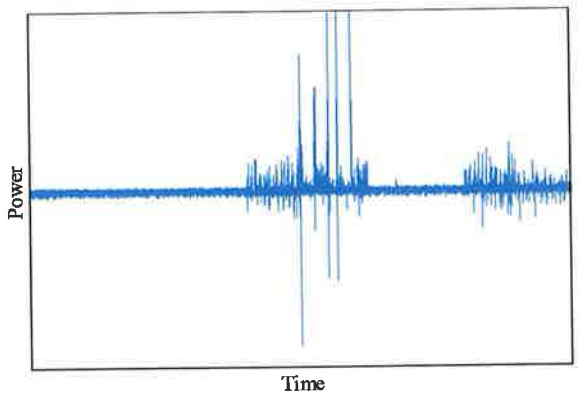

Figure 2.7: Example of a differential trace $\Delta_{k_{guess}}$.

DPA has proven to be a very powerful attack. It has been applied to a number of cryptographic algorithms and has lead to the development of more honed attacks. DPA variants have been developed either in response to a countermeasure proposed to deter DPA [84], to target a specific algorithm [86], or to overcome some of the defects that DPA was shown to exhibit [24]. The most notable developments have been multiple-bit DPA [82, 72], second-order DPA [131, 57], Correlation Power Analysis (CPA) [24] and

Template attacks [28, 104].

2.2.2.2 Correlation Power Analysis

Some of the assumptions that DPA attacks make are incomplete and inaccurate. CPA was developed by [24] to address some of the defects of DPA. In contrast to DPA, CPA [24] is based on the Hamming distance model (although it can also be used with the Hamming weight model) and so takes into consideration previous states. In CPA the correctness of a key hypothesis is established as follows. For each key hypothesis k_{guess} where $0 \leq k_{guess} < 2^n$, the hypothetical output of the selection function $\beta_i = S(\alpha_i, k_{guess})$ is calculated, where α_i is related to the known plaintext P_i. CPA not only takes the entire output of the selection function into consideration, but considers the entire processors word. The estimated power consumption of this word is calculated based on the power model and then compared to the actual power consumption using a correlation test such as Pearson's correlation coefficient [90] (Equation (2.2)),

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\sqrt{E(Y^2) - E^2(Y)}} \tag{2.2}$$

where X is the power consumption at a given point in time and Y is the estimated power consumption derived from the power model adopted. The correlation coefficient with the value closest to $+1$ is identified as the correct key guess.

2.2.2.3 First-Order Power Analysis

Both previous forms of power analysis attacks are types of first-order power analysis. First-order power analysis is defined by [131] as "attacks characterized by the property that they exploit highly local correlation of the secret with the power trace. Typically, the secret-correlated power draw occurs at a consistent time during the encryption and has consistent sign and magnitude." In particular, a first-order attack exploits situations

where known data α comes in direct contact with secret data k, that is a selection function of the form $\beta = S(\alpha, k)$ exists.

2.2.2.4 Second-Order Power Analysis

A second-order DPA attack was originally defined by [66], and has since been developed by [131, 57]. Second-order DPA combines one or more samples within a single power trace, then formulates an attack based on the joint statistical properties of multiple aspects of the power traces. For example, an operation of the form $\beta = S(\alpha, k)$ does not exist in the computation, but an operation of the form $y = M(\alpha, x)$ exists where x is a random value and y is then later involved in a operation of the form $\beta = S(y, k)$ with the secret data k. This describes a typical masking scheme specifically designed to deter first-order power analysis by eliminating suitable selection functions involving known and secret data. Since there exists a relationship between the two functions $y = M(\alpha, x)$ and $S(y, k)$, the power consumption still may be exploited by examining the locations in the power trace relating to these operations. This is just one (very trivial) example of second-order power analysis. Second-order DPA is more powerful than first-order DPA and has shown to be effective against implementations resistant to first-order DPA. For example, masking inhibits first-order DPA, but succumbs to second-order DPA.

2.2.3 An Example of Power Analysis

Various cryptographic algorithms and underlying functions in both public and private key algorithms have been assessed for vulnerability to power analysis. For example, the Data Encryption Standard (DES) [66, 46, 6], the Advanced Encryption Standard (AES) [27, 83, 6], XTR [47], RSA [86, 63], IDEA [73], RC5 [61] and RC6 [73] and cryptosystems based on elliptic curve cryptography (ECC) [31, 134, 9], have been the subject of power analysis attacks. This has seen the development of both power analysis attacks and countermeasures tailored for specific algorithms.

Here a particular example of a CPA attack of a naive implementation of AES, which was implemented on a Xilinx FPGA. The AES is a symmetric key encryption algorithm, which was introduced in 2001 as the successor to DES [34, 98]. An overview of AES is given in Figure 2.8.

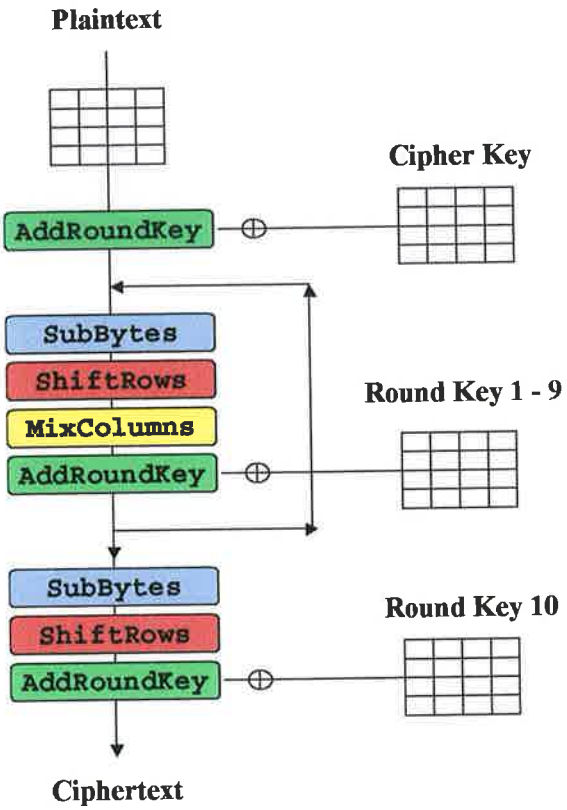

Figure 2.8: The Advanced Encryption Standard (AES).

The intermediate operation or selection function of interest that CPA targets is the `SubBytes` function in the first round of the algorithm. This function satisfies all the properties required to make it a suitable selection function; it takes some known data related to the input $\alpha_{i,j}$, where $\alpha_{i,j}$ represents a byte of the plaintext P_i, and some unknown data related to the key k_j, where k_j represents a byte of the key K, to produce some unknown intermediate output data $\beta_{i,j}$, i.e. $\beta_{i,j} = \texttt{SubBytes}(\alpha_{i,j} \oplus k_j)$ for $1 \leq j \leq 16$

bytes of the 128 bit key and the 128 bit plaintext. Since AES was specifically designed to be byte-oriented and is implemented over the field GF(2^8), an 8-bit processor will be considered as the target for CPA, and so a byte of the key will be extracted at a time.

If the first round of the encryption is targeted, $\alpha_{i,j}$ will relate directly to the plaintext, and k_j will relate directly to the key. To find k_0, the first byte of the key, the hypothetical output of the `SubBytes` function is calculated for all input plaintexts[2] P_i and possible values for k_0, i.e. $0 \leq k_0 < 2^8$. This will produce an $N \times 2^8$ matrix. Each entry in the matrix will be converted to its corresponding estimated power consumption value based on the power model chosen, which in this case was the Hamming weight model. To test each key hypothesis, the correlation coefficient (given in Equation (2.2)) is calculated between each column in the matrix and the discrete time interval in all power traces where the `SubBytes` operation is identified to be executed. The column with the highest correlation is identified to be the correct key byte for k_0. The following figure shows the result of the correlation when 5000 plaintexts were encrypted using AES, and consequently relates to the number of acquisitions. Therefore, $N = 5000$. The correct key, which was the value 0x84, shows the strongest correlation of ≈ 0.18 as illustrated[3] in Figure 2.9. With more acquisitions, this correlation would be expected to tend towards 1. To find the remaining bytes of the key, the same process is performed except with the relevant plaintext bytes.

The cryptanalytic process that was carried out to find the successor to DES, AES, was a rigorous exercise. The selection process attracted 15 different cryptographic designs from several countries. Each of these cryptosystems were publicly scrutinised by the cryptographic community. AES, originally known as Rijndael, was selected the winner and successor to DES after a five year standardisation process. While various countermeasures have been implemented to protect AES against power analysis, the ease with

[2]Actually only the first byte of the plaintext, which is exclusive or-ed with k_0, is required.

[3]This figure was generated from acquisitions captured during collaboration with University College Cork, Department of Electrical & Electronic Engineering.

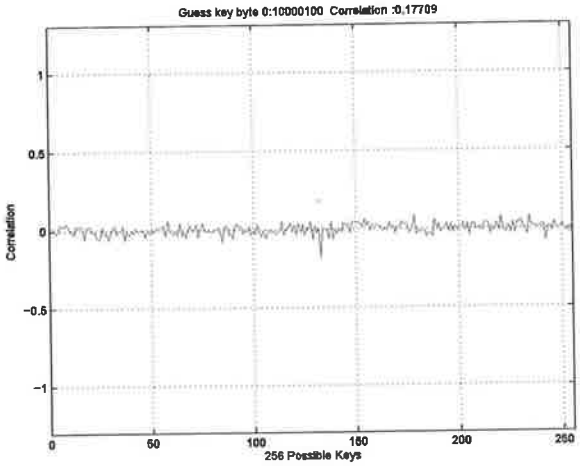

Figure 2.9: Correlation for k_0.

which a power analysis attack can be launched on an approved algorithm, highlights the significance of such implementation attacks.

2.2.4 Common Countermeasures

There have been two main ideologies in defining deterrents to power analysis attacks. The first approach is to bombard the power dissipation with noise so that the number of acquisitions required to distinguish the signal of interest from noise is infeasible. Insertion of random delays, random reordering of operations and usage of hardware and software noise generators are examples of some of the techniques developed to increase the number of acquisitions that an adversary has to capture.

The second approach is to make the information present in the power trace unrelated to the secret. For example, boolean and arithmetic masking of the plaintext with a random value prevent predictions about the intermediate output of the selection function being made. Many masking mechanisms have been proposed [46, 44, 6, 33]. However they have also been shown to be vulnerable to second-order DPA [83, 32] and template attacks [104].

2.3 Fault Analysis

A fault attack is a type of attack, which purposely seeks to disrupt the normal execution of a device, with the objective of producing faulty results which facilitate the extraction of secret information. In this section, the various methods that have been developed to induce a fault attack, types of faults that exist, and consequently the effects that these faults can have, will be described. To demonstrate the potential of a fault attack, an example where an otherwise mathematically robust cryptographic algorithm breaks down when exposed to a fault attack is presented. Hardware and software countermeasures developed for fault detection and prevention will then be described.

2.3.1 Fault Analysis Attack Methodology

Performing a fault attack involves disrupting the normal execution of a device in some way so that faulty output is produced. To do this an appropriate mechanism by which to induce a desirable fault must be identified.

2.3.1.1 Mechanisms for Fault Creation

Various mechanisms for fault creation and propagation have been researched and developed. One of the most obvious approaches is to alter the normal working environment of the device. During chip manufacture, a number of predefined thresholds for characteristics such as supply voltage, clock or temperature are set. By forcing the device to work outside of these boundaries, abnormal behaviour can be provoked.

Equipment such as an alcoholic cooler can be used to vary temperature until the chip exceeds the threshold's bounds. Glitch attack equipment, which consists of a modified reader with special capabilities, can be used to dictate nonstandard supply voltage or clock frequency to the device. Figure 2.10 shows a glitch attack set up. Depicted on the left is a CLIO reader and signal generator as described in Figure 2.2. The CLIO reader

accepts nonstandard clock frequency from the signal generator, which it in turn dictates to the smart card. The change in clock frequency generally only lasts for a short period of time at a certain stage during the execution where it is hoped to achieve the desired fault affect. On the right, a CLIO reader with differential probe is depicted. The CLIO reader alone possesses the functionality to supply nonstandard voltage to the smart card during execution.

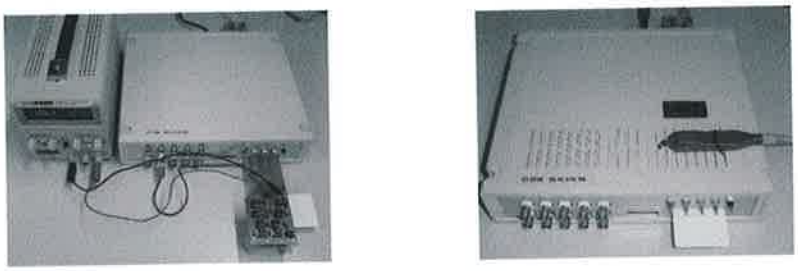

Figure 2.10: Glitch fault attack set up consisting of a modified reader, signal generator and differential probe.

Another approach to induce a fault is to expose the chip to extreme conditions. For example, by exposing a chip to intense light or a laser beam, since all electric circuits are sensitive to light due to photoelectric effects, the current induced by photons can induce a fault. The disadvantage with these attacks is they require more elaborate equipment such as a Scanning Electron Microscope (SEM) or Focused Ion Beam (FIB). A typical laser injection laboratory environment is depicted in Figure 2.11.

Figure 2.11: Outer (left) and inner (right) view of laser fault injection equipment.

2.3.1.2 Types of Faults

Once a fault has been injected, two types of faults may be produced, namely provisional (transient) and destructive (permanent) faults.

In a provisional fault, silicon is locally ionized so as to induce a current that, when strong enough, is falsely interpreted by the circuit as an internal signal. As ionization ceases so does the induced current (and the resulting faulty signal) and the chip recovers its normal behavior. Provisional faults have reversible effects and the circuit will recover its original behavior after the system is reset or when the fault's stimulus ceases. Examples of provisional faults include single [100] and multiple event upsets [110] and dose rate faults [68].

In contrast, destructive faults, created by purposely inflicted defects to the chip's structure, have a permanent effect. Once inflicted, such destructions will affect the chip's behavior permanently. Examples of permanent faults include single event burnout [70, 127], single event snap back [67], single event latch-up [4] and total dose rate faults [26].

When using fault injection as an attack strategy, provisional faults are the method of choice. These allow for faults under numerous experimental conditions to be attempted until the desired effect is achieved. As a bonus, the system remains functional after the attack's completion. In contrast, a destructive fault would (usually) render the target unusable and will necessitate the manufacturing of a clone. For this dissertation, only provisional faults will be considered.

2.3.1.3 Effects of Faults

A provisional fault attack can have a number of effects. Memory can be modified. Data can be misread. Operations such as read and write operations can be broken. In an algorithm these effects are witnessed in the form of data being corrupted, loops running over or ending prematurely, or instructions being omitted or misinterpreted.

The exact effect that the induced fault causes, and interpretation or knowledge of that

effect, depends on the capability of the adversary. At one end of the spectrum exists a very powerful adversary who can dictate the time, location and effect of the fault. At the other, an adversary who has no control over these factors and must make use of a fault with unknown effect. To aid the adversary and empower the fault attack, the power consumption of the target can be used. As described in Section 2.2, the power consumption represents the execution of an algorithm where different sections in the power trace relate to different stages in the execution of the algorithm. For example, the loop iteration or round can be identified in the power trace. Therefore by monitoring the power trace, the optimal time to inject the fault can be identified. The power trace can also be used to tell whether the fault has been successful and has disturbed the normal execution in any way. Examples of how the power trace may be useful are depicted in Figures 2.12 and 2.13. These are an example of a glitch attack where the device's normal execution's power trace is compared to the device's faulty execution's power trace. In the first example (Figure 2.12), the processor has skipped a number of instructions and resumed normal execution several microseconds after the glitch has been injected. This fault allows the selective execution of instructions in a program. In the second example (Figure 2.13), not only does the processor skip instructions, but the value of data manipulated by the processor is also modified.

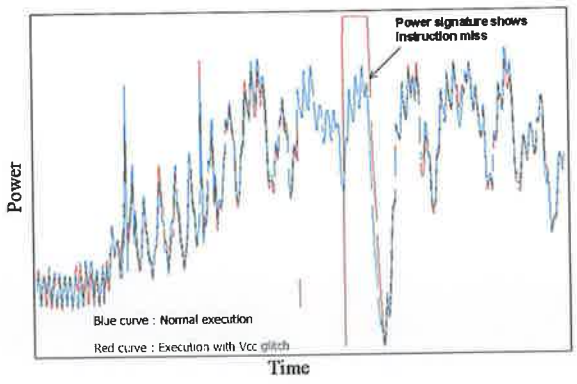

Figure 2.12: Example of a glitch attack which effect that a number of instructions are missed.

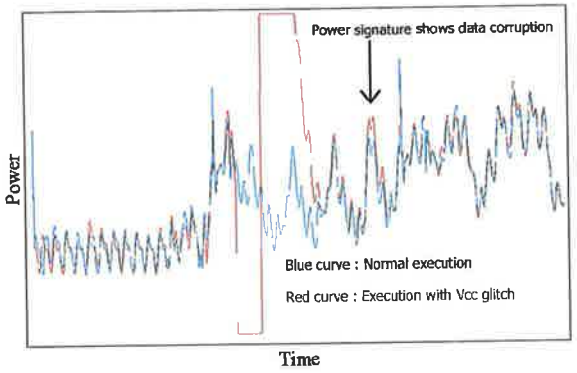

Figure 2.13: Example of a glitch attack which effect that a number of instructions are missed and data is corrupted.

The implications of these effects of faults on cryptographic algorithms are potentially catastrophic. This will be demonstrated by an example of a fault attack on an implementation of RSA using the Chinese Remainder Theorem (CRT) in Section 2.3.3.

2.3.2 Types of Fault Attacks

Similar to power analysis attacks, there are two types of fault attacks, which are distinguished by the number of samples that are required and the analysis procedures that are subsequently applied.

2.3.2.1 Simple Fault Analysis

Simple Fault Analysis (SFA) exploits a direct relationship between a faulty result and the secret in the implementation, and allows the secret to be derived directly from the erroneous output. The most prominent key exposure attack is on an RSA-CRT implementation that can succeed with one fault. This example of SFA will be discussed below.

2.3.2.2 Differential Fault Analysis

Differential Fault Analysis (DFA) requires a certain number of faulty computational results using the same secret. Here one assumes that faults are caused in a transient way. The faulty outcomes are used in conjunction with other techniques to reduce the key space. An example of DFA will be given in Chapter 4, when lattice reduction techniques are used to extract the secret from multiple faulty DSA computations.

2.3.3 An Example of Fault Analysis

The effect of opportunistic faults on cryptographic algorithms has not been researched as thoroughly as SCA. In this section, an example of how a fault attack can weaken the security of a cryptographic algorithm and allow extraction of the secret key is provided. This example is based on the attack of an implementation of RSA using the Chinese Remainder Theorem (CRT), which was presented in the first academic fault attack paper [21].

The attack on RSA using CRT is very simple, only requiring one fault to be inserted in order to factor the RSA modulus. Basically the attack works as follows: Let $N = p \times q$, where p and q are two large prime numbers. Let $m \in \mathbb{F}_N^*$ be the message to be signed, d the private key and s the RSA signature. The pre-computed values required for use in the CRT are a and b, such that

$$\begin{cases} a \equiv 1 \pmod{p} \\ a \equiv 0 \pmod{q} \end{cases} \text{ and } \begin{cases} b \equiv 0 \pmod{p} \\ b \equiv 1 \pmod{q} \end{cases}$$

and d_p and d_q such that

$$d_p = d \pmod{p-1}$$
$$d_q = d \pmod{q-1}.$$

Using repeated square and multiply calculate

$$s_p = m^{d_p} \pmod{p}$$
$$s_q = m^{d_q} \pmod{q}.$$

The RSA signature s is then obtained by the linear combination $s = a \times s_p + b \times s_q \pmod{N}$.

The attack is based on being able to obtain two signatures of the same message, where one signature is correct and the other faulty. The faulty signature is where a fault is injected during the computation and corrupts either the computation of s_p or s_q. Figure 2.14 illustrates the power consumption of a device performing RSA with CRT. This depicts four sources of power dissipation. The two traces of interest are the power consumption of a device performing RSA with CRT and the laser. Notice that the execution of the two exponentiations is clearly visible in the power trace and so the point in time where to inject the fault can be easily identified.

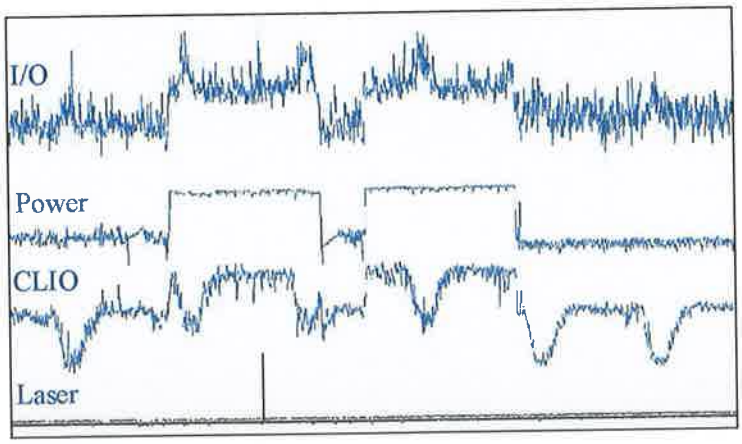

Figure 2.14: Power consumption measurements taken during a fault attack on RSA with CRT.

Let $\hat{s} = a \times s_p + b \times \hat{s}_q \pmod{N}$ be the faulty signature (here it is assumed that the error occurred during the computation of s_q, but the attack works just as well when s_p is

corrupted). Subtraction yields:

$$\begin{aligned}\Delta &\equiv s - \hat{s} \\ &\equiv (a \times s_p + b \times s_q) - (a \times s_p + b \times \hat{s}_q) \\ &\equiv b(s_q - \hat{s}_q) \pmod{N}\end{aligned}$$

Hence, as $b \equiv 0 \pmod{p}$ and $b \equiv 1 \pmod{q}$ it follows that $\Delta \equiv 0 \pmod{p}$ (but $\Delta \not\equiv 0 \pmod{q}$) meaning that Δ is a multiple of p (but not of q). Hence, a greatest common divisor (GCD) calculation gives the secret factors of N, i.e. $\text{GCD}(\Delta \pmod{N}, N) = p$ and $q = N/p$.

In summary all that is required to break RSA is one correct signature and one faulty one. This attack will be successful regardless of the type or number of faults injected during the process provided that all faults affect the computation of s_p or s_q. The attack was extended in [56] to show that it is not even necessary to generate a correct signature. The faulty signature alone with the message, $\text{GCD}(\hat{s}^e - m \pmod{N}, N) = p$, where e is the public verification exponent, will yield the straightforward factorisation of N (an otherwise very difficult problem).

2.3.4 Common Countermeasures

Methods to avoid, detect and/or correct faults have been developed at two levels. At one level there are hardware protection, at the other software. Generally, best practices employ a combination of protection at both levels.

Hardware protections are implemented by the chip manufacturer and can be either active or passive. Some of the most common active protections include detectors such as light, supply voltage and frequency, which react to any changes outside the predefined tolerance thresholds of the chip. Metal meshes, known as active shields, that cover the entire chip and have data continuously passing through them identify if there is a disconnection or modification of the mesh. Time and hardware redundancy techniques such

as simple time redundancy with comparison (STRC) [7], multiple time redundancy with comparison, re-computing with swapped operands, re-computing with shifted operands [108], simple duplication with comparison (SDC), multiple duplication with comparison (MDC), simple duplication with complementary redundancy (SDCR), dynamic duplication [76] and hybrid duplication. These methods not only detect faults, but have the capability to correct them. Passive protections include mechanisms that introduce dummy random cycles during code processing, bus and memory encryption, and a passive shield which is a metal layer covering some sensitive chip parts making light or electromagnetic beam attacks more difficult as the shield needs to be removed before the attack can proceed.

Software countermeasures are implemented when hardware countermeasures are insufficient or as protection against future attack techniques that might defeat present-generation hardware countermeasures. The advantage of software countermeasures is that they do not increase the hardware block size, although they do impact on the protected functions' execution time. Examples of software countermeasures include the use of checksums, randomisation of the execution or repeating an execution and comparing the results (this is known as execution redundancy).

2.4 Conclusion

The objective of this chapter was to give an introduction to the research area. In particular, it was intended to demonstrate the power of implementation attacks and show how even cryptosystems which are widely accepted by the cryptographic community as otherwise secure, are weak and terrifyingly insecure when it comes to implementation attacks. The goal of our research is to use power and fault analysis to attack unassessed bilinear pairings. Therefore background material on pairings will be provided in the next chapter.

Chapter 3

Bilinear Pairings

3.1 Introduction

Since the introduction of bilinear pairings in the constructive sense, a multitude of pairing based protocols have been suggested. The viability of such applications however, ultimately depend on the efficiency and ease of computing the bilinear pairing. Computation of a bilinear pairing requires knowledge and understanding of both finite field and elliptic curve arithmetic.

The purpose of this chapter is to provide the necessary mathematical background, on both finite fields and elliptic curves, for bilinear pairings. In particular, it is the implementation of such pairings which are of interest, since the topic concerns assessing pairings for vulnerability to power and fault analysis. Hence, it is not the intention of this chapter to be comprehensive, providing only the necessary details required for the implementation of pairings.

Firstly, finite fields are examined. Finite field are central objects in cryptography, because they enjoy very special properties. Initially they were the core of cryptosystems such as RSA and ElGamal [77]. In elliptic curve cryptography they serve as the building blocks upon which the elliptic curve is defined. Hence, the choice of field and arithmetic

over the field are vital to the pairing computation. Specifically the binary and prime fields are dealt with since these form the basis for the candidate pairing implementations examined later in Chapter 5 and Chapter 6.

Secondly, elliptic curves are examined. The use of elliptic curves for cryptography was suggested by Koblitz and Miller in 1985. The algebraic structure of elliptic curves makes them particularly attractive for public key cryptography. More importantly they form the basis from which pairings are constructed. Specifically, the relevant theory and required operations on elliptic curves for bilinear pairings will be discussed.

Finally, bilinear pairings are examined. Bilinear pairings ultimately rely on the underlying elliptic curve, and hence the underlying finite field. Building on the material presented in the previous sections, relevant material on bilinear pairings will be presented. Specifically, background on pairings, how they are computed, and the recent advances in the field are discussed. The particularly efficient and optimised candidate pairings, to be used in later analysis, are given. A pairing based protocol, namely Boneh and Franklin's IBE [22], is described to contextualise this research. The associated security of pairings and the study that has been performed in the area of power and fault analysis of elliptic curve cryptography is also discussed.

3.2 Finite Fields

Let $\mathbb{F}_q$ be a finite field where $q = p^m$ and p is a prime known as the characteristic and m is a positive integer known as the extension degree of $\mathbb{F}_q$ over $\mathbb{F}_p$. The non-zero elements of $\mathbb{F}_q$ form a group under multiplication, known as the multiplicative group $\mathbb{F}_q^*$, i.e. $\mathbb{F}_q^* = \mathbb{F}_q - \{0\}$.

3.2.1 The Prime Field

The prime field, where $q = p$, is constructed from integers modulo p, i.e. $\{0, 1, \ldots, p-1\}$, with addition and multiplication performed modulo p. The prime field is denoted by $\mathbb{F}_p$.

3.2.2 The Binary Field

The binary finite field, where $q = 2^m$, can be constructed using a polynomial basis representation. The elements of the binary field are the binary polynomials where the degree is at most $m - 1$. The binary field, also referred to as the characteristic two field, is denoted by $\mathbb{F}_{2^m}$.

$$\mathbb{F}_{2^m} : \{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \ldots a_2z^2 + a_1z + a_0 | a_i \in \{0, 1\}\}$$

Addition of field elements is addition of the polynomials, with coefficient arithmetic performed modulo 2. Multiplication of field elements is performed modulo an irreducible polynomial, denoted by $f(z)$. The irreducible polynomial $f(z)$ is of degree m.

3.2.3 The Extension Field

The polynomial basis representation for binary fields can be generalised to all extension fields. Let p be a prime and $m \geq 2$ known as the extension degree. Let $\mathbb{F}_p[z]$ denote the set of all polynomials in the variable z with coefficients from $\mathbb{F}_p$. Let $f(z)$ be an irreducible polynomial of degree m in $\mathbb{F}_p[z]$. The elements of the extension field $\mathbb{F}_{p^m}$ are the polynomials in $\mathbb{F}_p[z]$ of degree at most $m - 1$.

$$\mathbb{F}_{p^m} : \{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \ldots a_2z^2 + a_1z + a_0 | a_i \in \mathbb{F}_p\}$$

Addition of field elements is the usual addition of polynomials, with coefficient arithmetic modulo $\mathbb{F}_p$. Multiplication of field elements is performed modulo the irreducible

polynomial $f(z)$.

3.2.4 Finite Field Arithmetic

The main arithmetic operations performed in the finite field which are of interest for pairings are addition, subtraction, multiplication, division, exponentiation and inversion. Various hardware and software algorithms exist to compute each of these operations.

To assess pairings in the context of implementation attacks, such as power and fault analysis, it must be considered how finite field elements are stored and operated on. For elements of prime characteristic, let $n = \lceil \log_2 p \rceil$ be the bit length of p and $t = \lceil n/w \rceil$ be its word length, where the device has a w-bit processor. Elements in $\mathbb{F}_p$ will be stored in arrays in memory as illustrated in Figure 3.1.

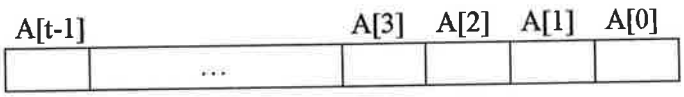

Figure 3.1: Storage of elements in $\mathbb{F}_p$.

Elements in the binary field will be stored similarly, where the bit length of each element is simply its degree m. Elements in the field extension will consist of a number of base field elements determined by the extension degree. For example, for the extension field $\mathbb{F}_{p^m}$ where $m = 4$, elements will consist of four $\mathbb{F}_p$ components.

When implemented on an actual device, such as a smart card, the finite field arithmetic operations will operate on the finite field elements as stored in these structures.

3.3 Elliptic Curves

An elliptic curve $E(\mathbb{F}_q)$ is the set of solutions (x, y) over $\mathbb{F}_q$ to an equation of the form

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_i \in \mathbb{F}_q$, together with an additional point at infinity, denoted $\mathcal{O}$. Simplified variants of this equation exist for characteristic $2, 3$ and p prime.

The number of points in $E(\mathbb{F}_q)$, denoted by $\#E(\mathbb{F}_q)$, is called the order of E over $\mathbb{F}_q$. Hasse's theorem states that

$$\#E(\mathbb{F}_q) = q + 1 - t$$

where the value t is the trace of Frobenius at q and $|t| \leq 2\sqrt{q}$.

3.3.1 The Group Law

The points on an elliptic curve together with an identity element $\mathcal{O}$ and addition operation, form an abelian group which is used in the construction of cryptographic systems. Therefore given two points $P, Q \in E(\mathbb{F}_q)$, there is a third point on $E(\mathbb{F}_q)$ such that $P + Q = R$. Let $P = (x_1, y_1), Q = (x_2, y_2)$ and $R = (x_3, y_3) \in E(\mathbb{F}_q)$. Let the negative of a point $P = (x_1, y_1)$ be $-P = (x_1, -y_1)$ for $q = p$ and $-P = (x_1, x_1 + y_1)$ for $q = 2^m$. The addition rule is as follows.

1. If Q $= \mathcal{O}$ then $P + Q = P$.
2. If $Q = -P$ then $P + Q = \mathcal{O}$.
3. If $P \neq Q$ then $P + Q = R$, where

$$R = \begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases} \quad \text{and} \quad \lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)$$

when the elliptic curve is over the prime field and defined by the equation

$$E : y^2 = x^3 + ax + b,$$

and

$$R = \begin{cases} x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \end{cases} \quad \text{and} \quad \lambda = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)$$

when the elliptic curve is over the binary field and defined by the equation,

$$E : y^2 + xy = x^3 + ax^2 + b.$$

4. If $P = Q$ then $P + Q = R$, where for prime characteristic

$$R = \begin{cases} x_3 = \lambda^2 - 2x_1 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases} \quad \text{and} \quad \lambda = \left(\frac{3x_1^2 + a}{2y_1}\right)$$

and characteristic two

$$R = \begin{cases} x_3 = \lambda^2 + \lambda + a \\ y_3 = x_1^2 + \lambda x_3 + x_3 \end{cases} \quad \text{and} \quad \lambda = \left(\frac{x_1 + y_1}{x_1}\right).$$

Note that the operation of $P + Q$ is known as point addition when $P \neq Q$ and point doubling when $P = Q$.

3.3.2 Point Scalar Multiplication

Given that addition can be defined on elliptic curves, scalar multiplication can also be defined. The computation of $[n]P$, where $n \in \mathbb{Z}$ and $P \in E(\mathbb{F}_q)$, given by

$$[n]P = P + P + \ldots + P \quad (n \text{ times})$$

is known as the scalar multiplication of a point. It is the primary operation in elliptic curve cryptography and achieved through a combination of point additions and doublings.

A number of algorithms exist to perform point scalar multiplication. The most widely

known and straightforward methods are the double and add and the double and add/subtract methods. These methods are derived from the square and multiply algorithm for modular exponentiation, where the binary representation of n is used to determine the sequence of doubles and additions.

3.4 Bilinear Pairings on Elliptic Curves

Pairings in elliptic curve cryptography are functions which map a pair of elliptic curve points to an element of the multiplicative group of the underlying finite field. Two types of pairings were originally defined, namely the Weil [22] and the Tate [14] pairing. Both of these pairings will be defined in this section.

3.4.1 Pairing Related Concepts

There are a number of additional concepts that are required for bilinear pairings. Each of these will be discussed here.

The order of a point $P \in E$ is defined as the smallest positive integer r such that $[r]P = \mathcal{O}$. Points of order r are also known as r-torsion points. The subgroup of r-torsion points in E is denoted by $E[r]$, so

$$E[r] = \{P \in E \mid [r]P = \mathcal{O}\}.$$

The elliptic curve point input to Miller's algorithm, and so input to the bilinear pairing, must be a r-torsion point.

The elliptic curve on which the pairing is implemented over, must possess some special properties. Two categories of elliptic curves, namely supersingular and non-supersingular elliptic curves are used for bilinear pairings. An elliptic curve E defined over $\mathbb{F}_q$ is supersingular if p divides t, where t is the trace of Frobenius [48]. Conversely, an elliptic curve E defined over $\mathbb{F}_q$ is non-supersingular if p does not divide t. Non-

supersingular curves are also referred to as ordinary.

An extension field with special properties is also required. This is known as the embedding degree or security multiplier and is denoted by k. For technical reasons it is assumed that $k > 1$. The embedding degree is an extension degree that provides an extension field that contains the r-th roots of unity in $\mathbb{F}_{q^k}$, where the r-th roots of unity are defined as $\mu_r = \{x \in \mathbb{F}^*_{q^k} | x^r = 1\}$. The value of the embedding degree depends on a number of factors, one of which is the type of elliptic curve. For supersingular elliptic curves, the embedding degree is bounded by $k \leq 6$, although the upper bound of the embedding degree depends on the underlying field. For supersingular elliptic curves over the large prime field, the embedding degree is bounded by $k \leq 2$. For supersingular elliptic curves over the binary field, the embedding degree is bounded by $k \leq 4$. For supersingular characteristic three elliptic curves, the embedding degree is bounded by $k \leq 6$. Hence, generally supersingular curves are used in conjunction with fields of characteristic two or three. For non-supersingular elliptic curves suitable for pairings, the bound for the embedding degree is not as restrictive. For a non-supersingular elliptic curve to be pairing friendly, it must have a reasonably small embedding degree $k \leq 24$ (for computational reasons) and contain a large subgroup r of prime order. However, to randomly find an ordinary curve with these properties, is not easy. Hence, a number of curve construction methods, which construct elliptic curves with these properties, have been developed. For example, MNT [89, 119] give a method for constructing ordinary elliptic curves with embedding degree 3, 4 or 6 and Barreto and Naehrig [16] give a method to construct elliptic curves with embedding degree 12. For bilinear pairings, the elliptic curve or "pairing friendly curve" must have a relatively small embedding degree and contain a large prime order subgroup.

3.4.2 Pairing Definitions

Let E be an elliptic curve over a finite field $\mathbb{F}_q$. Let $r|\#E(\mathbb{F}_q)$ be a prime not equal to the characteristic of the field $\mathbb{F}_q$. The embedding degree k is the smallest positive integer such that $r|q^k - 1$. The embedding degree is chosen this way to ensure that the full r-torsion group $E[r]$ of the elliptic curve is defined over the field $E(\mathbb{F}_{q^k})$, i.e. $E[r] \subset E(\mathbb{F}_{q^k})$. The group of r-th roots of unity in $\mathbb{F}_{q^k}$ is defined as $\mu_r = \{x \in \mathbb{F}_{q^k}^* | x^r = 1\}$. The Weil pairing is a mapping

$$\omega : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})[r] \rightarrow \mu_r, \tag{3.1}$$

and the Tate pairing is a mapping

$$\langle \cdot, \cdot \rangle_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \rightarrow (\mathbb{F}_{q^k}^*)/(\mathbb{F}_{q^k}^*)^r. \tag{3.2}$$

The Weil and the Tate pairing differ both on their input and their output. The Weil pairing accepts both input points from the subgroup of points of order r, $E(\mathbb{F}_{q^k})[r]$, and produces a unique value. The Tate pairing accepts one point from the subgroup $E(\mathbb{F}_{q^k})[r]$ and the other from the quotient group $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$. The output of the Tate pairing is not unique and is a member of a coset of the group $(\mathbb{F}_{q^k}^*)/(\mathbb{F}_{q^k}^*)^r$ which is defined up to r-th powers. To produce a unique value, the reduced Tate pairing is defined. The reduced Tate pairing is defined as

$$e(P, Q) = \langle \cdot, \cdot \rangle_r^{(q^k-1)/r} \tag{3.3}$$

This additional operation of raising the output of $\langle \cdot, \cdot \rangle_r$ to the power of $(q^k - 1)/r$ is known as the final exponentiation. The Weil and the Tate pairing are related [136] in that

$$\omega(P, Q)^{(q^k-1)/r} = e(P, Q)/e(Q, P). \tag{3.4}$$

Pairings exhibit the following properties, making them attractive and useful in a cryptographic context.

1. ω is *bilinear* in each variable. This means that $\omega(S_1 + S_2, T) = \omega(S_1, T)\omega(S_2, T)$ and $\omega(S, T_1 + T_2) = \omega(S, T_1)\omega(S, T_2)$ for all $S, S_1, S_2, T, T_1, T_2 \in E(\mathbb{F}_{q^k})[r]$.

2. ω is *non-degenerate* in each variable. This means that if $\omega(S, T) = 1$ for all $T \in E(\mathbb{F}_{q^k})[r]$ then $S = \mathcal{O}$ and also if $\omega(S, T) = 1$ for all $S \in E(\mathbb{F}_{q^k})[r]$ then $T = \mathcal{O}$.

The same properties hold for the Tate pairing $e(S, T)$.

3.4.3 Computation of Pairings

Two methods to compute a pairing have been put forward. The first method was proposed in an unpublished manuscript by Miller [88] in 1986. This method is based on divisor theory [136] and to date has been the foundation from which all pairing algorithms have been constructed. In 2006, an alternative technique was proposed by Stange [126]. Stange's method, which is based on elliptic nets, is less efficient than Miller's algorithm requiring more field multiplications.

Miller's algorithm consists of a double and add algorithm for elliptic curve point multiplication with additional functionality. The input points P and Q to Miller's algorithm are chosen such that the point P is a point of order r. During the point scalar multiplication of the point $[r]P$, which upon completion should result in $\mathcal{O}$, a distance relationship between the lines produced from the point addition and a static point Q is calculated. If $l_{A,B}$ and v_{A+B} are the diagonal and vertical lines which arise in the addition rule for adding $A = [a]P$ to $B = [b]P$ to produce $C = A + B$, and the point A has coordinates (x_i, y_i), the point C has coordinates (x_{i+1}, y_{i+1}), the point Q has coordinates (x_Q, y_Q), and the line through A and B has a slope of λ_i, then explicitly

$$\begin{aligned} l_{A,B}(Q) &= (y_Q - y_i) - \lambda_i(x_Q - x_i) \\ v_{A+B}(Q) &= (x_Q - x_{i+1}) \end{aligned}$$

In each round of the Miller loop, $l_{A,B}$ is divided by v_{A+B} to produce an element in the

extension field $m \in \mathbb{F}_{q^k}$ referred to as the Miller variable. This value is multiplicatively accumulated and eventually outputted from Miller's algorithm. Miller's algorithm is given in Algorithm 1.

Algorithm 1 Miller's Algorithm [88]

INPUT: $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(\mathbb{F}_{q^k})$
OUTPUT: $m \in \mathbb{F}^*_{q^k} \mid m^r = 1$
1: $T \leftarrow P, m \leftarrow 1$
2: **for** $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ **to** 0 **do**
3: $\quad m \leftarrow m^2 \cdot l_{T,T}(Q)/v_{2T}(Q)$
4: $\quad T \leftarrow 2T$
5: $\quad$ **if** $r_i = 1$ **then**
6: $\quad\quad m \leftarrow m \cdot l_{T,P}(Q)/v_{T+P}(Q)$
7: $\quad\quad T \leftarrow T + P$
8: $\quad$ **end if**
9: **end for**
10: **return** m

The Weil pairing is calculated with two applications of Miller's algorithm. The output from the two calls to Miller's algorithm are then divided to produce the bilinear map value. Calculation of the Tate pairing requires only one application of Miller's algorithm. In this case the output from Miller's algorithm is raised to the power $(q^k - 1)/r$ to produce the bilinear map value.

A vital requirement for the successful computation of all pairings is that the input points P and Q be linearly independent. If they are linearly dependent the bilinear map value will degenerate to 1. This can however be ensured through special choice of input points. Improvements to Miller's algorithm will be discussed next.

3.5 Practical Pairing Implementations

The efficient computation of pairings is essential to the efficiency and employability of the related pairing based cryptosystem. The general definitions of the Weil and Tate pairing were given in the previous section. However when implemented in their original form, they are uneconomical and time-wise, unattractive. Since their definition, many endeavours have been made to produce efficient pairing algorithms. These advances have

lead to the development of extremely efficient pairing implementations. Some of the main advances in the optimisations of pairings and the resulting pairing algorithms, will be presented in this section. Specifically, five types of pairings algorithms, which are relevant for power and fault analysis of pairings in later chapters, will be presented.

3.5.1 Optimisations

Some of the main advances in the optimisation of pairings are as follows. The choice of elliptic curve and underlying field affect the efficiency of the pairing. Choosing an elliptic curve and finite field, in particular where efficient algorithms for finite field arithmetic are optimised, can immediately make the pairing more economical.

Special choice of the input parameters P and Q can result in more efficient finite field operations. In practice one often works with specific subgroups to speed up the pairing computation. For example, choosing an elliptic curve point as an element of the base field $E(\mathbb{F}_q)$ as opposed to the extension field $E(\mathbb{F}_{q^k})$ will result in arithmetic being performed over the smaller and more efficient base field as opposed to the larger extension field. Both parameters however, cannot be chosen from the same group since P and Q must be linearly independent in order for the pairing not to degenerate. The modified Tate pairing satisfies this requirement and defines a pairing where both P and Q are efficiently chosen. The modified Tate pairing is specifically defined for supersingular elliptic curves since a distortion map (or non-rational endomorphism), denoted ϕ, allows elements from the base field $E(\mathbb{F}_q)$ to be mapped to elements in the extension field $E(\mathbb{F}_{q^k})$. For example, for $E(\mathbb{F}_p) : y^2 - x^3 + ax$ where $p \equiv 3 \pmod 4$ the distortion map $(x, y) \mapsto (-x, iy)$ where $i^2 = -1$ exists. The modified Tate pairing is defined as

$$\hat{e} : E(\mathbb{F}_q)[r] \times E(\mathbb{F}_q) \rightarrow \mu_r$$

and so

$$\hat{e}(P,Q) = \langle P, \phi(Q) \rangle_r^{(q^k-1)/r} \tag{3.5}$$

where the distortion map $\phi : E(\mathbb{F}_q) \rightarrow E(\mathbb{F}_{q^k})$ allows the parameter Q to be chosen as a point on the elliptic curve over the base field while still ensuring that P and $\phi(Q)$ are linearly independent. Note that $\phi(Q) \notin E(\mathbb{F}_q)[r]$ and thus is linearly independent of P. In addition to the advantageous efficient arithmetic over the smaller field, if the distortion map is chosen so that the x coordinates always lie in a subfield (for example $\mathbb{F}_q$ of $\mathbb{F}_{q^k}$), then all terms $v_{2T}(Q)$ and $v_{T+P}(Q)$ in Miller's algorithm (Algorithm 1) can be eliminated. As a result, the divisions in Miller's algorithm can be dropped. This optimisation is known as denominator elimination and was proposed by [14].

Special choice of the order r also has implications for the pairings efficiency. Miller's algorithm involves a number of arithmetic operations proportional to the Hamming weight of r. Therefore, it is advantageous to choose r with low Hamming weight. For example, Solinas primes which have the form $2^a \pm 2^b \pm 1$, where a and b are coefficients from the elliptic curve equation, have low Hamming weight [125]. Choosing r as a Solinas prime will minimise the number of additions performed in the point scalar multiplication of $[r]P$ in the Miller loop. Instead of choosing r with low Hamming weight, another idea is to choose r as a multiple of the group order, that is if that multiple should have a lower Hamming weight. If r is chosen in this way then r must evenly divide $q^k - 1$ to give a smaller factor, thus resulting in reducing the complexity of the final exponentiation.

3.5.2 The Weil Pairing

The Weil pairing $\omega(P,Q)$ in its original definition consists of two applications of Miller's algorithm. The first application is with the point scalar multiplication of the point $[r]P$ and the distance relationship being calculated to the static point Q. The second is with the point scalar multiplication of the point $[r]Q$ and the distance relationship being calculated to the static point P. This is why both input points P and Q must be chosen so that they

both are of order r. The output of both applications of Miller's algorithm, say m_1 and m_2, are divided m_1/m_2 to produce the bilinear pairing. No final exponentiation is required for the Weil pairing.

The Weil pairing is most notably associated in the cryptanalysis sense with the MOV attack, where the complexity of solving the Discrete Logarithm Problem (DLP) on a group of points on an elliptic curve is reduced to solving the DLP over a finite field [81], and in the cryptographic sense with Boneh and Franklin's Identity Based Encryption (IBE) [22]. However, it is the least efficient pairing algorithm. Its sibling Tate is more efficient, and so has received greater attention by the research community. In this thesis, a variant of the original Weil pairing will be considered [117]. By introducing a simple final exponentiation, the denominator elimination optimisation can be exploited. Algorithm 2 and 3 describe the variant of the Weil algorithm considered here. To distinguish this version from the original, it will be referred to as ω_D as opposed to ω.

The implementation assessed in this research was implemented over non-supersingular elliptic curves of prime characteristic $E(\mathbb{F}_p)$ (see Section 3.5.3). The embedding degree $k = 2$ and p is chosen as a 512 bit prime so as to achieve 1024 bit security. Elements in the extension field $\mathbb{F}_{p^2}$ are represented as polynomials $a + xb$ with $a, b \in \mathbb{F}_p$ and irreducible polynomial $x^2 + 1$. Elements in $\mathbb{F}_{p^2}$ can be multiplied as normal and then reduced modulo $x^2 + 1$. If p is chosen such that it is congruent to 3 $(\mathrm{mod}\ 4)$, then -1 is a quadratic non residue modulo p. In the occurrence of x^2 in the multiplication of $\mathbb{F}_{p^2}$ elements, x^2 can be replaced with -1. This means that x can be considered as the imaginary root of the irreducible polynomial, in which case elements of the extension field can be represented as $a + ib$ where i is the imaginary square root of -1. An additional advantage of choosing $p \equiv 3 \pmod 4$ is that a simple method exists to compute the square root, i.e. $\sqrt{x} = x^{(p+1)/4} \pmod p$, which is advantageous for point compression. The additional exponentiation appended to this Weil variant, $p - 1$, is actually very cheap due to the

Frobenius action;

$$\begin{aligned} (a+ib)^p &= (a^p + i^p b^p) = (a - ib) \pmod p \\ \Rightarrow (a+ib)^{p-1} &= \frac{(a-ib)}{(a+ib)} \pmod p \end{aligned} \tag{3.6}$$

requiring only a conjugation and division, where $\bar{m} = a - ib$ is the conjugate of $m = a + ib$.

Algorithm 2 This algorithm, referred to as the g function, adds elliptic curve points and calculates the most recent contribution to the Miller variable.

INPUT: $A = (x_A, y_A), B = (x_B, y_B), C = (x_C, y_C), D = (x_D, y_D), P = (x_P, y_P), Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^2})$
OUTPUT: $g \in \mathbb{F}_{p^2}$ and updates A and C
1: $(A', \lambda_1) \leftarrow A + B$
2: $(C', \lambda_2) \leftarrow C + D$
3: $u \leftarrow \lambda_1(x_A + x_Q) - y_A + y_Q i$
4: $v \leftarrow y_P + (y_C - \lambda_2(x_C + x_P))i$
5: $A \leftarrow A', C \leftarrow C'$
6: **return** $g \leftarrow u * v$

Algorithm 3 Computation of $\omega_D(P, Q)$ on $E(\mathbb{F}_p) : y^2 = x^3 + ax + b$ [117].

INPUT: $P = (x_P, y_P) \in E(\mathbb{F}_p)[r], Q = (x_Q, y_Q) \in E'(\mathbb{F}_p)$
OUTPUT: $m \in \mathbb{F}_{p^2}^* \mid m^r = 1$
1: $m \leftarrow 1, A \leftarrow P, C \leftarrow Q$
2: **for** $i \leftarrow \lfloor \log_2(r) \rfloor - 2$ **to** 0 **do**
3: $\quad m \leftarrow m^2 * g(A, A, C, C, P, Q)$
4: $\quad$ **if** $r_i = 1$ **then**
5: $\quad\quad m \leftarrow m * g(A, P, C, Q, P, Q)$
6: $\quad$ **end if**
7: **end for**
8: **return** $m \leftarrow \frac{\bar{m}}{m}$

3.5.3 The Tate Pairing

The Tate pairing in contrast to Weil, requires only one application of Miller's algorithm. This is coupled with a final exponent of $(q^k - 1)/r$ to produce a bilinear map value. This is why the point of order r requirement is only enforced on the first input point P. As mentioned previously, the only restriction for Q is that it is linearly independent of P.

A number of algorithms have been developed to calculate the Tate pairing, the two

most notable being the Duursma-Lee algorithm [37] and the BKLS algorithm [14]. The Duursma-Lee algorithm for computing the Tate pairing was originally introduced to compute a pairing on a family of hyperelliptic curves, and supersingular elliptic curves in characteristic three $E(\mathbb{F}_{3^m})$. The BKLS algorithm (Barretto, Kim, Lynn and Scott) [14] was also developed for characteristic three supersingular elliptic curve, but more recently has been associated mostly with non-supersingular elliptic curves of prime characteristic $E(\mathbb{F}_p)$ [118]. The Duursma-Lee algorithm will be briefly mentioned in Chapter 5 and 6, however, it is the BKLS algorithm [118] that will be the focus of the analysis in this thesis. Both algorithms are given here for completeness. Algorithm 4 and 5 describe the BKLS algorithm for the Tate pairing and Algorithm 6 describes the Duursma-Lee algorithm.

The BKLS algorithm is an extremely efficient implementation of the Tate pairing in which denominator elimination is employed, the input points P and Q are chosen such that finite field arithmetic over the base field can be exploited, and r is chosen as a Solinas prime with low Hamming weight [125]. The implementation assessed in this research was implemented over non-supersingular elliptic curves of prime characteristic $E(\mathbb{F}_p)$ with embedding degree $k = 2$.

The modified Tate pairing does not apply to non-supersingular elliptic curves, hence no distortion map exists to map $Q \in E(\mathbb{F}_p)$ to a point on $E(\mathbb{F}_{p^k})$. Another technique for non-supersingular elliptic curves, which allows the point Q to be on $E(\mathbb{F}_p)$, involves the use of the twist of the elliptic curve, denoted by E'. Every elliptic curve has a quadratic twist. However, the twist becomes useful when both the original curve E and the twist E' are defined over the quadratic extension field $\mathbb{F}_{p^2}$, since the twist $E'(\mathbb{F}_p)$ is isomorphic to $E(\mathbb{F}_{p^2})$. This allows the point Q to be chosen as a point in $E'(\mathbb{F}_p)$, which is subsequently mapped to a point in $E(\mathbb{F}_{p^2})$. Note that choosing Q as an element over a smaller field actually has no implications for the efficiency of the pairing, since in the algorithm Q is operated on as a quadratic field element. Choosing Q as an element over a smaller field is only useful for the transmission and sharing of Q in cryptographic protocols.

The final exponentiation for the Tate pairing involves raising the output of the Miller loop to the power of $(p^k-1)/r$. By breaking $(p^k-1)/r$ into its factors $(p^d-1)(p^d+1)/r$, where k is even and $k = 2d$, exponentiation in parts can be performed. Exponentiation to the factor $p^d - 1$ is straightforward due to the Frobenius action (Equation (3.6)). The exponentiation to additional factors can be facilitated by fast exponentiation methods such as windowing or NAF of the exponent. Elements in the extension field $\mathbb{F}_{p^2}$ are represented as $a + ib$ for the same reasons as above. The modulus p is chosen as a 512 bit prime so that 1024 bit security is achieved, as the "security multiplier" $k = 2$.

Algorithm 4 This algorithm, referred to as the g function, adds elliptic curve points and calculates the most recent contribution to the Miller variable.

INPUT: $A = (x_A, y_A), B = (x_B, y_B), Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^2})$
OUTPUT: $g \in \mathbb{F}_{p^2}$ and updates A
1: $(A', \lambda_1) \leftarrow A + B$
2: $u \leftarrow y_A - \lambda_1(x_Q + x_A) - y_Q i$
3: $A \leftarrow A'$
4: **return** $g \leftarrow u$

Algorithm 5 Computation of $e(P, Q)$ on $E(\mathbb{F}_p) : y^2 = x^3 + ax + b$ [14].

INPUT: $P = (x_P, y_P) \in E(\mathbb{F}_p)[r], Q = (x_Q, y_Q) \in E'(\mathbb{F}_p)$
OUTPUT: $m \in \mathbb{F}_{p^2}^* \mid m^r = 1$
1: $m \leftarrow 1, A \leftarrow P$
2: **for** $i \leftarrow \lfloor \log_2(r) \rfloor - 2$ **to** 0 **do**
3: $\quad m \leftarrow m^2 * g(A, A, Q)$
4: $\quad$ **if** $r_i = 1$ **then**
5: $\quad\quad m \leftarrow m * g(A, P, Q)$
6: $\quad$ **end if**
7: **end for**
8: $m \leftarrow \frac{\bar{m}}{m}$
9: **return** $m^{(p+1)/r}$

3.5.4 The η Pairing

The η pairing $\eta(P, Q)$ generalises the results of the Duursma-Lee algorithm for the Tate pairing for supersingular curves in characteristic three. The η pairing specialises in pairings over supersingular curves of small characteristic. The main distinction between the η pairing and its siblings is that it chooses the order of the Miller loop r as a multiple of

Algorithm 6 Duursma-Lee Algorithm for $E(\mathbb{F}_q) : y^2 = x^3 - x + b$ where $b = \pm 1$ and $q = 3^m$ [107]

INPUT: $P = (x_1, y_1), Q = (x_2, y_2) \in E(\mathbb{F}_{q^6})$
OUTPUT: $m \in \mathbb{F}^*_{q^6} \mid m^r = 1$

1: $m \leftarrow 1$
2: **for** $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ **to** 0 **do**
3: $\quad x_1 \leftarrow x_1^3$
4: $\quad y_1 \leftarrow y_1^3$
5: $\quad \mu \leftarrow x_1 + x_2 + b$
6: $\quad \lambda \leftarrow -y_1 \cdot y_2 \sigma - \mu^2$
7: $\quad g \leftarrow \lambda - \mu\rho - \rho^2$
8: $\quad m \leftarrow m \cdot g$
9: $\quad x_2 \leftarrow x_2^{1/3}$
10: $\quad y_2 \leftarrow y_2^{1/3}$
11: **end for**
12: **return** m^{q^3-1}

the group order such that it divides $q^k - 1$ nicely to give a small factor. This results in simplifying the final exponentiation. For example if $q^k - 1 = 2^{4m} - 1$ and $r = 2^{2m} + 1$, then the final exponentiation basically involves a conjugation and division. In addition the formulae for point addition and point double is explicitly derived and built into the algorithm.

Recently Galbraith et al. [41] proposed a variant of the η pairing requiring no final exponentiation. This is enabled by reintroducing line function evaluations $v()$, which the original η [13] does not have. It is this version of the η pairing that will be considered in this thesis, and so will be referred to as the η_G pairing to distinguish it from the original η pairing. Algorithms 7 and 8 describe the η_G pairing.

The implementation assessed in this research was implemented over supersingular curves of characteristic two. The embedding degree $k = 4$ and $m = 271$. Elements in the extension field $E(\mathbb{F}_{2^{4m}})$ are represented as polynomials $a_0 + a_1 x + a_2 x^2 + a_3 x^3$ with $a_i \in \mathbb{F}_{2^m}$ and the irreducible polynomial $x^4 + x + 1$. Elements in $E(\mathbb{F}_{2^{4m}})$ can be multiplied as normal and then reduced modulo $x^4 + x + 1$. In practice, the coefficients of elements in $\mathbb{F}_{2^{4m}}$ are stored in four different cells in memory. The notation $[a_0][a_1][a_2][a_3]$ is defined to represent the storage of each of these elements.

Algorithm 7 This algorithm, referred to as the g function, adds elliptic curve points and calculates the most recent contribution to the Miller variable.

INPUT: $A = (x_A, y_A), C = (x_C, y_C), Q = (x_Q, y_Q) \in E(\mathbb{F}_{2^m})$
OUTPUT: $g \in \mathbb{F}_{2^{4m}}$ and updates A
1: $(A', \lambda_1) \leftarrow A + C$
2: $l \leftarrow [y_Q + y_A + \lambda_1(x_Q + x_A + 1)][\lambda_1 + x_Q + 1][\lambda_1 + x_Q][0]$
3: $v \leftarrow [x_Q + x_C + 1][1][1][0]$
4: $A \leftarrow A'$
5: **return** $g \leftarrow \frac{l}{v}$

Algorithm 8 Computation of $\eta_G(P, Q)$ on $E(\mathbb{F}_{2^m}) : y^2 + y = x^3 + x + b$ [41].

INPUT: $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(\mathbb{F}_{2^m})$
OUTPUT: $m \in \mathbb{F}^*_{2^{4m}} \mid m^r = 1$
1: $m \leftarrow 1, A \leftarrow P$
2: **for** r **to** 0 **do**
3: $\quad m \leftarrow m^2 * g(A, A, Q)$
4: **end for**
5: **return** m

3.5.5 The η_T Pairing

The η_T pairing $\eta_T(P, Q)$ is a truncated version of the η pairing, that is the number of iterations of Miller's loop is reduced. The BGOhES algorithm (Barreto, Galbraith, O'hEigeartaigh and Scott) [13] is used to calculate the η_T pairing.

The particular implementation assessed here was implemented over supersingular curves of characteristic two $E(\mathbb{F}_{2^m})$. The embedding degree $k = 4$ and $m = 379$. Both parameters input to the pairing, P and Q, are elements in $E(\mathbb{F}_{2^m})$. The field $\mathbb{F}_{2^{4m}}$ has elements s, t such that $s^2 = s + 1$ and $t^2 = t + s$. Elements in $\mathbb{F}_{2^{4m}}$ will be represented using the basis $1, s, t, st$. To map a point from the base field to a point on the quartic extension field $\mathbb{F}_{2^{4m}}$, the distortion map $(x, y) = (x + s2, y + sx + t)$ is used. The distortion map and the formulae for point addition and doubling are explicitly derived and built into the algorithm. Elements in the extension field $E(\mathbb{F}_{2^{4m}})$ are represented as before as polynomials $a_0 + a_1x + a_2x^2 + a_3x^3$ with $a_i \in \mathbb{F}_{2^m}$ and the irreducible polynomial is $x^4 + x + 1$. Algorithm 9 describes the η_T pairing.

Algorithm 9 Computation of $\eta_T(P,Q)$ on $E(\mathbb{F}_{2^m}) : y^2 + y = x^3 + x + b$ [13].

INPUT: $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(\mathbb{F}_{2^m})$
OUTPUT: $f \in \mathbb{F}^*_{2^{4m}} \mid f^r = 1$

1: $u \leftarrow x_P + 1$
2: $f \leftarrow u \cdot (x_P + x_Q + 1) + y_P + y_Q + b + 1 + (u + x_Q)s + t$
3: **for** $i \leftarrow 1$ **to** $(m+1)/2$ **do**
4: $\quad u \leftarrow x_P, x_P \leftarrow \sqrt{x_P}, y_P \leftarrow \sqrt{y_P}$
5: $\quad g \leftarrow u \cdot (x_P + x_Q) + y_P + y_Q + x_P + (u + x_Q)s + t$
6: $\quad f \leftarrow f \cdot g$
7: $\quad x_Q \leftarrow x_Q^2, y_Q \leftarrow y_Q^2$
8: **end for**
9: **return** $f^{(2^{2m}-1)(2^m - 2^{(m+1)/2}+1)(2^{(m+1/2)}+1)}$

3.5.6 The Ate Pairing

The Ate pairing [50] $a(P,Q)$ is the most recently discovered pairing algorithm. It is related both to the Tate and the η pairing. The Ate pairing is a mapping

$$a : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_q) \rightarrow \mu_r$$

Note that Ate prefers to make the first parameter P over the larger field, which is the opposite to the Tate pairing. In particular, $P \in E'(\mathbb{F}_{p^{k/d}})$ and the second parameter $Q \in E(\mathbb{F}_p)$, where d is the degree of the twist. Here only the quadratic twist is considered, hence $d = 2$. This affects the point scalar multiplication of $[r]P$, which is now calculated over the extension field $E'(\mathbb{F}_{p^{k/d}})$. If $\lg(t)/\lg(r) < 1$ where t is the trace of Frobenius, then the Ate pairing is faster than the Tate pairing for non-supersingular curves $E(\mathbb{F}_p)$, since a truncated loop of length $\lg(t)$ is used instead of $\lg(r)$. To avail of this shorter Miller loop however, an elliptic curve with special properties must be used. Algorithms 10 and 11 describe the Ate pairing.

The implementation assessed in this research was implemented over non-supersingular elliptic curves of prime characteristic $E(\mathbb{F}_p)$. The embedding degree $k = 4$ and so $d = 2$ since $k = 2d$. The modulus p is chosen as a 512 bit prime so that 2048 bit security is achieved. There are two ways to represent elements in $\mathbb{F}_{p^4}$. Either as a quartic $a + xb + cx^2 + dx^3$ with $a, b, c, d \in \mathbb{F}_p$ or as a quadratic built on top of a quadratic $a + bx$

where a and $b \in \mathbb{F}_{p^2}$. The second method is more efficient for implementations. If p is chosen such that it is congruent to 5 (mod 8), then -2 is a quadratic non residue in $\mathbb{F}_p$ and $\sqrt{-2}$ is a quadratic non residue in $\mathbb{F}_{p^2}$. This allows elements in $\mathbb{F}_{p^4}$ to be represented as a pair of elements in $\mathbb{F}_{p^2}$, $m = m_R + im_I$ with $i = (-2)^{1/4}$. In Algorithm 10, which calculates the Miller variable contribution, the points on the twisted curve $E'(\mathbb{F}_{p^2})$ must be converted to elements in $E(\mathbb{F}_{p^4})$.

Algorithm 10 This algorithm, referred to as the g function, adds elliptic curve points and calculates the most recent contribution to the Miller variable.

INPUT: $A = (x_A, y_A), B = (x_B, y_B), Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^4})$
OUTPUT: $g \in \mathbb{F}_{p^4}$ and updates A
1: $(A', \lambda_1) \leftarrow A + B$
2: $u \leftarrow i^2 y_Q - i(i^2 y_A/2 + \lambda_1(i^2 x_A/2 + x_Q))$
3: $A \leftarrow A'$
4: **return** $g \leftarrow u$

Algorithm 11 Computation of $a(P, Q)$ on $E(\mathbb{F}_p) : y^2 = x^3 + ax + b$ [50].

INPUT: $P = (x_P, y_P) \in E'(\mathbb{F}_{p^2})[r], Q = (x_Q, y_Q) \in E(\mathbb{F}_p)$
OUTPUT: $m \in \mathbb{F}^*_{p^4} \mid m^r = 1$
1: $m \leftarrow 1, A \leftarrow P$
2: $n \leftarrow t - 1$
3: **for** $i \leftarrow \lfloor \log_2(n) \rfloor - 2$ **to** 0 **do**
4: $\quad m \leftarrow m^2 * g(A, A, Q)$
5: $\quad$ **if** $n_i = 1$ **then**
6: $\quad\quad m \leftarrow m * g(A, P, Q)$
7: $\quad$ **end if**
8: **end for**
9: $m \leftarrow \frac{\bar{m}}{m}$
10: **return** $m^{p^2+1/r}$

3.6 Cryptographic Applications of Bilinear Pairings

As mentioned previously, following the introduction of pairings in a constructive manner, a large amount of attention has been devoted to using bilinear pairings to build cryptosystems with novel properties. For example, a myriad of key agreement schemes, signature schemes and encryption schemes have been proposed. A survey of the many schemes can be found in [36]. In the context of this research, bilinear pairings where one of the

parameters input to the pairing is a secret are of particular interest. As motivation, a brief description of an important pairing based protocol is given, namely the IBE scheme of Boneh and Franklin.

The concept behind identity based cryptography was conceived in 1984 when Shamir [121] called for a public key, identity based encryption scheme in which the public key can be an arbitrary string. Shamir's original motivation for this scheme was to simplify the management of certificates in email systems. For example, if Alice wants to send an encrypted email to Bob, she encrypts the message using Bob's public key string, which simply corresponds to his email address. This differs considerably from traditional certificate based schemes, allowing the public key to be derived in a deterministic way from the user's identity parameters. When Bob receives the encrypted message he contacts a third party, known as the Private Key Generator (PKC). Bob authenticates himself to the PKG and obtains his private key. Bob can now decrypt the encrypted message received from Alice.

The first concrete implementation of IBE however, did not appear until 2001 when Boneh and Franklin [22] proposed an IBE scheme based on bilinear maps. The basic idea of Boneh and Franklin's IBE scheme is presented here.

Let $\omega : G_1 \times G_1 \rightarrow G_2$ be a bilinear pairing. In the context of pairings as just described G_1 is a group of points on an elliptic curve over a finite field, and G_2 is the underlying multiplicative finite field. Originally, Boneh and Franklin's IBE scheme was defined with the Weil pairing. However, any pairing definition can be used here. Let $H_1 : \{0,1\}^* \rightarrow G_1$ and $H_2 : G_2 \rightarrow \{0,1\}^l$ be cryptographic hash functions where l is the bit length of the plaintext m to be encrypted. The PKG selects a private key s at random and computes its public key as $Q = [s]P$, where $P \in G_1$.

Bob's private key is $D_B = [s]Q_B$ where $Q_B = H_1(ID_B)$ and ID_B is the public key string, such as Bob's email address, associated with Bob's identity. Note that to compute D_B, given P, Q and Q_B, is an instance of the Diffie Hellman Problem (DHP), in the

group G_1. The DHP is the problem of given $P, [x]P$ and $[y]P$, calculate $[z]P = [xy]P$ [77]. Hence, only the PKG can calculate D_B since it has access to the secret value s. Alice encrypts a message $m \in \{0,1\}^l$, by first randomly selecting an integer r and then computing the following values

$$\begin{aligned} Q_B &= H_1(ID_B) \\ C_1 &= [r]P \\ C_2 &= m \oplus H_2(\omega(Q_B, Q)^r) \end{aligned}$$

and sending the ciphertext pair (C_1, C_2) to Bob. Bob can recover the original message m from the ciphertext (C_1, C_2) by using his private key D_B to compute

$$m = C_2 \oplus H_2(\omega(D_B, C_1)).$$

To see how the bilinear property of the pairing enables the decryption of the ciphertext, observe that

$$\omega(D_B, C_1) = \omega([s]Q_B, [r]P) = \omega(Q_B, [s]P)^r = \omega(Q_B, Q)^r.$$

In the non-side channel and fault attack sense, an adversary who attempts to recover m from the ciphertext (C_1, C_2) has to compute $\omega(Q_B, Q)^r$ from (P, Q_B, Q, C_1), which is an instance of the Bilinear Diffie-Hellman Problem (BDHP) (see Section 3.7).

As can be seen here, it is the bilinear pairing in the decryption operation which relies on the private key D_B. If this decryption operation is executed on a device such as a smart card, the pairing operation, since it involves the private key, will be a target for implementation attacks.

3.7 Associated Security of Bilinear Pairings

The development of cryptosystems based on elliptic curves has seen the definition of various hard problems and related security concepts upon which the security of the elliptic curve cryptosystems are based. A number of security concepts are associated with bilinear pairings in particular. The relevant concepts will be briefly mentioned here.

In the case of IBE, where D_B is Bob's private key, the attack on this scheme will consist of the adversary trying to find D_B given (P, Q_B, Q, C_1). This is an instance of the Bilinear Diffie-Hellman Problem (BDHP) which is the problem of computing $e(P, P)^{xyz}$ given $P, [x]P, [y]P, [z]P \in G_0$. To solve the BDHP appears to require the adversary to be able to extract x, y and z from $[x]P$, $[y]P$ and $[z]P$, which are instances of the Elliptic Curve Discrete Log Problem (ECDLP), an accepted hard problem to solve.

When considering implementation attacks, these established security levels are null and void since the adversary uses additional information, such as side channel leakage like the power consumption or faulty pairing computation, to gain insights. Therefore, these established security concepts are irrelevant when considering power and fault analysis since pairings must be considered under new criteria. This is one of the main objectives of this thesis.

3.8 Power and Fault Analysis of Elliptic Curve Primitives

Elliptic curve cryptosystems have been open to much scrutiny in terms of implementation attacks such as power and fault analysis. Avanzi [9] presents a good overview of how elliptic curve primitives have been attacked with SCA. Also chapters 27, 28 and 29 of [30] provide a comprehensive overview on this topic.

On the power analysis front, many successful attacks on the point scalar multiplication of $[n]P$ where the scalar n is the secret, have been documented. For example, the classic double and add algorithm for point scalar multiplication is inherently vulnerable to SPA

since the execution path is dependent on key bits [31]. This is particularly vulnerable since the formulae for point addition and point double differ, and so are differentiable in the power trace. Several alterations to the algorithm for point scalar multiplication have been proposed which effectively make the algorithm behave uniformly. For example, the double and add always algorithm which carries out dummy operations [31] and unified code for point addition and doubling [134] have been proposed to hide the presence or absence of point additions, since the knowledge of this information indicates a 1 or a 0 in the secret key n. However, these attempts have proved to be still open to attack as demonstrated by [134] and [128].

Power analysis attacks such as DPA and CPA, have been equally successful on both the classic double and add algorithm and the double and add algorithms borne out of SPA attacks. Common countermeasures to deter these more powerful attacks include randomisation, which can be easily incorporated using the basic field properties. In particular, measures such as random projective coordinates, random elliptic curve isomorphisms or random field isomorphisms can be employed without much alteration of the algorithm. However, these so-called protected algorithms have also proved vulnerable to high-order DPA as demonstrated by [45].

Fault attacks have received less attention than their power analysis attack counterparts. The attack on the point scalar multiplication algorithm has again been prominent, where the scalar is the secret. The main types of attacks that have been developed have sought to corrupt an elliptic curve point involved in the computation so that it no longer lies on the correct curve. For example, inputting faulty elliptic curve points which are not on the prescribed elliptic curve can under certain circumstances reveal the secret [18]. By creating a fault during the computation of $[n]P$, then if the faulty output is not checked to be a point on the elliptic curve, then the secret key can be revealed [18]. Simple checks in these situations, such as checking whether $P \in E$, can detect whether there has been an attack or not, and so suffices to prevent the adversary from extracting the key. However,

this detection mechanism is not sufficient to catch all types of faults. An attack described in [20], targets the sign of the elliptic curve point. For example, a fault corrupts the point (x, y) so it becomes $(x, -y)$. This faulty point is a valid element on the elliptic curve, and so will not be detected.

As described, the majority of research has focused on the scenario where the target operation is the point scalar multiplication of the point $[n]P$ and n is the secret scalar of interest. In pairings however, the scalar r which is required for point scalar multiplication of the point $[r]P$, is the order of the point P, and is a public value. Therefore, the research that has been performed on elliptic curve primitives is not relevant to pairings. In pairing based protocols, such as the IBE described in Section 3.6, it is the elliptic curve point input to the pairing which is the secret, and so the value of interest.

3.9 Conclusion

In this chapter, the relevant background on bilinear pairings was provided. Mathematical definitions and algorithm descriptions were given of the candidate pairing implementations to be assessed in later chapters. In the next chapter, the process of power and fault analysis will be investigated in more detail. Subsequently, the application of these techniques to the pairing algorithms discussed here will be addressed.

Chapter 4

Further Analysis of Power and Fault Attacks

4.1 Introduction

In this chapter, a number of aspects of implementation attacks are examined in more detail. In particular, work that was performed during the familiarisation period of the research area is detailed.

Firstly, a fault attack on the Digital Signature Algorithm (DSA) is described. The DSA is an asymmetric algorithm that is central to the Digital Signature Standard (DSS). The DSA provides authentication, integrity and non-repudiation [97]. A number of theoretical attacks on the DSA, based on lattice reduction techniques, are widely known and have been reasonably researched over the previous years. Most of these attacks assume that there is some information known about the nonce, which is used in the generation of the DSA signature. One obvious technique to gain information on the nonce is to manipulate the generation of the nonce using a fault attack. However, most of the attacks in the literature neglect to deal with a vital aspect of the attack, how the fault was actually injected in the first place. By performing a glitch attack on a smart card implementing the

DSA, the generation of the nonce can be corrupted in such a way that a number of bits are set to zero. This provides the adversary with the necessary information about the nonce so that theoretical findings of previous work can be used to extract the entire private key [94, 51]. In Section 4.2, the first experimental results of this attack are described.

Next, the process of performing Differential Power Analysis (DPA) is examined. This is generally a lengthy process. The data acquisition phase alone can take a long period of time, depending on a variety of practical factors, such as the duration of the monitored process, the oscilloscope to PC transfer rate, the equipment's sampling frequency, the number of power traces necessary to overpower the target's signal to noise ratio, etc. Then, when the data collected are passed to the data analysis phase, the data are manipulated in various ways to derive information on the key used. This is a tedious and time consuming process, involving repeated operations that can be improved upon. In Section 4.3, a number of computational improvements to the data analysis phase of a SCA are proposed. These improvements, which do not affect the attack's precision or probability of success, are described in terms of DPA [66]. The methods described are applicable to other forms of SCA, such as Differential Electromagnetic Analysis (DEMA) [111, 42], where once the acquisition stage of the attack is complete, the data analysis phase is exactly the same as DPA. As an illustrative example, assuming that a classic DPA of a given device requires approximately 64,000 power trace operations, the techniques described would obtain exactly the same results by spending only 3% of the computational effort (namely approximately 2100 operations).

Section 4.2 describes work that was performed in collaboration with Phong Nguyễn, Mike Tunstall and David Naccache and was published at the Public Key Cryptography conference in 2005 [95]. Section 4.3 describes work that was performed in collaboration with Mike Tunstall and David Naccache and was published at NATO Advanced Research Workshop on Security and Embedded Systems in 2005 [92]. Both projects were performed in Gemplus research and development laboratories in La Ciotat, France, using

custom built Gemplus tools [52].

4.2 Fault Attack on the DSA: First Experimental Results

In this section, the different stages that are required to execute a fault attack on the DSA are described. Firstly, DSA signature generation and verification is reviewed.

4.2.1 DSA Signature and Verification

The system parameters for the DSA [97] are $\{p, q, g\}$, where p is prime (between 512 and 2048 bits), q is a 160 bit prime dividing $p-1$ and g is an element of order q in the group $\mathbb{F}_p^*$, i.e. $g^q \equiv 1 \pmod{p}$. Each of these values is public information. The private key is an integer $\alpha \in \mathbb{F}_p^*$ and the public key is the group element $\beta = g^\alpha \pmod{p}$.

Signature: To sign a message μ, the signer picks a random $k < q$ known as the nonce and computes

$$r \leftarrow \left(g^k \pmod{p}\right) \pmod{q} \quad \text{and} \quad s \leftarrow \frac{\mathrm{H}(\mu) + \alpha r}{k} \pmod{q}$$

where H is a hash function. The signature of μ is the pair: (r, s).

Verification: To check (r, s) the verifier ascertains whether

$$r \stackrel{?}{=} \left(g^{wh}\beta^{wr} \pmod{p}\right) \pmod{q} \quad \text{where} \quad w \leftarrow \frac{1}{s} \pmod{q} \quad \text{and} \quad h \leftarrow \mathrm{H}(\mu)$$

4.2.2 Attack Overview

The attack on DSA proceeds as follows. Firstly, several DSA signatures are generated where the random value generated for k has been modified so that a few of k's least

significant bits[1] are reset to zero[2]. This faulty k will then be used to generate a DSA signature. Using lattice reduction (see Section 4.2.6), the secret key α can be recovered from a collection of such signatures.

4.2.3 Experimental Conditions

The first question that arises in such an attack is how will this very specific fault attack be realised in practice. To determine the viability of such a targeted attack, the generation of k was implemented alone in a closed environment and analysed to determine the optimal locations for fault injection. To generate k, a random number generator is used to generate a 160 bit number. This number, the nonce k, is then compared to q to ensure that it is in $\mathbb{F}_q^*$. If $k \geq q-1$, then the nonce is discarded and a new k is generated. Two approaches can be used to analyse the generation of k. The first approach is to use Simple Power Analysis (SPA) to characterise the power trace and identify when the random number generator is in operation. The second is to purposely include special commands in the implementation to indicate the start and end of the generation of k, thereby indicating the time in which to inject the fault. The technique detailed in [95] used the latter approach. A typical code fragment to generate k is as follows. Note that `acCoproMessage[]` is used to hold k as it is being generated.

```
PutModulusInCopro(PrimeQ);
RandomGeneratorStart();
status = 0;
do {
  IOpeak();
  for (i=0; i<PrimeQ[0]; i++) {
    acCoproMessage[i+1] = ReadRandomByte();
```

[1]The attack can also just as validly target the most significant bits of k.
[2]It is also possible to run a similar attack if the target bits are set to one as opposed to zero.

```
}
IOpeak();
```

The I/O peak, `IOpeak` in the code fragment is a manually inserted command, which basically is a quick movement on the I/O channel from one to zero and back again. By placing a probe on the I/O (which is the C7 pin on the smart card as described in Chapter 2), this action can be detected by an oscilloscope. Since the command `IOpeak` indicates the beginning and end of the generation of k, the desired location in which to inject the fault can be easily identified.

To corrupt the fault in the desired way, a CLIO reader was used to create a glitch attack during the execution of the generation of the nonce. A CLIO is a specialised high precision reader that allows one glitch to be introduced following any arbitrarily chosen number of clock cycles after the command sent to the smart card. A photograph of a CLIO reader was shown in Chapter 2.

4.2.4 Generating a Faulty Nonce

The command that generated k was attacked in every position between the two `IOpeaks` in the code. It was found that the fault did not affect the assignment of k (the instruction `acCoproMessage[i+1] = ReadRandomByte();`) which always executed correctly. However, it was possible to cause the loop to end prematurely. It was found that tampering with the evaluation of `i`, caused a number of the least significant bytes of k not to be assigned, and so were effectively set to zero. An evaluation of a position that reset the last two bytes was performed. Out of 2000 attempts 857 were corrupted.

4.2.5 Glitching The Nonce During DSA Computations

The next step to the attack is to recreate the same fault attack during the generation of the nonce k, except while it is part of the computation of the DSA signature. The position found where a fault could be induced in the closed environment was equated to

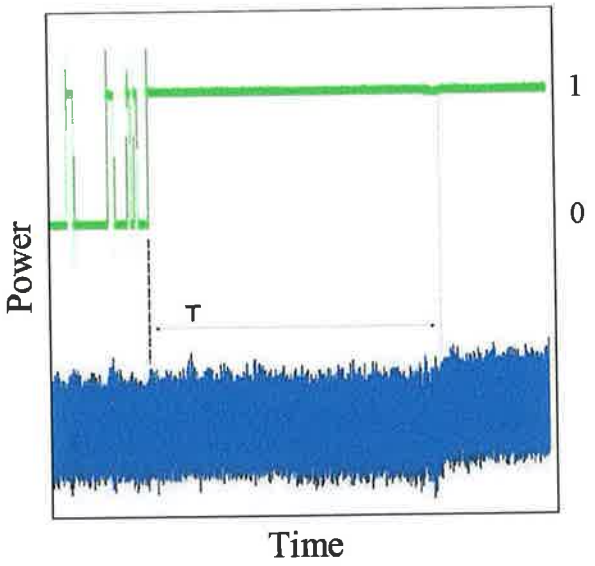

Figure 4.1: The top signal is the I/O. The bottom signal is the power consumption.

the generation of k in the natural working environment of the DSA. As changes in the value of k were not visible in the signature, this makes the attack more complex, since a faulty signature must be determined from a valid signature. To distinguish between such signatures, the I/O and the power traces can be used. An example of such monitoring is given in Figure 4.1. The objective of acquiring the power traces is to measure the time T elapsed between the end of the command sent to the card and the beginning of the calculation of r. This can be seen in the power consumption, as the chip will require more energy when the crypto-coprocessor is ignited.

Let t denote the time that it takes to reach the start of the calculation of r knowing that the picked k was smaller than q (i.e. that it was not necessary to restart the picking process) then, if $T = t$ the command has executed properly and k was picked correctly the first time. If $T > t$ then any fault targeting k would be a miss (as k was regenerated given that the value of k originally produced was greater than q). Signatures resulting from commands that feature such running times can be discarded as the value of k will not present any exploitable weaknesses. When $T < t$ the execution of the code generating

k has been cut short, inferring that some of the least significant bytes will be equal to zero. This allows signatures generated from corrupted k values to be identified.

Once the faulty signatures are generated, they are passed on to the next phase of the attack which involves the use of lattice reduction techniques.

4.2.6 Use of Lattice Reduction to Retrieve α

To extract the DSA signer's private key α, the lattice attack of Nguyễn and Shparlinski [94] is employed. This attack requires partial information on the nonce k and a number of signatures, where these particular nonces are used to generate the signatures. The number of signatures required in order for the attack to successfully retrieve the private key depends on the amount of information known. Since the glitch seeks to prevent the whole 160 bits of k being assigned, it is assumed that the known information used in the attack is the number of bits of k that are zero. The technique based on lattice reduction involves solving the closest vector problem. The lattice reduction aspect of the attack, which was executed by Nguyễn, one of the authors of the attack [94], is presented here for completeness.

For a rational number z and $m \geq 1$, $\lfloor z \rfloor_m$ denotes the unique integer a, such that $a \equiv z \pmod{m}$ (provided that the denominator of z is relatively prime to m) and $0 \leq a \leq m-1$. The symbol $|.|_q$ is defined as $|z|_q = \min_{b \in Z} |z - bq|$ for any real z.

Assume that the ℓ least significant bits of a nonce $k \in \{0, \ldots, q-1\}$ which will be used to generate a DSA signature, are known. Hence, an integer a exists such that $0 \leq a \leq 2^\ell - 1$ and $k - a = 2^\ell b$ for some integer $b \geq 0$. Given a message μ signed with the nonce k, the congruence

$$\alpha r \equiv sk - h \pmod{q},$$

can be rewritten for $s \neq 0$ as:

$$\alpha r 2^{-\ell} s^{-1} \equiv (a - s^{-1}h)2^{-\ell} + b \pmod{q}. \tag{4.1}$$

Now define the following two elements

$$\begin{aligned} t &= \left\lfloor 2^{-\ell} r s^{-1} \right\rfloor_q, \\ u &= \left\lfloor 2^{-\ell}(a - s^{-1}) \right\rfloor_q \end{aligned}$$

and remark that both t and u can easily be computed by the adversary from the publicly known information. Recalling that $0 \leq b \leq q/2^{\ell}$,

$$0 \leq \lfloor \alpha t - u \rfloor_q < q/2^{\ell}$$

can be obtained and therefore

$$|\alpha t - u - q/2^{\ell+1}|_q \leq q/2^{\ell+1}. \tag{4.2}$$

Thus, the adversary knows an integer t and a rational number $v = u + q/2^{\ell+1}$ such that

$$|\alpha t - v|_q \leq q/2^{\ell+1}.$$

In some sense, an approximation of αt modulo q is known. Now, suppose this can be repeated for many signatures, that is, d DSA signatures (r_i, s_i) of hashes h_i (where $1 \leq i \leq d$) such that the ℓ least significant bits of the corresponding nonce k_i are known, are generated. From the previous reasoning, the adversary can compute integers t_i and rational numbers v_i such that

$$|\alpha t_i - v_i|_q \leq q/2^{\ell+1}.$$

The goal of the adversary is to recover the DSA private key α. This problem is very similar to the so-called hidden number problem introduced by Boneh and Venkatesan in [23]. The problem is solved by transforming it into a lattice closest vector problem (for background on lattice theory and its applications to cryptography, see [93]). More precisely, consider the $(d+1)$-dimensional lattice L spanned by the rows of the following matrix

$$\begin{pmatrix} q & 0 & \cdots & 0 & 0 \\ 0 & q & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & q & 0 \\ t_1 & \dots & \dots & t_d & 1/2^{\ell+1} \end{pmatrix} \tag{4.3}$$

The inequality $|v_i - \alpha t_i|_q \leq q/2^{\ell+1}$ implies the existence of an integer c_i such that

$$|v_i - \alpha t_i - qc_i| \leq q/2^{\ell+1}. \tag{4.4}$$

Notice that the row vector $\vec{c} = (\alpha t_1 + qc_1, \dots, \alpha t_d + qc_d, \alpha/2^{\ell+1})$ belongs to L, since it can be obtained by multiplying the last row vector by α and then subtracting appropriate multiples of the first d row vectors. Since the last coordinate of this vector discloses the hidden number α, known as the hidden vector $\vec{c}$. The hidden vector is very close to the (publicly known) row vector $\vec{v} = (v_1, \dots, v_d, 0)$. By trying to find the closest vector to $\vec{v}$ in the lattice L, one can thus hope to find the hidden vector $\vec{c}$ and therefore the private key α.

In the attack described above, the lattice is constructed as previously mentioned, then the closest vector problem is solved with respect to $\vec{v}$, using the so-called embedding technique that heuristically reduces the lattice closest vector problem to the shortest vector problem (see [94] for more details).

4.2.7 Countermeasures

The heart of this attack lies with the ability to induce faults that reset some of k's bits. Hence, any strategy which permits such anomalies to be avoided or detected will help thwart the attacks described. Any of the countermeasures described in Chapter 2 can be used to achieve this. Checksums implemented in software or hardware can be applied to buffers of data. The execution can be randomised, thus making it difficult to predict what the smart card is doing at any given cycle, and so difficult to target the fault. Repeated refreshments will refresh k by generating several nonces and exclusive-oring them with each other, separating each nonce generation from the previous one by a random delay. This forces the adversary to inject multiple faults in randomly shifting time windows in order to reset specific bits of k.

It may also be possible to have real time testing of the random numbers being generated by the smart card, such as that proposed in the FIPS140-2 [99]. However, even if this is practical, it may be of limited use as the attack requires very few signatures to be successful. Consequently, the attack may well be complete before it gets detected.

4.2.8 Results

In the experiment described, NTL's [123] implementation of Schnorr–Euchner's BKZ algorithm [116] was used as the lattice basis reduction algorithm. To check results for any candidate y for the private key α, $\beta = g^y \pmod{p}$ was checked. To compute the success rates, the attack was executed 100 times with different parameters. Results can be seen in Table 4.1. For a successful attack, the speed at which the private key is retrieved will depend on the number of bytes reset in k. Naturally, there will be a tradeoff between the fault injection and the lattice reduction, meaning that when generating signatures with nonces with more reset bytes, the lattice phase of the attack will retrieve the key faster. Conversely if signatures generated with nonces having only one or two bytes reset, the lattice reduction phase will not run as quickly, but the fault injection part of the attack

will be much simpler.

Number of Bytes Set to Zero	Number of Signatures	Success Rate %
1	27	100
2	12	100
3	8	100
4	6	100
5	5	100
6	4	100
11	2	100

Table 4.1: Attack success rates.

4.3 Computational Improvements to DPA

Differential Power Analysis (DPA), and similar Side Channel Attacks (SCA), are in practice a lengthy process. The attack consists of a data acquisition phase followed by analysis of the data. In the context of DPA, the data analysis phase requires the data captured be added and subtracted in different combinations depending on the key hypotheses examined. This is a repetitive task with many redundancies. By introducing a number of basic precalculations this phase of DPA can be accelerated. As a benchmark, the calculations that are involved in classic DPA [66] will be briefly detailed.

4.3.1 DPA Calculations

The process of a DPA attack was described in Chapter 2. To briefly reiterate, a differential trace $\Delta_{k_{guess}}$ is calculated by finding the average of each set, S_0 and S_1, and then subtracting the resulting values from each other,

$$\Delta_{k_{guess}} = \frac{\sum_{t_i \in S_0} t_i}{|S_0|} - \frac{\sum_{t_i \in S_1} t_i}{|S_1|}$$

where all operations on power traces are computed in a point-wise fashion.

Let N equal the number of acquisitions collected. Calculation of a single differential

trace $\Delta_{k_{guess}}$ involves $N-2$ additions (as there will be an average of $\frac{N}{2}$ elements in each set), two divisions and a subtraction. Therefore, for all hypotheses, the total number of operations to calculate all differential traces is $2^n \times (N+1)$, where n is equal to the bit length of k_{guess}.

The operations involved in the calculation of differential traces are addition, subtraction and division of power traces. Of these, the most utilised operation is addition of power traces. Therefore, any efforts to minimise the number of additions will see the optimisation of the data analysis phase. The division (calculation of mean) and subtraction operations will be treated as constant, as these are fundamental operations in the calculation of the differential traces and so cannot be enhanced.

4.3.2 The Global Sum

The optimisation of DPA in terms of the calculations that it performs in the data analysis phase of the attack, involves changing how the differential traces are calculated. The simplest optimisation involves the calculation of the global sum G. The global sum is the summation of all the power traces that have been acquired,

$$G = \sum_{i=1}^{N} t_i.$$

The calculation of the differential traces now only involves the summation of a single set, i.e.

$$\Delta_{k_{guess}} = \frac{G - \sum_{t_i \in S_{\text{least}}} t_i}{N - |S_{\text{least}}|} - \frac{\sum_{t_i \in S_{\text{least}}} t_i}{|S_{\text{least}}|}.$$

Here S_{least} represents the set with the least number of elements,

$$S_{\text{least}} = \begin{cases} S_0 & \text{when } |S_0| \leq |S_1| \\ S_1 & \text{when } |S_0| > |S_1| \end{cases}$$

The expected number of additions required to generate S_{least} is

$$\begin{aligned}\mathrm{E}(X) &= \sum_{i=0}^{N} i \Pr[X = i] \\ &= \sum_{i=0}^{\frac{N}{2}} i \binom{N}{i} \left(\frac{1}{2}\right)^{N} + \sum_{i=\frac{N}{2}+1}^{N} (N-i) \binom{N}{i} \left(\frac{1}{2}\right)^{N-i}\end{aligned}$$

If, for example, 1000 acquisitions were taken, this would result in an expected number of additions per hypothesis of 487. This is an improvement over the case where a set is chosen arbitrarily, when the expected number of additions would be 499 (i.e. $\frac{N}{2} - 1$).

The cost of precalculating G for a single hypothesis is obviously not worthwhile. Note, however, that separate pre-computation of G is not mandatory. The trick here consists in computing a first hypothesis, just as in the original version of DPA, and then summing the two resulting average traces to get G, thereby allowing the complexity of the next $2^n - 1$ hypotheses calculations to be reduced at no extra cost.

4.3.3 Formation of Power Trace Equivalence Classes

Recall that inputs to the selection function $\beta_i = S(\alpha_i, k_{guess})$ consists of information known to the adversary α_i, which may be related to the plaintext, and partial bits of the key k_{guess}, to produce unknown intermediate output β_i. Generally $S()$ will accept n bits of both α_i and k_{guess}. For example, Figure 4.2 depicts a situation where different portions of the known data are entered into the selection function with different portions of the key. This would be typical of an n-bit processor working on words of data at a time. Similarly, in the case of symmetric ciphers such as AES and DES where the substitution box is chosen as the selection function, the S-Box will accept specific portions of the plaintext and the key.

Such portions of α and k, will be denoted by $\alpha_{i,j}$ and $k_{guess,j}$ respectively and $S_j()$ will be used to distinguish the selection function for these portions. Considering $\alpha_{i,j}$

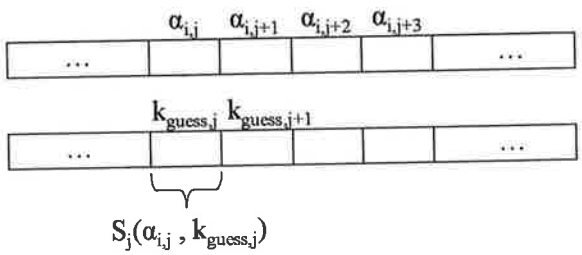

Figure 4.2: Selection function $S_j()$.

alone, there are only 2^n possible values which can enter a given selection function $S_j()$. Therefore, a number of the power traces will have the same value for $\alpha_{i,j}$, and thus can be treated in the same manner (i.e. they can be divided into a set of equivalence classes). A representative power trace T_l can be used to encapsulate all power traces t_i, where the inputs $\alpha_{i,j}$ to the selection function S_j are equivalent. T_l is defined for $S_j()$ as

$$T_l = \sum_{i=1}^{N} \chi_i t_i \quad \text{where} \quad \chi_i = \begin{cases} 1 & \text{if } \alpha_{i,j} = l \\ 0 & \text{otherwise} \end{cases}$$

where $0 \leq l \leq 2^n$. This will produce 2^n representative power traces from the N acquisitions, hence immediately reducing the number of power traces that must be worked with. Creating each representative power trace will require approximately $\frac{N}{2^n} - 1$ additions, that is assuming that there is $\frac{N}{2^n}$ power traces in each set. This must be repeated for all 2^n representative power traces. The differential trace can now be calculated as,

$$\Delta_{k_{guess}} = \frac{\sum_{T_l \in S_0} T_l}{|S_0|} - \frac{\sum_{T_l \in S_1} T_l}{|S_1|}.$$

The representative power traces T_l can be used in conjunction with the global sum optimisation. Precalculation for this optimisation involves the formation of the representative power traces T_l and the global sum G, where G can be constructed as a function of

T_l,

$$G = \sum_{n=1}^{2^n} T_l$$

requiring a further $2^n - 1$ additions. The differential trace can now be calculated as,

$$\Delta_{k_{guess}} = \frac{G - \sum_{T_l \in S_{\text{least}}} T_l}{N - |S_{\text{least}}|} - \frac{\sum_{T_l \in S_{\text{least}}} T_l}{|S_{\text{least}}|}.$$

To generate a single differential trace will require $2^{n-1} - 1$ additions. For all hypotheses, a total of $2^n \left(2^{n-1} - 1\right)$ additions will be required to calculate all the differential traces for one selection function.

Therefore, the total number of operations that will be incurred in generating all differential traces is

$$\begin{aligned} \text{Total Calculations} &= 2^n \left(\frac{N}{2^n} - 1\right) + (2^n - 1) + 2^n \left(2^{n-1} - 1\right) \\ &= N - 1 + 2^n \left(2^{n-1} - 1\right) \end{aligned}$$

where for the representative power traces T_1 to T_{2^n}, approximately $\frac{N}{2^n} - 1$ additions will be required to make up each T_l. An additional $2^n - 1$ summations will be required to form the global sum G, and $2^{n-1} - 1$ additions will be required to generate each differential trace (since only one set, S_{least}, needs summing), for the 2^n key hypotheses.

4.3.4 Combining Representative Power Traces

Further efforts can be made to minimise the number of additions for the set S_{least}. This involves precalculating certain representative power trace combinations so that computing a differential trace becomes more of a look up operation. In particular, it is the objective to precalculate certain representative power trace combinations so that groups of power traces that occur in the same set for one hypothesis, be recycled in subsequent hypothesis testing. Since there exist 2^n representative power traces, pre-computing all possible $2^n!$

combinations is infeasible. Hence, it is proposed that the representative power traces are partitioned into groups of size x, and the different possible combinations for each group are pre-computed.

For example, for $x = 2$, adjacent power traces are summed. Each pair is defined as the value $T_{2l,2l+1}$, where l takes the integer values in the interval $[0, 2^{n-1}]$ and

$$T_{2l,2l+1} = T_{2l} + T_{2l+1}.$$

The use of the combined power traces $T_{2l,2l+1}$ in the evaluation of the set S_{least}, results in three possible scenarios for each pair:

1. The pair occurs, i.e. $T_{2l} + T_{2l+1}$ appears in S_{least}. The probability of this occurring is $\frac{1}{4}$. In this case, one additional summation must be performed.

2. The pair does not appear in S_{least}, i.e. the pair is in G. The probability of this happening is also $\frac{1}{4}$. In this case, no action is performed.

3. The two traces T_{2l} and T_{2l+1} appear in two separate sets. There is a higher probability of this happening, i.e. $\frac{1}{2}$. In this case, a single addition is required.

The expected number of additions per $\Delta_{k_{guess}}$ (assuming the global set already exists), is therefore $\frac{3}{4} \times 2^{n-1}$. This is because there are 2^{n-1} groups of three elements (T_{2l},T_{2l+1} and $T_{2l,2l+1}$), one of which will be used to add to S_{least} with probability $\frac{3}{4}$.

The differential trace can now be calculated as,

$$\Delta_{k_{guess}} = \frac{G - S_{\text{least}}}{N - |S_{\text{least}}|} - \frac{S_{\text{least}}}{|S_{\text{least}}|}$$

where in calculating the set S_{least}, a check will be performed to determine whether the combinations of power traces T_l that arise in S_{least} are already added.

The precalculation of groups of representative power traces can be performed for any value x. However, it is preferable to choose x such that it evenly divides N, otherwise

the groups will not be evenly distributed. Once the combined power traces have been generated, the number of additions that are required for the set S_{least} can be described by

$$\left(\frac{2^x - 1}{2^x}\right)\left\lfloor\frac{2^n}{x}\right\rfloor$$

where the number of groups is determined by $\frac{2^n}{x}$ and the probability of elements in that group turning up in S_{least} is $\frac{2^x-1}{2^x}$. Note that this is only an approximation for the number of additions required for each set. In practice the number of additions involved in generating $\Delta_{k_{guess}}$ will depend on the contents of the set, and whether the pre-computed combinations appear or not.

Since the combined power traces must be pre-computed before the differential traces can be generated. This involves generating the $2^x - 1$ combinations necessary for each group of x power traces. This will involve $2^x - x - 1$ additions, as x power traces already exist. The amount of precalculation involved (incorporating G, T_l and the pre-computed combinations) is therefore,

$$N - 1 + (2^x - x - 1)\left\lfloor\frac{2^n}{x}\right\rfloor$$

where generating T_l and G will require $2^n\left(\frac{N}{2^n} - 1\right) + (2^n - 1)$ and so $N - 1$ additions, following the reasoning given in Section 4.3.3. The total number of operations per differential trace will include the additions both as part of the precalculation and as required for the hypothesis. Therefore, the number of calculations for all hypotheses is

$$\text{Total Calculations} = N - 1 + (2^x - x - 1)\left\lfloor\frac{2^n}{x}\right\rfloor + 2^n\left(\left(\frac{2^x - 1}{2^x}\right)\left\lfloor\frac{2^n - 1}{x}\right\rfloor\right).$$

The memory requirement is a vital factor, as the more pre-computed values that are required, the more memory they will take up, and the more time it will take to load these values into memory. In order to balance the time-memory trade off and achieve the

optimal attack, the amount of memory that is required needs to be projected. The number of power traces that need to be stored in memory include the original N power traces t_i, the global sum G, the representative power traces T_l and the combined power traces, which is determined by $2^x - x - 1 \left\lfloor \frac{2^n}{x} \right\rfloor$ for groups of size x.

4.3.5 Chosen Plaintext Differential Power Analysis

Unfortunately, the precalculations previously made for the selection function $S_j()$ will be redundant for the subsequent selection function $S_{j+1}()$, as the power traces, when classified according to the partial input $\alpha_{i,j}$, will fall into different equivalence classes from those that apply for $\alpha_{i,j+1}$. Therefore, for the selection function $S_{j+1}()$, regeneration of the representative power traces will be required.

In classical DPA, the message given to the algorithm under attack is random. However, if plaintexts can be chosen, the precalculated T_l can be used to attack subsequent selection functions. The simplest case is where the input to the selection function $S_1()$ can be made the same for $S_2(), S_3, \ldots, S_{N_S}$ (where N_S is the number of selection functions used in one round of the algorithm under consideration). For example, suppose that the plaintext is constructed such that all plaintext bytes are equal, i.e. byte[1] = byte[2] = $\ldots$ = byte[16] in AES [34]. This means that there are 256 possible values for the plaintext. Calculating the differential traces for the first S-box (which is the selection function in this situation) will calculate the differential trace for all others at the same time, giving sixteen peaks at the points in time at which the sixteen key bytes are being manipulated. However, using this method may not always be advantageous, as some confusion can arise as to which peak corresponds to which key byte.

A similar approach can be applied to an implementation of DES, where the plaintext can be generated such that the value $\alpha_{i,j}$ entered into the even-numbered S-boxes ($S_{j \bmod 2=0}()$) are all equal, and the value of $\alpha_{i,j}$ entered into the odd numbered S-boxes ($S_{j \bmod 2=1}()$) are all equal. All the even numbered S-boxes can use the same set of

data generated during the pre-calculation for the first S-box. The odd numbered S-boxes also use this data but with a permutation on the value associated with each representative power trace. This does not affect the quality of the results produced, as each S-box uses a different permutation. The differential trace will be at the same level as if a random plaintext was used.

Note that these optimisations are applicable when the attack is concentrating on the first round of a block cipher. If the attack focuses on the last round, where the differential traces are related to the ciphertext, then these optimisations will be useless as the data cannot be controlled.

4.3.6 Results

For demonstrative purposes, the results are presented where DPA is performed on both the DES and AES. The selection function for both of these cryptographic algorithms is the substitution box (S-box) or `SubBytes` function. The S-box for DES accepts 6 bits at a time and so $n = 6$. The S-box for AES accepts 8 bits at a time and so $n = 8$. It is assumed that the number of times that DES/AES is executed is 1000. Therefore, $N = 1000$, which consequently relates to the number of power traces t_i acquired. Table 4.2 details the number of calculations involved in generating the differential traces for DES. Table 4.3 details the number of calculations involved in generating the differential traces for AES.

The formulae that was used to derive these values were given in the previous section. The value for total calculations is derived from the number of precalculations required plus the number of additions for all hypotheses. As shown in Table 4.2 and Table 4.3, even the most straightforward optimisation, which involves forming a global sum value G, results in almost halving the number of calculations required to generate all differential traces. The optimisation of forming representative power traces reduces the number of calculations required to generate all differential traces even further. The amount of

	Precalc.	Additions per hypothesis	Additions for all hypotheses	Total Calculations	Number of Power Traces
Classic DPA	-	998	63872	63872	1000
Optimisation 1: Global Sum	999	487	31168	32167	1001
Optimisation 2: Equivalence Classes w/out Global Sum	936	62	3968	4904	1064
Equivalence Classes with Global Sum	999	31	1984	2983	1065
Optimisation 3: Trace Combining					
2 bits	1031	24	1536	2567	1097
4 bits	1175	15	960	2135	1241
8 bits	2975	≈ 8	512	3487	3041

Table 4.2: Optimisation results for DES.

computation required for DES, with and without global sum, reduces to 5% and 8% respectively of the classic DPA calculations. The amount of computation required for AES, with and without global sum, reduces to 13% and 26% respectively of the classic DPA calculations.

For the combined power trace optimisation, the number of calculations required to generate all differential traces continues to decrease, but this may be at the cost of increased storage requirements. The best results for a realistic amount of memory are obtained when 4 bits for both DES and AES are grouped together at a time. This reduces the workload to 3% and 7% of the original DPA for DES and AES respectively. However, the amount of memory required for this has increased. In addition, the fact that representative and combined power traces T_l will require more space than the original power traces t_i, must be accounted for. Note that the size of the power traces will vary depending on a number of factors, such as the storage capacity of the oscilloscope, the amount of time spent localising the selection function using SPA, and the algorithm being attacked. The worst possible scenario when using the combined power trace optimisation is where the

	Precalc.	Additions per hypothesis	Additions for all hypotheses	Total Calculations	Number of Power Traces
Classic DPA	-	998	255488	255488	1000
Optimisation 1: Global Sum	999	487	124672	125671	1001
Optimisation 2: Equivalence Classes w/out Global Sum	744	254	65024	65768	1256
Equivalence Classes with Global Sum	999	127	32512	33511	1257
Optimisation 3: Trace Combining					
2 bits	1127	96	24576	25703	1385
4 bits	1703	60	15360	17063	1961
8 bits	8903	≈ 32	8192	17095	9131

Table 4.3: Optimisation results for AES.

memory requirement becomes unmanageable and pre-computation actually inhibits the attack. There are two approaches that an adversary can employ to combat this situation. In the case where the acquisitions captured are large, the acquisitions can be split into sections, the DPA calculation performed on each section and the results concatenated to construct the full differential trace. Alternatively, the memory usage required can be determined so that the adversary can decide how much pre-computation is possible with the memory available, thus allowing maximum benefit from the optimisations described.

Note that the ideas expressed in Section 4.3.5 have not been discussed in the example, as the gain for the overall attack depends on how the message can be manipulated.

4.4 Conclusion

In this chapter, two aspects of implementation attacks were examined. Firstly, a fault attack on the DSA, which was already proven in theory, was validated in practice. The attack consisted of two stages. The first stage dealt with fault injection. The second involved forming a lattice for the data gathered in the previous stage and solving the

closest vector problem to reveal the secret key. The experimental results proved that launching an attack in practice was not only possible, but could find the key with just 27 faulty signatures when one byte of k is zeroed. Secondly, a number of computational improvements to the data analysis phase of DPA were proposed. In the best case scenario these improvements resulted in a 97% and 93% reduction in data processing for DES and AES respectively.

This research, formed the basis for the subsequent work which involved assessing bilinear pairings for vulnerability to both power and fault analysis. The next two chapters present the research performed in this area.

Chapter 5

Power Analysis of Bilinear Pairings

5.1 Introduction

In this chapter, first-order power analysis of three bilinear pairing algorithms is discussed. Specifically, the BKLS algorithm for the Tate pairing, the Ate pairing and the BOGhES algorithm for η_T pairing are assessed. In 2006, Scott et al. [120] demonstrated that each of these bilinear pairing algorithms was efficiently computable and comparable with alternative public key algorithms on a resource constrained smart card. As demonstrated in Chapter 2, side channel attacks such as power analysis are very powerful attacks, particularly in the context of smart cards. Therefore, analysis of bilinear pairings eligible for implementation on smart cards is necessary. To date, one publication on the topic of side channel analysis of pairings exists [107]. However, this is a large area of research with numerous attacks and various pairing algorithms and implementations to consider.

Firstly, the contribution of [107], which describes a DPA style attack of the Duursma-Lee algorithm for the Tate pairing, is briefly reviewed in Section 5.2. In Section 5.3, a number of observations about bilinear pairings are made. These observations are of importance in the context of power analysis.

In Section 5.4, finite field operations are examined for vulnerability to first-order

power analysis. Bilinear pairings ultimately rely on the underlying finite field arithmetic. Hence, the vulnerability of the finite field operations will have consequences for bilinear pairings. First-order power analysis of the relevant finite field operations is described. A novel technique to attack finite field operations from a structural perspective is proposed. This technique is algorithm independent and so has implications for a range of cryptographic algorithms.

The vulnerability of the underlying finite field operations are then described in the context of the three candidate pairing algorithms in Section 5.5. Potential first-order power analysis attacks on the Tate pairing, the Ate pairing and the η_T pairing are given. The susceptibility to power analysis of each bilinear pairing is subsequently summarised, highlighting some of the key findings of the research.

Countermeasures to deter first-order power analysis attacks are proposed in Section 5.6. Finally this chapter is concluded in Section 5.7. Appendix A of this dissertation contains numerical examples of the attacks described in this chapter.

The chapter contains joint work with Mike Scott, which was published at the International Conference on Cryptology - Vietcrypt 2006 [138]. This conference was held in Hanoi, Vietnam in September 2006. Sections 5.3, 5.4 and 5.5 are part of the contribution of this thesis.

5.2 Related Work

As mentioned previously, the first work[1] in the area of side channel attacks on pairings was performed by Page and Vercauteren [106] in 2004. In this paper, both passive and active implementation attacks (namely power and fault analysis) on the Duursma-Lee algorithm (given in Algorithm 12) for computing the Tate pairing on supersingular elliptic curves in characteristic three was considered, which could be extended to the general case.

[1] Since the publication of the works [106] and [138], there has been another publication on this topic by [62]. This deals with the case where classic DPA is applied to the multiplication operation over the binary field.

In this section, a brief overview of the passive side channel attack aspect of the paper is presented. The active attack contribution of the paper will be examined in the next chapter when fault analysis of pairings are described.

Algorithm 12 Duursma-Lee Algorithm for $E(\mathbb{F}_q) : y^2 = x^3 - x + b$ where $b = \pm 1$ and $q = 3^m$ [106].

INPUT: $P = (x_1, y_1), Q = (x_2, y_2) \in E(\mathbb{F}_{q^6})$
OUTPUT: $m \in \mathbb{F}_{q^6}^* \mid m^r = 1$
1: $m \leftarrow 1$
2: **for** $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ **to** 0 **do**
3: $\quad x_1 \leftarrow x_1^3$
4: $\quad y_1 \leftarrow y_1^3$
5: $\quad \mu \leftarrow x_1 + x_2 + b$
6: $\quad \lambda \leftarrow -y_1 \cdot y_2 \sigma - \mu^2$
7: $\quad g \leftarrow \lambda - \mu\rho - \rho^2$
8: $\quad m \leftarrow m \cdot g$
9: $\quad x_2 \leftarrow x_2^{1/3}$
10: $\quad y_2 \leftarrow y_2^{1/3}$
11: **end for**
12: **return** m^{q^3-1}

First-order power analysis is based on the property that the secret is highly correlated with the power trace. First-order attacks exploit operations where known data α comes in direct contact with secret data k, that is, where a selection function $\beta = S(\alpha, k)$ exists. In the Duursma-Lee attack, the authors of [106] identified line 6 of the algorithm as the operation to attack. Since y_1 is related to the public point P, it will be known to the adversary. The coordinate y_2 on the other hand relates to the secret point Q, so is of interest to the adversary. Hence, the operation $y_1 \cdot y_2$ can be used as an eligible selection function. Given that y_1 is known, the hypothetical product of the output of $y_1 \cdot y_2$ is calculated, given guesses for y_2.

Two types of power analysis attacks are detailed to determine the correctness of the guesses for y_2. Firstly an SPA attack is described. The success of this attack however, is reliant on the multiplication algorithm being the Shift and XOR method. Secondly, a DPA attack on the multiplication operation, where the multiplication algorithm is irrelevant, is proposed. This attack uses a classic DPA style approach to guess one bit of the secret coordinate y_2 at a time.

There are two main shortcomings of Page and Vercauteren's work. Firstly, power analysis of pairings was proposed in a generic sense. When they presented their results, an efficient implementation of a pairing on a smart card had not yet been witnessed. In 2006, Scott et al. [120] presented the first timings for the computation of three highly optimised pairing algorithms, which were shown to be computed as efficiently as alternative contemporary cryptosystems on a 32 bit smart card. Since the types of attacks being investigated are implementation specific, specific implementations of pairings must be assessed. Secondly, Page and Vercauteren's technique extracted one bit of the coordinate y_2 at a time using DPA. In the previous chapter, the work involved in performing a DPA attack was examined. The intensive process of data manipulation was demonstrated. For characteristic three implementations, with the embedding degree $k = 6$, the recommended length of elements in the underlying field is 160 bits. This approximately achieves the recommended 1024 (or 960) bit level for index calculus security. Given that the DPA attack of [106] reveals one bit of the secret coordinate y_2 at a time, extraction of the secret is inefficient. In addition, [24] demonstrated that the assumptions that DPA make are incomplete, and so prone to inaccuracies. These facts raise the question whether a more computationally realistic and reliable attack can be performed.

5.3 Bilinear Pairing Observations

In this section, some useful observations about bilinear pairings are made, which aid first-order power analysis. The secret parameter can be entered as the first or second parameter in the bilinear pairing. For instance, in Boneh and Franklin's IBE, the secret parameter is used in the decryption operation, i.e. the plaintext is uncovered as $m = C_2 \oplus H_2(\omega(D_B, C_1))$, where the private key D_B is entered as the first parameter in the pairing. Whereas, in Gentry and Silverberg's Hierarchial IBE (HIDE) [36], the secret parameter is also used in the decryption operation, but as the second parameter

to the pairing. The operations and transformation that the elliptic curve point undergoes depends on the parameter position the point is entered in. Therefore, depending on which parameter the secret is entered as, will affect the choices for attack. This is particularly relevant for certain pairing algorithms, such at the Tate and Ate pairing, which as will be demonstrated, consists of two very different paths for the first and second parameter, and so presents two varying choices for attack[2]. In certain circumstances however, such choices may not be available. For instance, if the elliptic curve is supersingular and a distortion map ϕ is used, as is the case with the η_T and modified Tate pairing $\hat{e}$, the parameters to the pairing can be arbitrarily switched. For example,

$$\hat{e}(P, Q) = \hat{e}(Q, P)$$

will yield the same result. In such situations, regardless of which parameter the secret is entered in, an equal number of choices for attack will exist.

Each bilinear pairing algorithm must therefore be assessed given that the secret parameter can be entered as either parameter. To distinguish between the different parameters, and so the various operations relating to that parameter choice, the term path will be used. Specifically when the secret is entered as the P parameter, it will be referred to as taking the P path and when the secret is entered as the Q parameter, it will be referred to as taking the Q path.

On an elliptic curve, when represented using affine coordinates, the secret point will consist of the coordinates (x, y). Due to point compression however, only one of these coordinates needs to be extracted, since given x, y can be found by solving the elliptic curve equation E. Therefore, in each of the pairing algorithms, the coordinate x is the data value of interest. In particular, for first-order power analysis, operations in which the secret coordinate x comes in contact with known data are of particular importance.

[2]Note that for the Tate and Ate pairing, while the secret parameter can take either path, it must hold its position for the entire protocol, so once the secret point has been allocated to a position in the pairing, it is fixed there.

Operations involving elements from the base field $\mathbb{F}_q$, where $q = p$ or 2^m, as opposed to the extension field $\mathbb{F}_{q^k}$, are also preferable since extension field elements are k-bit times larger than that of base field elements. Finite field operations central to the computation of the pairing will be discussed next.

5.4 Dissecting Bilinear Pairings

Each type of bilinear pairing is different and so is implemented differently. Therefore, each pairing algorithm must be assessed individually for its susceptibility to power analysis. There are common features between bilinear pairings and these are considered next.

All bilinear pairings are constructed from the same building blocks of finite field arithmetic. A number of finite field operations are required for bilinear pairing computation. The order and manner in which they are used is dependent on the type of pairing. Finite field addition, subtraction, multiplication, division, inversion, and exponentiation over both the base and extension field are common to the candidate bilinear pairings. For the Tate and the Ate pairing, arithmetic is over a prime field, whereas arithmetic is over a binary field for the η_T pairing.

Many algorithms exist to compute each finite field operation. For example, the operand scanning method, the product scanning method, the Karatsuba-Ofman method and/or the Comba method can be used to multiply two elements in the prime field, $a \cdot b$ where $a, b \in \mathbb{F}_p$. Similarly, the shift and XOR method, the right to left or left to right comb method for polynomial multiplication can be used with or without windows to multiply two elements in the binary field, $a \cdot b$ where $a, b \in \mathbb{F}_{2^m}$. A description of these methods can be found in [48].

In this section, a method is proposed by which the first-order power analysis attack, CPA, can be launched on a finite field operation regardless of the underlying algorithm used. This idea is based on analysing the structure of finite field operations and will be

discussed next.

5.4.1 Structural Analysis of Core Pairing Operations

Consider a function of the form $\beta = S(\alpha, k)$. If partial output of the function $S()$ can be deterministically calculated given portions of α and k, where α relates to known data and k relates to the secret, then this function constitutes an eligible selection function. A first-order power analysis attack, such as DPA or CPA, can be performed on selection functions of this form. In the context of this research, $S()$ represents a finite field operation. For example, $S()$ could represent finite field multiplication $\alpha \cdot k$. To perform a first-order power analysis attack, an adversary must be able to determine a link between the input to $S()$ and the corresponding output produced, β. By analysing how an operation progresses, that is, by looking at its structure, a connection between the input and output can be made. In practical implementations, finite field elements are stored and operated on as w-bit blocks, where w is typically 8, 16 or 32 bits, the word length of the processor. This allows the order and manner in which words of data are operated on, to be easily deduced. The interpretation of this order forms the basis for structural analysis. The examination of words of data is also appropriate for the methodology of CPA.

In this section, structural analysis of the finite field operations multiplication, square root, and reduction will be performed. These operations are chosen for the following reasons. First, addition and multiplication are the operations fundamental to a finite field. Subtraction of field elements can be defined in terms of addition. Similarly, squaring and division of field elements can be defined in terms of multiplication. Since the structure of finite field addition is straightforward, the structure of finite field multiplication will only be considered. In addition, in all of the candidate pairing algorithms, finite field multiplication operates on known and secret data at some point, allowing it to be selected as an eligible selection function. Secondly, reduction is central to finite fields and plays a role in almost all finite field operations. Finally, the square root operation, while it can be

defined in terms of addition, multiplication and reduction, is unique to the η_T pairing and so presents as an extra avenue of attack.

5.4.2 Multiplication

The basic method for multiprecision multiplication is the pencil-and-paper method for $\mathbb{F}_p$ and the shift and XOR method for $\mathbb{F}_{2^m}$ [115]. Field multiplication of $a, b \in \mathbb{F}_p$ can be accomplished by first multiplying a and b as integers, then reducing the result modulo p. Field multiplication of $a(z), b(z) \in \mathbb{F}_{2^m}$ can be accomplished by first multiplying $a(z)$ and $b(z)$ as polynomials, which in the case of binary fields infers carry-free binary polynomial multiplication, then reducing the result modulo the irreducible polynomial $f(z)$. Figure 5.1 illustrates the pencil-and-paper multiplication method for $\mathbb{F}_p$.

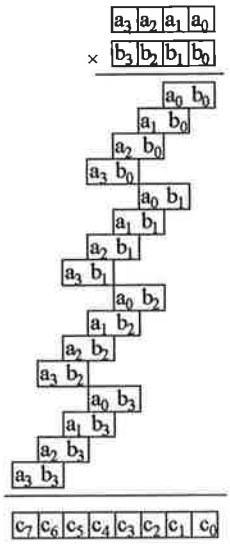

Figure 5.1: Multiplication of $\mathbb{F}_p$ elements: the pencil-and-paper method.

The specific input words that are responsible for the specific output words can be distinguished using the structural representation of the multiplication. For example, the least significant word of the product c, is determined by the least significant words of a and b in both fields, $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$. This is because $a_0 \cdot b_0$, where a_0 and b_0 are w-bit

words, produces a $2w$-bit product. Of this product the lower word c_0 contributes to the least significant word of the final product c and the upper word contributes to the word c_1 (along with other intermediate products).

In the situation where input to the multiplication is comprised of known and unknown data (for example, a may relate to α and b may relate to k), a first-order power analysis attack can be applied since predictions about the partial output of the multiplication can be made given partial input. So here, the multiplication operation is the selection function. The process of generating the key hypotheses when the multiplication operation is the selection function will be described next. This is described for the case where multiplication is performed over $\mathbb{F}_{2^m}$.

5.4.2.1 Hypotheses Generation

Let $\alpha_i \in \mathbb{F}_{2^m}$ be data known to the adversary, where $1 \leq i \leq N$. Let $k \in \mathbb{F}_{2^m}$ be a secret value unknown and of interest to the adversary. At some point in the target algorithm and on the target device, the operation $\alpha_i \cdot k \pmod{f(z)}$ will be calculated. It is assumed that numerous power traces t_i relating to the execution of $\alpha_i \cdot k \pmod{f(z)}$ have been acquired. Therefore, N power traces corresponding to the known data α_i should be in the possession of the adversary. Let β_i denote the $2m$ bit product resulting from the multiplication of $\alpha_i \cdot k$ before reduction is invoked.

In memory the elements α_i and k will be stored in arrays $\alpha_i = (\alpha_i[r-1], \alpha_i[r-2], \ldots, \alpha_i[0])$ and $k = (k[r-1], k[r-2] \ldots k[0])$ of w-bit words. Similarly, the product can be visualised as the array $\beta_i = (\beta_i[s-1], \beta_i[s-2], \ldots, \beta_i[0])$. Note that $r = \lceil \frac{m}{w} \rceil$ is the number of words required to store α_i and k, and $s = \lceil \frac{2m}{w} \rceil$ is the number of words required to store β_i.

To uncover the secret k, CPA can be used to extract words of k at a time. In the field $\mathbb{F}_{2^m}$ only one word of k, affects the most and least significant word of β_i since there is no

bit propagation, i.e.

$$\begin{aligned} \beta_i[0] &= \alpha_i[0] \cdot k[0] \\ \beta_i[s-1] &= \alpha_i[r-1] \cdot k[r-1]. \end{aligned}$$

All middle words of the product $\beta_i = \alpha_i \cdot k$ are affected by more than one word of k. Therefore, there are two possible positions from which the attack can commence.

To extract the least significant word of k, $k[0]$, CPA will correlate the hypothetical output of $\beta_i[0] = \alpha_i[0] \cdot k[0]$ for each prediction of $k[0]$, with the actual power consumption values. Algorithm 13 describes the steps to generate the data bank of hypothetical output that is used to correlate to the actual power consumption.

Algorithm 13 Generate hypothetical output of the multiplication $\beta_i[0] = \alpha_i[0] \cdot k[0]$.

INPUT: The set $\{\alpha_0, \alpha_1, \ldots, \alpha_N\}$ where the set is known/computable by the adversary
OUTPUT: An $N \times 2^w$ matrix H_0 containing the output of $\alpha_i[0] \cdot k[0]$

```
1: for 1 ≤ i ≤ N do
2:     X = α_i[0]
3:     for 0 ≤ j < 2^w do
4:         k[0] = j
5:         β_i[0] = X · k[0]
6:         H_0(i, j) = β_i[0]
7:     end for
8: end for
9: return H_0
```

This algorithm produces a $N \times 2^w$ matrix denoted by H_0. Note that the size of H_0 depends on N and the word length w. When the word length is large, predictions for sections of w can instead be made. Section 5.4.6 further discusses this issue. The matrix H_0 contains the hypothetical product of $\beta_i[0]$ given all possibilities for N known values of $\alpha_i[0]$ when multiplied by the 2^w possible values for $k[0]$. The data contained in H_0 will be translated into its estimated power consumption values, and used in the correlation test for determining $k[0]$. The process of establishing which guess for $k[0]$ is correct is described in Section 5.4.5.

The multiplication of $k[0] \cdot \alpha_i[0]$ will in fact produce a $2w$-bit product $\beta_i[0]$. Only

the least significant word of this product is required as entry in H_0. The most significant word of $\beta_i[0]$ contributes to the subsequent product word $\beta_i[1]$.

To extract the remaining interior words of k, the attack proceeds similar to Algorithm 13. Since all middle words in the $2m$-bit product β_i are influenced by more than one word in k, $k[1]$ cannot be found unless $k[0]$ is known, $k[2]$ cannot be found unless $k[1]$ and $k[0]$ is known, and so on. Therefore, line 5 in Algorithm 13 must be replaced by

$$\beta_i[v] = (\alpha_i[0] \cdot k[u]) + (\text{ auxiliary words })$$

where $1 \leq u \leq r-2$ and $1 \leq v \leq s-2$. For instance,

$$\beta_i[1] = (\alpha_i[0] \cdot k[1]) + (\alpha_i[0] \cdot k[0]) + (\alpha_i[1] \cdot k[0])$$

and

$$\beta_i[2] = (\alpha_i[0] \cdot k[1]) + (\alpha_i[0] \cdot k[2]) + (\alpha_i[1] \cdot k[1]) + (\alpha_i[2] \cdot k[0]).$$

An adversary could alternatively begin extracting the most significant word of k, $k[r-1]$. The attack on the most significant word is similar to the attack on the least significant word except the word positions that are focused on are $\alpha_i[r-1]$, $k[r-1]$ and $\beta_i[s-1]$. In the case of multiplication over the binary field, predictions for the least and most significant word of k can be generated simultaneously. This is due to the fact that the product words $\beta_i[0]$ and $\beta_i[s-1]$ depend only on the secret words $k[0]$ and $k[r-1]$ respectively. Once $k[0]$ and $k[r-1]$ are identified, the remainder of k can be found by iteratively stepping inwards, in that the next step would be to find $k[1]$ and $k[r-2]$.

When multiplication is performed over the prime field, i.e. $\alpha, k \in \mathbb{F}_p$, a similar attack can be launched. In this situation however, the attack is restricted to commencing from the least significant word. This is due to the propagation from carry bits which will significantly affect all other words.

5.4.3 Square Root

The square root method is only required for the η_T pairing. Therefore square root calculation over the binary field will be discussed. The method for calculating $\sqrt{a}$, $a \in \mathbb{F}_{2^m}$, is based on the Fermat's little theorem: $a^{2^m} \equiv a$. From this it can be seen that $\sqrt{a}$ can be calculated as $a^{2^{m-1}}$, which requires $m-1$ squarings. A more efficient method, proposed by Fong et al. [38], can be obtained from the observation that $\sqrt{a}$ can be expressed in terms of the square root of the element z. Let $a = \sum_{i=0}^{m-1} a_i z^i$, where $a_i \in \{0,1\}$. The square root of a can be written as

$$\begin{aligned}
\sqrt{a} &= (\sum_{i=0}^{m-1} a_i z^i)^{2^{m-1}} \\
&= \sum_{i=0}^{m-1} a_i (z^{2^{m-1}})^i \\
&= \sum_{i=0}^{m-1/2} a_{2i} (z^{2^{m-1}})^{2i} + \sum_{i=0}^{m-3/2} a_{2i+1} (z^{2^{m-1}})^{2i+1} \\
&= \sum_{i=0}^{m-1/2} a_{2i} z^i + \sum_{i=0}^{m-3/2} a_{2i+1} z^{2^{m-1}} z^i \\
&= \sum_{i \text{ even}} a_i z^{\frac{i}{2}} + \sqrt{z} \sum_{i \text{ odd}} a_i z^{\frac{i-1}{2}}.
\end{aligned}$$

This ultimately involves splitting a into its odd and even coefficients, then performing a field multiplication and addition.

If the irreducible polynomial is a trinomial and so represented as

$$f(z) = z^m + z^n + 1$$

or pentanomial and so represented as

$$f(z) = z^m + z^n + z^q + z^r + 1$$

then an efficient formula for calculating $\sqrt{z}$ can be used. For example,

$$\sqrt{z} = z^{\frac{m+1}{2}} + z^{\frac{n+1}{2}} (\text{mod } f(z))$$

when $f(z)$ is a trinomial and

$$\sqrt{z} = z^{\frac{m+1}{2}} + z^{\frac{n+1}{2}} + z^{\frac{q+1}{2}} + z^{\frac{r+1}{2}} (\text{mod } f(z))$$

when $f(z)$ is a pentanomial (assuming that m, n, q and r are odd). From the above, it can be understood that the process of calculating the square root involves splitting the value of interest into odd and even parts, hence it can be illustrated from a structural perspective as in Figure 5.2.

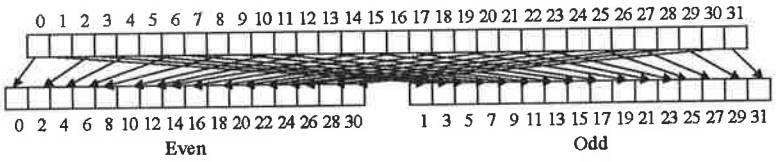

Figure 5.2: Square root of $\mathbb{F}(2^m)$ elements where $w = 32$.

The square root calculation involves a single operand. Hence, the square root operation will only be useful in a first-order power analysis attack if, when the input to the square root is secret, it is subsequently combined with some known data. Here, the selection function is of the form $\beta = S(\alpha, \sqrt{k})$. Therefore, an adversary can make predictions about portions of k, and determine portions of $\sqrt{k}$, which subsequently come in contact with known data. For this analysis it is assumed that the operation in which α and $\sqrt{k}$ are involved is a straightforward operation such as addition.

5.4.3.1 Hypotheses Generation

Let $\alpha_i \in \mathbb{F}_{2^m}$ be data known to the adversary, where $1 \leq i \leq N$. Let $k \in \mathbb{F}_{2^m}$ be a secret value unknown and of interest to the adversary. At some point in the target algorithm and on the target device, the operation $\alpha_i + \sqrt{k}$ will be calculated. It is assumed that numerous power traces t_i relating to the execution of $\alpha_i + \sqrt{k}$ have been acquired. Therefore, N power traces corresponding to the known data α_i should be in the possession of the adversary. Let β_i denote the m-bit result from the operation of $\alpha_i + \sqrt{k}$. The elements α_i, k and β_i will be represented as $\alpha_i = (\alpha_i[r-1], \alpha_i[r-2], \ldots, \alpha_i[0])$, $k = (k[r-1], k[r-2] \ldots k[0])$ and $\beta_i = (\beta_i[r-1], \beta_i[r-2], \ldots, \beta_i[0])$ respectively, where $r = \lceil \frac{m}{w} \rceil$ is the number of words required to store α_i, k and β_i, and u is used to index words, i.e. $0 \leq u < r-1$.

To uncover the secret k, CPA can be used to extract individual words of k at a time. Since the known data α_i, comes in contact with $\sqrt{k}$, knowledge about $\sqrt{k}$ given partial k is required. Figure 5.2, provides the necessary insight into how k and $\sqrt{k}$ are related. Given a word of k, $\frac{w}{2}$ bits of $\sqrt{k}$ are determined as follows:

Let $k[0]$ be the word of k to be extracted. The first step in the calculation of $\sqrt{k}$ involves k being partitioned into even and odd coefficients. Hence, $\frac{w}{2}$ bits of odd $k[0]$, and $\frac{w}{2}$ bits of even $k[0]$ will be known. Let k_e and k_o denote the even and odd coefficients of k and let $k_e[0]$ and $k_o[0]$ denote the even and odd coefficients of $k[0]$.

The second step involves the multiplication of two $\frac{m}{2}$ bit quantities, $k_o \cdot \sqrt{z}$. Since the multiplier $\sqrt{z}$ is known, and a portion of k_o is known from the guess from $k[0]$, a portion of the product can be calculated using the structural analysis technique of the multiplication operation as described in Section 5.4.2. This portion will be $\frac{w}{2}$ bits long.

The final step in the square root operation is the addition (or exclusive or) of k_e and the product $k_o \cdot \sqrt{z} \pmod{f(z)}$. Since $k_e[0]$ and $\frac{w}{2}$ bits of the product of $k_o[0] \cdot \sqrt{z}$ can be determined from $k[0]$, $\frac{w}{2}$ bits of $\sqrt{k}$ can thus be calculated.

Since there exists a deterministic link between portions of k and $\sqrt{k}$, by making

predictions about the possible values of words of k, and subsequently half words of the output of $\alpha_i + \sqrt{k}$, the power traces and the attack CPA can be used to determine which hypothesis for the word of k is correct. Algorithm 14 describes the steps to generate the data bank of hypothetical output for $\alpha_i + \sqrt{k}$.

Algorithm 14 Generate hypothetical output of the operation $\alpha_i[0] + \sqrt{k[0]}$.

INPUT: The set $\{\alpha_0, \alpha_1 ... \alpha_N\}$ where the set is known/computable by the adversary
OUTPUT: An $N \times 2^w$ matrix H_0 containing the output of $\alpha_i[0] + \sqrt{k[0]}$

```
for 1 ≤ i ≤ N do
    X = α_i[0]
    for 0 ≤ j < 2^w do
        k[0] = j
        k_e[0] = k[0] even coefficients
        k_o[0] = k[0] odd coefficients
        √k[0] = k_e[0] + (k_o[0] · √z)
        β_i[0] = X + √k[0]
        H_0(i, j) = β_i[0]
    end for
end for
return H_0
```

Once again a $N \times 2^w$ matrix H_0, to be input to the correlation test, will be produced from this algorithm. The process of establishing which guess for $k[0]$ is correct is described in Section 5.4.5.

Once the first word of k has been identified, $k[0]$, the remaining words of k can be found. For example, $k[0]$ affects the least significant lower half word of $\sqrt{k[0]}$, $k[1]$ affects the least significant upper half word of $\sqrt{k[0]}$, and so $k[0]$ and $k[1]$ affect the least significant word of $\sqrt{k}$, $\sqrt{k[0]}$.

A numerical example of how a portion of $\sqrt{k}$ can be deterministically calculated, given a portion of k is given in Appendix A.1.

5.4.4 Reduction

One attractive aspect of finite fields is the fact that all elements are contained within a particular finite group determined and enforced by the modulus. The procedure for modulo operations depends on the implementation, either lazy reduction can be performed [75],

where the reduction is performed at the end of an operation, or reduction can be interleaved with an operation. For example, the entire multiplication of $a \cdot b$ can be performed before it is reduced or at intermediate steps during the multiplication of say $a[0] \cdot b$, a reduction can be performed. When reduction is performed will affect how the predictions for the hypothetical output of $\beta = S(\alpha, k)$ are made. In the case of attack on the multiplication operations as described in Section 5.4.2, if lazy reduction is employed, then the attack can be applied as described. However, if interleaved reduction is used, the attack must be altered slightly so that the correct predictions are made.

For characteristic two implementations, the modulus is specially chosen such that it permits fast reduction. Specifically, irreducible trinomials or pentanomials are preferred. Straightforward reduction can be performed using the shift and subtract method, where subtract over $\mathbb{F}_{2^m}$ is XOR and subtract over $\mathbb{F}_p$ involves borrow bits. The reduction of $a(z) \equiv b(z) \pmod{f(z)}$ or $a \equiv b \pmod{p}$ basically involves lining the modulus up with the most significant bit of a (or $a(z)$) and subtracting to produce an intermediate value d. The modulus is then repeatedly lined up with intermediate d's until the bit length (or degree) of d is less than the bit length of p (or degree of $f(z)$).

If interleaved reduction is implemented, then it is more difficult to definitively calculate the hypothetical output of interest. For example, in the case of multiplication, prediction of partial output of $c \equiv a \cdot b \pmod{p}$, requires the ability to calculate all of the product $a \cdot b$. Knowing only portions of $a \cdot b$ is not sufficient since the waterfall effect of reduction will cause these portions to be lost in a manner unpredictable by the adversary.

5.4.4.1 Hypotheses Generation

Let $\alpha_i \in \mathbb{F}_p$ be data known to the adversary, where $1 \leq i \leq N$ and $n = \lceil \log_2 p \rceil$. Let $k \in \mathbb{F}_p$ be a secret value unknown and of interest to the adversary. At some point in the target algorithm and on the target device, the operation $\alpha_i \cdot k \pmod{p}$ will be calculated, where reduction is performed at intermediate steps during the multiplication $\alpha_i \cdot k[0]$

$(\text{mod } p)$. It is assumed that numerous power traces t_i relating to the execution of $\alpha_i \cdot k \pmod p$ have been acquired. Therefore, N power traces, corresponding to the known data α_i, should be in the possession of the adversary.

To uncover the secret k, CPA can be used to extract words of k at a time. To extract the least significant word of k, $k[0]$, CPA will correlate the hypothetical output of the intermediate output $d \equiv \alpha_i \cdot k[0] \pmod p$ for each prediction of $k[0]$, with the actual power consumption values.

Note that the operation $d \equiv \alpha_i \cdot k[0]$ will produce an $(n + w)$-bit result. d will subsequently be reduced by p to produce a n-bit value. Since the modulus is public, the resultant value will be used to verify the correct $k[0]$. Algorithm 15 describes the steps to generate the data bank of hypothetical output.

Algorithm 15 Generate hypothetical output of the multiplication $\beta_i[0] = \alpha_i \cdot k[0]$.

INPUT: The set $\{\alpha_0, \alpha_1, \ldots, \alpha_N\}$ where the set is known/computable by the adversary
OUTPUT: An $N \times 2^w$ matrix H_0 containing the output of $\beta_i[0] = \alpha_i \cdot k[0]$

```
for 1 ≤ i ≤ N do
    X = α_i
    for 0 ≤ j < 2^w do
        k[0] = j
        d = X · k[0] (mod p)
        H_0(i, j) = d
    end for
end for
return H_0
```

Once again a $N \times 2^w$ matrix H_0, to be input to the correlation test, will be produced from this algorithm. The process of establishing which guess for $k[0]$ is correct is described in Section 5.4.5.

To extract $k[1]$, partial hypothetical output of $(d \equiv \alpha_i \cdot k[1] \pmod p) + (d \equiv \alpha_i \cdot k[0] \pmod p)$, where it is assumed that $k[0]$ has been found, is calculated. This process is repeated until no words of k remain unknown. A similar process can be applied to reduction over the binary field $\mathbb{F}_{2^m}$.

5.4.5 Hypotheses Testing

Once the hypotheses generation phase is complete, the hypotheses are tested using the generated matrix H_l, where $0 \leq l < r$, and the data from the power traces t_i. The matrix H_l will be of dimensions $N \times 2^w$. This contains the hypothetical output of the target finite field operation given all possible values for $k[l]$ and known data α_i, i.e. $\beta_i = S(\alpha_i, k[l])$.

In CPA, the correlation is calculated between the estimated power consumption of a particular value and the actual power consumption. Therefore, H_l is translated into its associated estimated power consumption. This amount depends on the power model adopted. If the Hamming weight model is favoured, then H_l is replaced with its Hamming weight. If the Hamming distance model is preferred, then Hamming distance between the state H_l and the previous state R is calculated. Since R is an unknown constant reference state, the search space for this model is 2^{2w} as all previous reference states must be considered.

Consider the case of $k[0]$. To identify which is the correct $k[0]$, the correlation is calculated between the estimated power consumption of each column in H_0 (this contributes to Y in Equation (2.2) in Chapter 2) and a discrete time interval in the acquired power traces t_i where the target operation is being executed (this contributes to X in Equation (2.2) in Chapter 2). For example, the matrix H_0 is translated into its corresponding Hamming weight values as follows,

$$H_0 = \begin{pmatrix} \texttt{92A3EF02} & \texttt{3EEC627B} & \texttt{C63BFDC1} & \cdots \\ \texttt{045FBAD5} & \texttt{85EB1127} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix} \rightarrow \begin{pmatrix} 15 & 19 & 19 & \cdots \\ 17 & 15 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

Each column in the matrix H_0, is correlated to a discrete interval in the power traces t_i. Figure 5.3 depicts the typical power traces acquired[3]. Here the interval containing the

[3]This figure was generated from acquisitions captured during collaboration with University College Cork, Department of Electrical & Electronic Engineering.

target operation is highlighted. Each column in H_0, which correspond to the hypothetical output when $k[0] = 0$ up to $2^w - 1$, will be correlated to the area of interest.

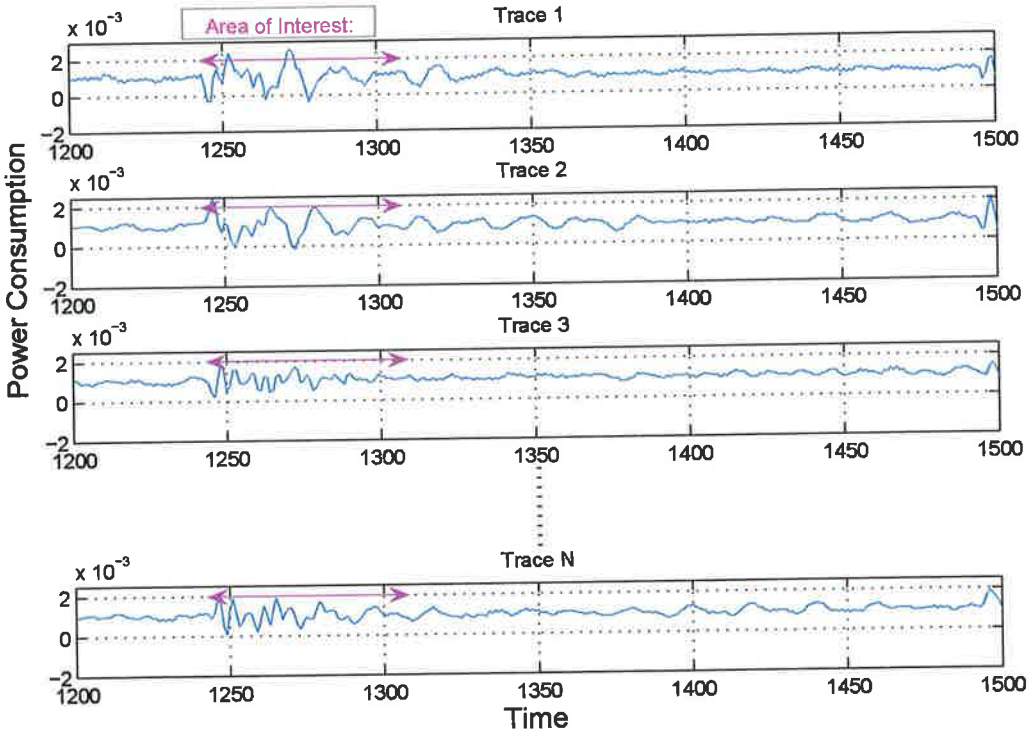

Figure 5.3: Discrete time interval in power traces t_i where target finite field operation is executing.

The hypothesis with the highest correlation, is identified as the correct least significant word $k[0]$. Identification of the remaining words of k proceeds in the same manner.

5.4.6 Computational Cost of Attacks

In each of the attacks described, w bits of the key k were guessed at a time. This means the total number of hypotheses that must be made for k are $2^w \times \frac{\lceil \log_2 k \rceil}{w}$. Therefore, the efficiency of the attack depends on w. In the implementation of Scott et al. [120], where $w = 32$, it will be a computationally intensive task to extract one word. However, it is

possible to calculate the partial correlation of practical portions of w.

The authors of [24], who developed CPA, pointed out that if the correlation coefficient of f independent bits amongst w is calculated, a partial correlation still exists and can be predicted as a function of the coefficient that would be generated if all the bits of w were included. This proportion is determined by

$$\rho_{(X,Y)_{f/w}} = \rho_{X,Y}\sqrt{\frac{f}{w}} \tag{5.1}$$

where f bits from a w-bit word are known. Since the estimations for the power consumption now only considers portions of the word, the correlation will be proportional to the portion of the word considered, and so partial correlation will be witnessed.

This can be used to attack implementations on processors with large word sizes, and so is particularly relevant for this work. For example, consider the multiplication of two values a and b to produce c, where a is known and b is unknown and the multiplication is carried out on a 32 bit processor. As highlighted in Section 5.4.2, the least significant word of the product c, is determined by the least significant words of a and b. Let $a[0]$, $b[0]$ and $c[0]$ denote the least significant word of a, b and c respectively. Furthermore, let $b_{LSB}[0]$, $b_{MLSB}[0]$, $b_{LMSB}[0]$, $b_{MSB}[0]$ denote the four bytes of $b[0]$. Firstly, hypotheses can be made for $b_{LSB}[0]$ and the relating hypotheses tested. Since there only exist 2^8 hypotheses, this is feasible to do. The next byte of $b[0]$, $b_{MLSB}[0]$ can then be tested in the same way. Based on Equation (5.1), $\sqrt{\frac{8}{32}} = 0.5$ of the correlation that should be produced if all the bits were predicted, should be witnessed.

Once the value (or the most likely candidate) for each byte has been identified, to gain further confidence in the bytes for $b_{LSB}[0]$ and $b_{MLSB}[0]$, the correlation for the least significant 16 bits of $b[0]$ can be determined. The attacker should witness an increase in the correlation which should again be proportional to the correlation that should be witnessed given the correct 32 bits, specifically $\sqrt{\frac{16}{32}} = 0.707$ of this correlation should

be seen. This can be continued for the remaining bytes of $b[0]$. This process requires the generation of $4 \times 2^8 = 2^{10}$ hypotheses, which is considerably less that if extracting a word of b, where a total of 2^{32} hypotheses would be required. Figure 5.4 demonstrates typical partial correlations that might be witnessed for different size portions of a 32 bit word[4].

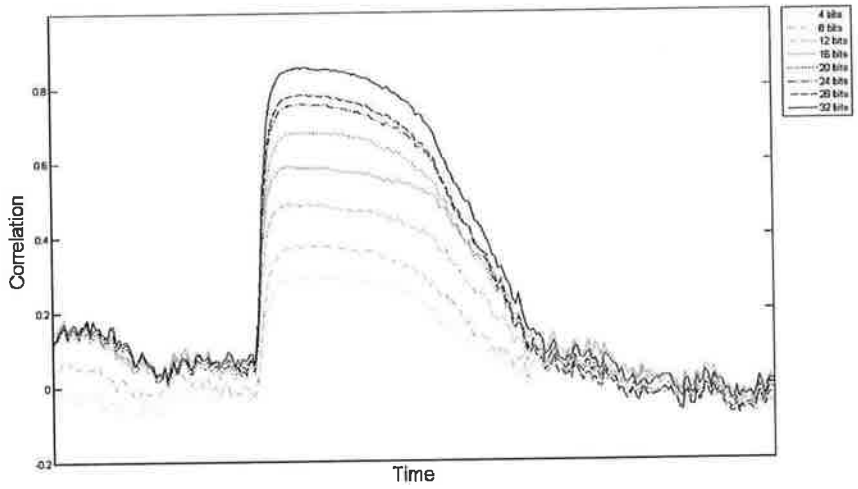

Figure 5.4: Various partial correlation peaks for a 32 bit word. As can be seen, the correlation for 4, 8, 16 bits is proportional to the correlation for the full 32 bits.

5.5 Power Analysis of Candidate Bilinear Pairings

The susceptibility of finite field operations to power analysis has implications for the vulnerability of the reliant pairing. When and how the various finite field operations are used in the pairing will affect the attack options for first-order power analysis. In this section, three particularly efficient pairing algorithms, which are presented in [120], are analysed for potential vulnerability to first-order power analysis. Building on the analysis of the previous section, that is, given that the underlying finite field operations can be

[4]This image is courtesy of Gemplus Research and Development Laboratories [52].

attacked as described, the consequences are assessed in the following pairing algorithms.

5.5.1 Attack Assumptions

The following assumptions are made about the attack on the target bilinear pairings. Firstly, the device in which the bilinear pairing algorithm is stored and executed, is a valid target for side channel attacks such as power analysis. This means that it is possible to capture a power consumption measurement, referred to as a power trace, from the target device.

Secondly, it is assumed that all parameters bar a secret elliptic curve point are published. Therefore, the finite field $\mathbb{F}_{q^k}$, the elliptic curve E, the other point entered into the pairing (P if Q is secret and Q if P is secret), and order of the Miller loop r are accessible to an adversary.

Thirdly, it is assumed that the adversary can invoke multiple pairing computations with various valid input points. This allows the adversary to capture the power consumption of a number of pairing computations. The number of times that the pairing is executed, and corresponding power traces acquired, is N.

Finally, the class of attacks that will be considered are first-order power analysis attacks such as CPA, which are based on the availability of an eligible selection function.

5.5.2 The BKLS Algorithm for the Tate Pairing

The BKLS algorithm [14] for the Tate pairing $e(P, Q)$ is implemented on non-supersingular elliptic curves over the large prime field $E(\mathbb{F}_{p^k})$. The pairing accepts a point $P \in E(\mathbb{F}_p)$ of order r and a point $Q \in E'(\mathbb{F}_{p^{k/d}})$ on a twist of the curve. $e(P, Q)$ is evaluated to an element in the finite field $\mathbb{F}_{p^k}$. In the specific implementation described in [120], the embedding degree $k = 2$ and $d = 2$. Hence, the points P and Q have coordinates in $\mathbb{F}_p$.

The algorithm for the Tate pairing is described in Chapter 3. Briefly to recap, in this pairing the point scalar multiplication of the point $[r]P = \mathcal{O}$ is calculated involving

calculation of a distance relationship between the lines produced during the point multiplication and the point Q. This distance relationship is the heart of Miller's loop, and is calculated as

$$y_j - \lambda_j(x_Q + x_j) - y_Q i \tag{5.2}$$

where an element of the form $a + ib \in \mathbb{F}_{p^2}$ is produced with $a, b \in \mathbb{F}_p$. Usually a and b are stored separately.

Problem: *Given that the BKLS algorithm for the Tate pairing $e(P, Q)$ is the pairing implementation that is stored and being executed on a target device D, is it susceptible to first-order power analysis?*

Two scenarios will be considered.

Scenario I. The cryptographic protocol requires the computation of $e(P, Q)$, where P is a public parameter (x_P, y_P) and Q is a private parameter (x_Q, y_Q).

As described previously, during the computation of the Tate pairing, the P parameter is dynamic and the Q parameter is static. Since the order r is a publicly available parameter, the intermediate points $[j]P$ and so (x_j, y_j) and the slope λ_j, which are calculated during a point addition or doubling in the computation of the point $[r]P$, can be generated by the adversary.

From the perspective of an adversary utilising the power consumption, steps in the algorithm involving the secret with known data are of particular interest. Therefore, any meeting point between x_P, y_P, x_j, y_j and λ_j, and x_Q and y_Q will be subject to attack. The contact points that exist are tabulated in Table 5.1. These contact points are potential selection functions.

From this table it can be seen that there are multiple options to attack x_Q in every

Round	Operation	Secret
1	$x_Q + x_P$	x_Q
1	$\lambda_{P,2P}(x_Q + x_P)$	x_Q
1	$y_P - \lambda_{P,2P}(x_Q + x_P)$	x_Q
2 to $(\lfloor \log_2 r \rfloor - 2)$	$x_Q + x_j$	x_Q
2 to $(\lfloor \log_2 r \rfloor - 2)$	$\lambda_j(x_Q + x_j)$	x_Q
2 to $(\lfloor \log_2 r \rfloor - 2)$	$y_j - \lambda_j(x_Q + x_j)$	x_Q

Table 5.1: Attack options for the Tate Pairing: P public, Q private.

round, yet no opportunities to attack y_Q exist. This is due to the fact that y_Q is not combined with known data at any stage during the algorithm. Hence, no eligible selection function can be constructed. Explicitly, in round one the operations that y_Q is involved in can be described as

$$ac - b^2 \quad \text{and} \quad (a+b)(b+c) - ac - b^2$$

where b represents y_Q and a and c are unknown. In all subsequent rounds, the operations that y_Q is involved in are of the form

$$ac - bd \quad \text{and} \quad (a+b)(d+c) - ac - bd$$

where again b relates to y_Q and a, c and d are unknown. Therefore, no selection function exists for the secret coordinate y_Q and so first-order power analysis is only possible on the coordinate x_Q. Due to point compression however, x_Q alone is sufficient to extract the secret Q.

To extract x_Q any of the operations listed in Table 5.1 can be focused on. For demonstrative purposes, the operation $\lambda_j(x_Q + x_j)$ will be assessed. Given the point P, the order r and the modulus p, the intermediate values x_j and λ_j which arise in the calculation of $[r]P$ can be computed.

As previously demonstrated, the multiplication operation can be attacked using power

analysis when operating on known and secret data. Therefore, given x_j and λ_j, predictions can be made about portions of x_Q, which can be verified using the hypothetical output of $\lambda_j(x_Q + x_j)$ and the power traces t_i.

Let the target round in the Miller loop be the second round. Let x_2 and λ_2 denote the values related to P involved in $\lambda_2(x_Q + x_2)$. Let $\lambda_2[0]$, $x_2[0]$ and $x_Q[0]$ denote the least significant words of λ_2, x_2 and x_Q respectively. By predicting possible values for $x_Q[0]$, the hypothetical output word $r_2[0]$ where $r_2[0] = \lambda_2[0](x_2[0] + x_Q[0])$, can be computed. Then, for all points P_i input to the pairing $e(P_i, Q)$ where $1 \leq i \leq N$ and possible values of $x_Q[0]$, the hypothetical output of the function $\lambda_2[0](x_2[0] + x_Q[0])$ is calculated to populate the matrix H_0. The matrix H_0 is in turn translated into its estimated power consumption values and compared with the actual power. This will determine which hypothesis for $x_Q[0]$ gives the highest correlation. This procedure is used to extract the remainder of x_Q.

Since the secret x_Q is combined with known data in every round of the Miller loop, it is possible to further validate a guess for $x_Q[0]$ by checking the correlation at multiple rounds. For example, given that the value g has exhibited the highest correlation for the least significant word of x_Q in $r_2[0] = \lambda_2[0](x_2[0] + x_Q[0])$, it can be validated further by checking the correlation of

$$r_3[0] = \lambda_3[0](x_3[0] + g)$$

and

$$r_4[0] = \lambda_4[0](x_4[0] + g).$$

A numerical example of how the hypothetical output of $\lambda_j(x_Q + x_j)$ is deterministically calculated given x_j, λ_j and portions of x_Q is given in Appendix A.2.

Scenario II. The cryptographic protocol requires the computation of $e(P, Q)$, where P is

a private parameter and Q is a public parameter.

In the scenario where P is secret and Q is public, the known value remains static through the computation, and the secret parameter is constantly changing. The values x_P, y_P and intermediate points $[j]P$ and so x_j, y_j and λ_j, which are calculated during the computation of the point $[r]P$, are now secret. Attacking the first round will reveal x_P and y_P directly. Attacking any other round will reveal x_j and y_j, and so will require knowledge of the number of additions that have been used to produce $[j]P$, so the required number of point subtractions can be applied to reveal the original point P.

Again, from the perspective of an adversary utilising the power consumption, steps in the algorithm involving the secret with known data are of interest. The potential selection functions that exist in this case are tabulated in Table 5.2.

Round	Operation	Secret
1	$x_Q + x_P$	x_P
1	$\lambda_{P,2P}(x_Q + x_P)$	x_P, $\lambda_{P,2P}$
1	$y_P - \lambda_{P,2P}(x_Q + x_P)$	x_P, $\lambda_{P,2P}$, y_P
2 to $(\lfloor \log_2 r \rfloor - 2)$	$x_Q + x_j$	x_j
2 to $(\lfloor \log_2 r \rfloor - 2)$	$\lambda_j(x_Q + x_j)$	x_j, λ_j
2 to $(\lfloor \log_2 r \rfloor - 2)$	$y_j - \lambda_j(x_Q + x_j)$	x_j, λ_j, y_j

Table 5.2: Attack options for the Tate Pairing: P private, Q public.

From this table it can be seen that there exists no options to extract the secret coordinate y_P and the only operations where x_P is accessible are $x_Q + x_P$ and $x_Q + x_j$. All other operations involve more than one unknown value. Therefore, in implementations where P is private and Q is public, first-order power analysis relies on the attack on the addition operations.

Also, unlike the situation where Q is the secret, the focus of the attack can be on one round at a time. Multiple rounds to strengthen the confidence of a guess for a portion of x_P cannot be performed, since the portion of x_j used in one round will not be related to

the portion of x_j used in the next round.

5.5.3 The BGOhES algorithm for the η_T Pairing

The BGOhES algorithm [13] for the η_T pairing $\eta_T(P, Q)$ is implemented on supersingular elliptic curves over the binary field $E(\mathbb{F}_{2^m})$. Both parameters to the pairing P and Q are points on the curve $E(\mathbb{F}_{2^m})$. Hence P and Q will have coordinates in $\mathbb{F}_{2^m}$ and so the coordinates (x, y) will be approximately the same bit length as m. $\eta_T(P, Q)$ is evaluated to an element in $\mathbb{F}_{2^{km}}$. In the specific implementation described in [120], the embedding degree $k = 4$.

The algorithm for the η_T pairing is described in Chapter 3. The η_T pairing is quite different to the Tate and Ate pairing: no distortion map is explicitly used as the map to the extension field is integrated into the algorithm, operations on points are completely avoided. A preliminary computation takes place outside the Miller loop:

$$m \leftarrow u \cdot (x_P + x_Q + 1) + y_P + y_Q + b + 1 + (u + x_Q)s + t,$$

and the iterative square root of x_P and y_P and squaring of x_Q and y_Q is required throughout the Miller loop. The distance relationship at the heart of Miller's loop is calculated as

$$g \leftarrow u \cdot (x_P + x_Q) + y_P + y_Q + x_P + (u + x_Q)s + t$$

where an element $a + bz + cz^2 + dz^3 \in \mathbb{F}_{2^{4m}}$ is produced with a, b, c and $d \in \mathbb{F}_{2^m}$. Generally a, b, c and d are stored separately. Note that $s = (0, 1, 1, 0)$ and $t = (0, 1, 0, 0) \in \mathbb{F}_{2^{4m}}$ are public elements that are derived from the distortion map and used to map elements from the base field to the extension field.

Problem: *Given that the BGOhES algorithm for the η_T pairing $\eta_T(P, Q)$, is the pairing implementation stored and being executed on a target device D, is it susceptible to first-*

order power analysis?

Two scenarios will be considered.

Scenario I. The cryptographic protocol requires the computation of $\eta_T(P, Q)$, where P is a public parameter and Q is a private parameter.

During the computation of the η_T pairing both parameters, P and Q are dynamic. The values x_P, y_P and intermediate values x_i and y_i, which are related to the repeated square root of x_P and y_P respectively, can be computed by the adversary. The secret values x_Q, y_Q or intermediate values x_j and y_j, which are related to the repeated squaring of x_Q and y_Q respectively, are of interest to the adversary. Attacking outside the Miller loop or the first round of the Miller loop will reveal x_Q and y_Q directly. Attacking any other round will reveal x_j and y_j, and will require knowledge of the number of squarings that have been applied to x_Q and y_Q, so the required number of square root operations can be applied to reveal the original point Q.

Any steps in the algorithm involving the secret with known data will be open to power analysis. Therefore, any meeting point between x_P, y_P, x_i and y_i, and x_Q, y_Q x_j and y_j will be subject to attack. The contact points that exist are tabulated in Table 5.3. These contact points are again eligible selection functions.

Round	Operation	Secret
Outside loop	$(x_P + 1) \cdot (x_P + x_Q + 1) + y_P + y_Q + b + 1$	x_Q, y_Q
Outside loop	$(x_P + 1 + x_Q)s$	x_Q
1	$x_P \cdot (x_i + x_Q) + y_i + y_Q + x_i$	x_Q, y_Q
1	$(x_P + x_Q)s$	x_Q
2 to $(m+1)/2$	$x_{i-1} \cdot (x_i + x_j) + y_i + y_j + x_i$	x_j, y_j
2 to $(m+1)/2$	$(x_{i-1} + x_j)s$	x_j

Table 5.3: Attack options for the η_T Pairing: P public, Q private.

From this table it can be seen that there are multiple options to attack x_Q or a relation of x_Q, x_j, and y_Q or a relation of y_Q, y_j, in every round. In addition, a number of operations can be combined to validate a prediction for a guessed portion. For example,

$$(x_P[0] + 1) \cdot (x_P[0] + x_Q[0] + 1)$$

and

$$x_P[0] \cdot (x_i[0] + x_Q[0])$$

involving addition and multiplication modulo the known irreducible polynomial $f(z)$, can be used to validate a prediction for the portion $x_Q[0]$. Given that the deterministic output for guesses of portions of x_Q can be made in the above operations, CPA can be used to determine portions of x_Q using the power traces.

Scenario II. The cryptographic protocol requires the computation of $\eta_T(P, Q)$, where P is a private parameter and Q is a public parameter.

The η_T pairing is different from the Tate and Ate pairing algorithms assessed in that the two paths in the pairing are almost symmetric and so are equally vulnerable. The values x_Q, y_Q or intermediate values x_j and y_j, which are related to the repeated squaring of x_Q and y_Q, can be computed by the adversary. The secret values x_P, y_P and intermediate values x_i and y_i, which are related to the repeated square root of x_P and y_P, are of interest to the adversary. Attacking the outside the Miller loop or the first round of the Miller loop will reveal x_P and y_P directly. Attacking any other round will reveal x_i and y_i and so will require knowledge of the number of square root operations that have been applied to x_P and y_P, so that the required number of squaring operations can be applied to reveal the original point P. The steps in the algorithm that involve the secret interacting with known data are tabulated in Table 5.4.

Round	Operation	Secret
Outside loop	$(x_P+1)\cdot(x_P+x_Q+1)+y_P+y_Q+b+1$	x_P, y_P
Outside loop	$(x_P+1+x_Q)s$	x_P
1	$x_P\cdot(x_i+x_Q)+y_i+y_Q+x_i$	x_P, $x_i=\sqrt{x_P}$, $y_i=\sqrt{y_P}$
1	$(x_P+x_Q)s$	x_P
2 to $(m+1)/2$	$x_{i-1}\cdot(x_i+x_j)+y_i+y_j+x_i$	x_{i-1}, x_i, y_i
2 to $(m+1)/2$	$(x_{i-1}+x_j)s$	x_{i-1}, x_i

Table 5.4: Attack options for the η_T Pairing: P private, Q public.

An equal number of opportunities to attack x_P or a relation of x_P, x_i, and y_P or a relation of y_P, y_i, exist when the point P is the secret. In addition to calculating the hypothetical output for any of the operations listed in Table 5.4, the structural analysis of the square root operation can be used in the attack of x_P. For example, given $x_P[0]$, it was previously demonstrated how the least significant $\frac{w}{2}$ bit portion of $\sqrt{x_P}$ could be determined from $x_P[0]$. Therefore, a guess can be made for $x_P[0]$ and subsequently validated by $x_P[0]\cdot(\sqrt{x_P[0]}+x_Q)$.

5.5.4 The Ate Pairing

The Ate pairing $a(P,Q)$ is implemented on non-supersingular elliptic curves over the large prime field $E(\mathbb{F}_{p^k})$. The pairing accepts a point $P\in E'(\mathbb{F}_{p^{k/d}})$ of order r over the twisted curve, and a point $Q\in E(\mathbb{F}_p)$ over the base curve. $a(P,Q)$ is evaluated to an element in the finite field $\mathbb{F}_{p^k}$. In the specific implementation described in [120], the embedding degree $k=4$ and $d=2$. Hence the point P has coordinates in $\mathbb{F}_{p^2}$ and Q has coordinates in $\mathbb{F}_p$.

The algorithm for the Ate pairing is provided in Chapter 3. Briefly to recap, in this pairing the point scalar multiplication of the point $[r]P=\mathcal{O}$ is calculated involving calculation of a distance relationship between the lines produced during the point multiplication and the point Q. This distance relationship is the heart of Miller's loop, and is

calculated as

$$i^2 y_Q - i(i^2 y_j/2 + \lambda_1(i^2 x_j/2 + x_Q)) \tag{5.3}$$

where an element of the form $a + ib \in \mathbb{F}_{p^4}$ is produced with $a, b \in \mathbb{F}_{p^2}$. This formula is derived by combining the line evaluation with the coordinates of P "untwisted" from $E'(\mathbb{F}_{p^2})$ to $E(\mathbb{F}_{p^4})$. Note that point scalar multiplication is calculated over the field $E'(\mathbb{F}_{p^2})$ which means the underlying finite field arithmetic fundamental to point addition and doubling is performed over $\mathbb{F}_{p^2}$.

Problem: *Given that the Ate pairing $a(P, Q)$, is the pairing implementation stored and being executed on a target device D, is it susceptible to first-order power analysis?*

Two scenarios will be considered.

Scenario I. The cryptographic protocol requires the computation of $a(P, Q)$, where P is a public parameter and Q is a private parameter.

As with the Tate pairing, the P parameter is dynamic and the Q parameter is static. Since the order r is a publicly available parameter, the intermediate points $[j]P$ and so x_j, y_j and λ_j, which are calculated during the computation of the point $[r]P$, can be generated by the adversary. The potential selection functions involving known data values x_P, y_P, x_j, y_j and λ_j, and the secret data values x_Q and y_Q are tabulated in Table 5.5.

Round	Operation	Secret
1	$-i(i^2 y_P/2 + \lambda_{P,2P}(i^2 x_P/2 + x_Q))$	x_Q
2 to $(\lfloor \log_2 n \rfloor - 1)$	$-i(i^2 y_j/2 + \lambda_j(i^2 x_j/2 + x_Q))$	x_Q

Table 5.5: Attack options for the Ate Pairing: P public, Q private.

From this table it can be seen that opportunities exist to attack x_Q in every round, yet

none exist to attack y_Q. This is similar to the case in the Tate pairing where y_Q only comes in contact with other unknown values. To extract x_Q, the operation $\lambda_j(i^2x_j/2 + x_Q)$ which involves addition of an element in $\mathbb{F}_p$ to an element in $\mathbb{F}_{p^2}$ and multiplication over $\mathbb{F}_{p^2}$, can be isolated. To attack this operation, structural analysis of the multiplication operation can once again be used. Even though multiplication is performed over $\mathbb{F}_{p^2}$, multiplication in $\mathbb{F}_{p^2}$ is constructed from multiplication in $\mathbb{F}_p$. Let $i^2x_j/2 = (x_1, y_1)$, $\lambda_j = (x_2, y_2)$ and $x_Q = (x_3, y_3)$ where x_i and $y_i \in \mathbb{F}_p$. Firstly, internal addition, which involves addition in $\mathbb{F}_p$, yields (x_1+x_3, y_1) since $x_Q \in \mathbb{F}_p$ and so $y_3 = 0$. The addition is followed by multiplication of the result $(x_1 + x_3, y_1)$ by λ_j. Again a series of operations over $\mathbb{F}_p$ are used to compute this product. The multiplication of $(x_2, y_2) \cdot (x_1 + x_3, y_1)$ is calculated as

$$(x_2 \cdot (x_1 + x_3) - y_1 \cdot y_2, (x_2 + y_2)(y_2 + (x_1 + x_3)) - (x_2 \cdot (x_1 + x_3) - y_1 \cdot y_2),$$

where internal multiplications are over $\mathbb{F}_p$. In this equation, all coordinates bar x_3 are known. Since x_3 is in fact x_Q, by guessing portions of x_Q the hypothetical output of any of the above operations can be used to verify a guess using the power traces. For example, $x_2 \cdot (x_1 + x_3)$ will serve as an eligible selection function.

In addition, since the secret x_Q is combined with known data in every round of the Miller loop, it is possible to further validate a guess for $x_Q[0]$ by checking the correlation at multiple rounds.

Scenario II. The cryptographic protocol requires the computation of $a(P, Q)$, where P is a private parameter and Q is a public parameter.

In the scenario where P is secret and Q is public, the known value remains static through the computation, and the secret parameter is constantly changing. The values x_P, y_P and

intermediate points $[j]P$ and so x_j, y_j and λ_j, which are calculated during the computation of the point $[r]P$, are now secret. Attacking the first round will reveal x_P and y_P directly. Attacking any other round will reveal x_j and y_j and so will require knowledge of the number of additions that have been used to produce $[j]P$, so the required number of point subtractions can be applied to reveal P. The potential selection functions in the algorithm that involve the secret interacting with known data are tabulated in Table 5.6.

Round	Operation	Secret
1	$-i(i^2y_P/2 + \lambda_1(i^2x_P/2 + x_Q))$	x_P, y_P
2 to $(\lfloor \log_2 n \rfloor - 1)$	$-i(i^2y_j/2 + \lambda_1(i^2x_j/2 + x_Q))$	x_j, y_j

Table 5.6: Attack options for the Ate Pairing: P private, Q private.

The attack of the Ate pairing is less forthcoming when the secret is P. When P is secret, the secret coordinate is over $\mathbb{F}_{p^2}$ as opposed to $\mathbb{F}_p$ as in the previous case. This not only affects the complexity of the point addition and doubling operations which are over the extension field $\mathbb{F}_{p^2}$, but affects the size of the secret that must be extracted. The only option for attack is the coordinates related to x_P, accessible in the either of the operations $i^2x_P/2+x_Q$ or $i^2x_j/2+x_Q$. All other operations involve more than one unknown value. For instance, the output of the operation

$$\lambda_1(i^2x_P/2 + x_Q))$$

given guesses for portions of the secret x_P, cannot be made. Therefore, in implementations where P is private and Q is public, first-order power analysis will rely on the attack on $i^2x_P/2 + x_Q$ or $i^2x_j/2 + x_Q$.

In addition, unlike the situation where Q is the secret, the focus of the attack can be on one round at a time. Multiple rounds to strengthen the confidence of a guess for a portion of x_P cannot be performed, since the portion of x_j used in one round will not be related to the portion of x_j used in the next round.

5.5.5 Summary of Findings

Three candidate bilinear pairings were assessed for vulnerability to first-order power analysis. This analysis was performed based on the criteria required for first-order power analysis to be possible. The main findings are as follows.

For the Tate pairing: The Tate pairing:

- For $e(P, Q)$, where P is public and Q is private, there exists multiple access points for first-order power analysis of the coordinate x_Q at each and every round. In addition, multiple rounds can be combined to obtain a stronger level of confidence in the guesses for x_Q.
- For $e(P, Q)$, where P is public and Q is private, no access points for first-order power analysis of the coordinate y_Q exist.
- For $e(P, Q)$, where P is private and Q is public, access points for first-order power analysis of the coordinate x_P are restricted to the operations: $x_Q + x_P$ and $x_Q + x_j$, and so a single round for either coordinate. In addition, if the attack focuses on x_j, then the required subtractions will have to be applied in order to access the secret P.
- For $e(P, Q)$, where P is private and Q is public, no access points for first-order power analysis of the coordinate y_P exist.
- **Summary:** The P path presents fewer options to attack than the Q path. Hence, it is recommended that the Tate pairing is executed as $e(S, \cdot)$, where S is the secret elliptic curve point.

The η_T pairing:

- For $\eta_T(P, Q)$, where P is public and Q is private, there exists multiple access points for first-order power analysis of both coordinates x_Q and y_Q at each and

every round. However, multiple rounds cannot be combined to obtain a stronger level of confidence for the guesses for x_Q or y_Q.

- For $\eta_T(P, Q)$, where P is private and Q is public, there exists multiple access points for first-order power analysis of both coordinates x_P and y_P at each and every round. However, multiple rounds cannot be combined to obtain a stronger level of confidence for the guesses for x_P or y_P.
- **Summary:** Both paths, P and Q are equally vulnerable to first-order power analysis in the η_T pairing.

The Ate pairing:

- For $a(P, Q)$, where P is public and Q is private, there exists multiple access points for first-order power analysis of the coordinate x_Q at each and every round. In addition, multiple rounds can be combined to obtain a stronger level of confidence in the guesses for x_Q.
- For $a(P, Q)$, where P is public and Q is private, no access points for first-order power analysis of the coordinate y_Q exist.
- For $a(P, Q)$, where P is private and Q is public, access points for first-order power analysis of the coordinate x_P are restricted to the operations: $i^2 x_P/2 + x_Q$ or $i^2 x_j/2 + x_Q$, and so a single round for either coordinate. In addition, if the attack focuses on x_j, then the required subtractions will have to be applied in order to access the secret P.
- For $a(P, Q)$, where P is private and Q is public, no access points for first-order power analysis of the coordinate y_P exist.
- **Summary:** The P path presents fewer options to attack than the Q path. Hence, it is recommended that the Ate pairing is executed as $a(S, \cdot)$, where S is the secret elliptic curve point.

These findings can summarised as in Table 5.7.

	P Public, Q Private		P Private, Q Public	
	x_Q	y_Q	x_P	y_P
$e(P,Q)$	✓, $\{+,\cdot\}$, M	-	✓, $\{+\}$, S	-
$\eta_T(P,Q)$	✓, $\{+,\cdot\}$, S	✓, $\{+,\cdot\}$, S	✓, $\{+,\cdot,\sqrt{\ }\}$, S	✓, $\{+,\cdot,\sqrt{\ }\}$, S
$a(P,Q)$	✓, $\{+,\cdot\}$, M	-	✓, $\{+\}$, S	-

Table 5.7: Summary: ✓ indicates the presence of an avenue of attack, – indicates the absence of one. The accompanying symbols $+$, $\cdot$ or $\sqrt{\ }$, indicate what operations are targeted. The accompanying letter S or M indicate whether a single round or multiple rounds can be used to validate an hypothesis.

5.6 Countermeasures

The main approach to deter first-order power analysis is to break the link between known data and secret data so $\beta \neq S(\alpha, k)$. This disables the adversary from forming hypotheses. A number of countermeasures have already been anticipated to protect bilinear pairings against SCA [118, 107], which achieve this effect.

The property of bilinearity allows the either the known or the secret parameter in the pairing to be easily blinded. A pairing can be calculated as

$$e(P,Q) = e(aP, bQ)^{1/ab}$$

where a and b are random values in $\mathbb{F}_q$, or

$$e(P,Q) = \frac{e(P, Q+R)}{e(P,R)}$$

where R is a random point in $E(\mathbb{F}_q)$ and e is the bilinear pairing in question.

While these may be effective in deterring SCA since a new random value will be used every time the bilinear pairing is called, they are expensive, ultimately requiring point scalar multiplication and calculation of two pairings respectively.

A more subtle countermeasure proposed by [118] observes that repeated multiplication of the Miller variable m by a random element in $\mathbb{F}_p$ will have no effect on the final pairing value since they will be eliminated in the final exponentiation. This is a less expensive deterrent only requiring a field multiplication per iteration of the Miller loop. In order for this countermeasure to be effective, the random value must not only be multiplied by the Miller variable, but must be multiplied by all intermediate values that make up the Miller variable. For example in the case of Tate (where the original unrandomised line function is given in Algorithm 4):

$$m_j = r \cdot y_j - \lambda_j(r \cdot x_Q + r \cdot x_j) - r \cdot y_Q i$$

where $r \in \mathbb{F}_p$. If a new random value is multiplied at every iteration of the loop, then first-order power analysis would no longer be possible.

5.7 Conclusion

In this chapter, it was shown that the BKLS algorithm for the Tate pairing, the BOGhES algorithm for the η_T pairing, and the Ate pairing, in theory exhibit characteristics that enable first-order power analysis. Firstly, a novel technique to attack finite field operations using CPA was given. It was shown how finite field multiplication, square root and reduction could be attacked by analysing the structural evolution of the operation. This technique, which was derived based on empirical knowledge about CPA, is algorithm independent.

It was then detailed how the attack on the finite field operations, upon which bilinear pairings are constructed, has implications for the security of the reliant bilinear pairing. Each of the candidate bilinear pairing algorithms were assessed for first-order power analysis, given the attack of the finite field operations. Consequently, a description of possible attacks of each pairing was detailed. Depending on the path that the secret takes, each

bilinear pairing is open to varying degrees of a power analysis attack. The overall findings show that the two paths for the Tate and the Ate pairing present different opportunities for attack, i.e. the Q path is more accessible than the P path. In contrast, the η_T pairing, while found to be the most efficient pairing on the smart card [120], is the least secure against first-order power analysis with both paths being equally open to attack. Preventative mechanisms to deter first-order power analysis, were subsequently detailed.

It should be noted that the work presented in this chapter is theoretical. Execution of the attacks in practice requires access to the device on which the analysed bilinear pairings are implemented. To date such hardware implementations has not been available. In the next chapter, bilinear pairings will be assessed for vulnerability to fault analysis.

Chapter 6

Fault Analysis of Bilinear Pairings

6.1 Introduction

In this chapter, the security of bilinear pairings in the context of fault analysis is examined. In particular, the Weil pairing, the η pairing and the Tate pairing are investigated. As already discussed, there has been concentrated research into the computation of efficient bilinear pairings. Endeavours to find suitable elliptic curves [15, 39, 16], optimisations to Miller's algorithm [14] and simplification of the final exponentiation [41], have produced fast bilinear pairings. However, there reaches a point where efficiency can compromise security. In the previous chapter, implementation attacks that passively monitor the execution were investigated. As demonstrated in Chapter 2, active implementation attacks such as fault analysis are also very powerful attacks, posing a serious threat to any cryptographic algorithm. Therefore, fault analysis of bilinear pairings is also necessary. In particular, features which are incorporated to optimise the pairing must be examined to determine whether they may actually weaken it.

The first, and only to date, description of a fault attack on bilinear pairings was by Page and Vercauteren [107], when they demonstrated an attack on the Duursma-Lee [37] and Kwon-BGOS [71, 13] algorithm for the Tate and η pairing. The fault type they

focused on was one which caused the Miller loop to run over. By inducing extra iterations they are able to isolate a single contribution to the Miller loop, which could be trivially picked apart to find the secret. However, a range of faults exist, which can have various effects on an algorithm. In this chapter, another fault type is focused on, namely faults which cause data used in the computation to be corrupted in some way. This type of fault attack, and its consequences, is investigated for a range of pairing algorithms.

Firstly the work of Page and Vercauteren is reviewed in Section 6.2. In Section 6.3, an overview of the type of fault attack described in this thesis is presented. This type of fault attack is entitled a data corruption fault attack. An overview of the candidate bilinear pairings that are assessed is presented. The general idea behind why such a fault attack can facilitate the extraction of the secret elliptic curve point is also discussed.

In Section 6.4, concrete instances of a data corruption fault attack are given. In particular, fault attacks on the η and Weil pairings, which enable extraction of the secret elliptic curve point, are detailed. An argument as to why such attacks are not applicable to the Tate pairing is also given, leading to an argument that the final exponentiation is vital in deterring fault attacks. Evidence is provided to support this argument by demonstrating that pairings having no final exponentiation or a simple final exponentiation are susceptible to data corruption fault attacks, whereas pairings with more complex final exponentiations are resistant to such attacks. This section is concluded by presenting a summary of the findings.

Countermeasures to deter fault attacks come in two main forms, fault detection mechanisms and fault prevention mechanisms. Fault detection mechanisms determine whether or not a fault has been injected. Fault prevention mechanisms obfuscate the execution so that even if a fault has been injected, the effects of the fault are completely lost and so useless to an adversary. In Section 6.5, countermeasures that exploit essential properties of pairings are presented. Both fault detection and fault obfuscation approaches to thwarting fault attacks are addressed.

Finally this chapter is concluded in Section 6.6. Appendix B of this dissertation contains numerical examples of the attacks described in this chapter.

This chapter contains joint work with Mike Scott, which was published at the International Conference on Pairing Based Cryptography - Pairing 2007 [137]. This conference was held in Tokyo, Japan in July 2007. Sections 6.3, 6.4 and 6.5 are part of the contribution of this thesis.

6.2 Related Work

Chapter 5 recounts the passive side channel attack aspect of the work of Page and Vercauteren [107]. In this section, the active fault attack contribution of the paper is examined.

Page and Vercauteren [107] describe a fault attack on the Duursma-Lee algorithm for computing the Tate pairing on supersingular elliptic curves in characteristic three and the Kwon-BGOS algorithm for the η pairing on supersingular elliptic curves in characteristic two. The type of fault considered causes the Miller loop to run over such that the Miller loop bound r is replaced by δ using a fault attack. The attack on Duursma-Lee algorithm (given in Algorithm 16) will be briefly reviewed here.

Algorithm 16 Duursma-Lee Algorithm for $E(\mathbb{F}_q) : y^2 = x^3 - x + b$ where $b = \pm 1$ and $q = 3^m$ [107].

INPUT: $P = (x_1, y_1), Q = (x_2, y_2) \in E(\mathbb{F}_{q^6})$
OUTPUT: $m \in \mathbb{F}_{q^6}^* \mid m^r = 1$
1: $m \leftarrow 1$
2: **for** $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ **to** 0 **do**
3: $\quad x_1 \leftarrow x_1^3$
4: $\quad y_1 \leftarrow y_1^3$
5: $\quad \mu \leftarrow x_1 + x_2 + b$
6: $\quad \lambda \leftarrow -y_1 \cdot y_2\sigma - \mu^2$
7: $\quad g \leftarrow \lambda - \mu\rho - \rho^2$
8: $\quad m \leftarrow m \cdot g$
9: $\quad x_2 \leftarrow x_2^{1/3}$
10: $\quad y_2 \leftarrow y_2^{1/3}$
11: **end for**
12: **return** m^{q^3-1}

The implementation assessed considers supersingular elliptic curves over the field $\mathbb{F}_q$ with $q = 3^m$, $k = 6$ and with a final exponent of $q^3 - 1$. Using the notation of Page and Vercauteren, elements in the sextic extension field $\mathbb{F}_{q^6}$ will be represented as

$$(a_5\rho^2 + a_4\rho + a_3)\sigma + (a_2\rho^2 + a_1\rho + a_0) \quad \text{where} \quad a_i \in \mathbb{F}_q$$

where $\mathbb{F}_{q^3} = F_q[\rho]/(\rho^3 - \rho - b)$ and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 + 1)$. The field is constructed as a tower of extensions, in that a quadratic is build on top of a cubic. This is a more efficient alternative for constructing extension fields as opposed to a straight sextic [35]. For the remainder of this section, elements in $\mathbb{F}_{q^3}$ will be represented as

$$\alpha = \langle a_2 \;\; a_1 \;\; a_0 \rangle = a_2\rho^2 + a_1\rho + a_0,$$

where $a_i \in \mathbb{F}_q$ and elements in $\mathbb{F}_{q^6}$ will be represented as

$$[\alpha_1 \;\; \alpha_0] = \alpha_1\sigma + \alpha_0,$$

where $\alpha_i \in F_{q^3}$.

The heart of the Miller loop is the calculation of the Miller variable m. In each round the value g is multiplicatively incorporated into the Miller variable m as follows

$$m \leftarrow m \cdot g$$

where

$$g = -y_1 \cdot y_2\sigma - (x_1 + x_2 + b)^2 - (x_1 + x_2 + b)\rho - \rho^2.$$

Using the notation defined above g has the form

$$g = [\langle 0 \;\; 0 \;\; -y_1 \cdot y_2 \rangle \langle -1 \;\; -(x_1 + x_2 + b) \;\; -(x_1 + x_2 + b)^2 \;\; \rangle]. \tag{6.1}$$

If the fault causes the Miller loop to run over, then additional evaluations of g will be performed. If the Miller loop executes just a single extra iteration, then dividing a pairing $e_{r+1}(P,Q)$ where the Miller loop has executed $r+1$ times, by a pairing $e_r(P,Q)$ where the Miller loop has executed the required number of times r, will yield,

$$\frac{e_{r+1}(P,Q)}{e_r(P,Q)} = (-y_1^{3^{r+1}} \cdot y_2\sigma - (x_1^{3^{r+1}} + x_2 + b)^2 - (x_1^{3^{r+1}} + x_2 + b)\rho - \rho^2)^{q^3-1}$$

and so

$$\frac{e_{r+1}(P,Q)}{e_r(P,Q)} = [\langle 0 \;\; 0 \;\; -y_1^{3^{r+1}} \cdot y_2\rangle\langle -1 \;\; -(x_1^{3^{r+1}} + x_2 + b) \;\; -(x_1^{3^{r+1}} + x_2 + b)^2\rangle]^{q^3-1}.$$

This corresponds to the factor produced from the extra round of the Miller loop induced by the fault. If this factor was not subsequently raised to the power of q^3-1, then the secret could be easily extracted. For example, if the secret was related to the coordinates (x_2, y_2), then by simply calculating $x_2 = -(x_1^{3^{r+1}} + b)$, one of the secret coordinates could be extracted. To access this simple factor g, and solve for the secret elliptic curve point, the final exponentiation has to be reversed however.

The technique that Page and Vercauteren use to reverse this exponentiation will be described here, although a number of approaches could be taken. Elements in $\mathbb{F}_{q^6}$ with order q^3-1 are all in $\mathbb{F}_{q^3}$, or of the form $\kappa_i = [0 \;\; \alpha^i]$ for $0 \leq i < q^3$. Let $R = g^{q^3-1}$. If a root μ of R is found such that $R = \mu^{q^3-1}$, then all roots have the form $\mu\kappa_i = [\mu_1\alpha^i \;\; \mu_0\alpha^i]$. Therefore, there must exist a particular κ that when multiplied by the root μ is equal to the root of interest g. The method used in [107] to find this root μ is known as the q-polynomial method, and is described by Lidl and Niederreiter [74]. This root can be manipulated so that it is easier to work with. Specifically, the authors of [107] normalise μ. Let $\mu = [\mu_1 \;\; \mu_0]$ and so $\mu = [\langle m_5 \;\; m_4 \;\; m_3\rangle\langle m_2 \;\; m_1 \;\; m_0\rangle]$, the normalisation of the

root μ, denoted $\bar{\mu}$, is calculated as

$$\begin{aligned}
\bar{\mu} &= [\mu_1 \ \mu_0][0 \ \mu_0^{-1}] \\
&= [\mu_0^{-1}\mu_1 \ \mu_0\mu_0^{-1}] \\
&= [\mu_0^{-1}\mu_1 \ 1] \\
&= [\langle m_2 \ m_1 \ m_0\rangle\langle 0 \ 0 \ 1\rangle].
\end{aligned}$$

Let $\kappa = [\langle 0 \ 0 \ 0\rangle\langle k_2 \ k_1 \ k_0\rangle]$. Now, the multiplication of κ by $\bar{\mu}$, will yield

$$\begin{aligned}
\kappa\bar{\mu} &= [\langle 0 \ 0 \ 0\rangle\langle k_2 \ k_1 \ k_0\rangle][\langle m_2 \ m_1 \ m_0\rangle\langle 0 \ 0 \ 1\rangle] \\
&= [\langle \bar{m}_2 \ \bar{m}_1 \ \bar{m}_0\rangle\langle k_2 \ k_1 \ k_0\rangle]
\end{aligned}$$

where

$$\bar{m}_2 = m_2k_2 + m_2k_0 + m_1k_1 + m_0k_2 \tag{6.2}$$

$$\bar{m}_1 = m_2k_2b + m_2k_1 + m_1k_2 + m_1k_0 + m_0k_1 \tag{6.3}$$

$$\bar{m}_0 = m_2k_1b + m_1k_2b + m_0k_0 \tag{6.4}$$

Since it is known that there must exist some value $\kappa \in \mathbb{F}_{q^3}$ which when multiplied by the normalised root $\bar{\mu}$ will give the correct root g, i.e. $g = \bar{\mu} \cdot \kappa$, these equations along with the information known about g can be used to solve for g. From Equation (6.1), it can assumed that $k_2 = -1 = 2$ and $\bar{m}_1 = \bar{m}_2 = 0$, allowing Equation (6.2) and (6.3) to be simplified to

$$m_2 + m_0 = m_2k_0 + m_1k_1 \tag{6.5}$$

$$m_2b + m_1 = m_2k_1 + m_1k_0 + m_0k_1 \tag{6.6}$$

Therefore, given just one root of R, by normalising that root the corresponding values for

m_0, m_1 and m_2 can be substituted into Equation (6.5) and (6.6) to solve for κ. Multiplying κ by $\bar{\mu}$ will reveal the root of interest, g.

The ability to reverse the final exponentiation depends on a special feature of the root of interest. The root g is not a full element in the sextic extension field, i.e. some of the coefficients are zero,

$$g = [\langle 0 \;\; 0 \;\; -y_1 \cdot y_2 \rangle \langle -1 \;\; -(x_1 + x_2 + b) \;\; -(x_1 + x_2 + b)^2 \rangle].$$

thus providing additional information which aids the adversary to reverse the final exponentiation.

Note that this attack depends on the fault causing the Miller loop to execute exactly one extra iteration. If however, a random number of iterations is induced, which is a more realistic scenario, then a number of faulty pairings need to be computed until the pair $(e_{r+z}(P,Q), e_{r+z+1}(P,Q))$ are identified, for z random. Identifying these pairs is assumed to be possible, since SPA can be used on the power trace to identify the number of rounds that have been executed.

6.3 Overview of Attack

In this section, the general idea behind fault analysis of bilinear pairings is presented. First, the type of fault that is considered and the reasoning behind the choice of candidate pairings is given.

6.3.1 Fault Type

Fault attacks will target memory locations or registers used for computation. Therefore, the way in which data values in the bilinear pairing are stored and operated on, will influence the focus of a fault attack. Fault attacks can be categorised as having three main effects on an algorithm. The first effect seeks to knock out a step in the computation, i.e.

a no-op replaces another working instruction. This allows selective execution of instructions in an algorithm. The second effect seeks to cause a loop either to end prematurely or to run over. The third effect seeks to cause the data being operated on to be corrupted in some way. Page and Vercauteren deal with the second type of fault. In this chapter, the effects of the third type of fault, a data corruption fault, on a number of candidate bilinear pairings are examined.

A data corruption fault is defined as the modification of a data variable in some way. For example, a field element $a \in \mathbb{F}_{2^{4m}}$ will be represented as $a_0 + a_1 z + a_2 z^2 + a_3 z^3$ with $a_i \in \mathbb{F}_{2^m}$ and so will be stored in four different memory locations. The notation $[a_0][a_1][a_2][a_3]$ represents the storage of these elements. Each component of this representation is referred to as a cell. A fault can target any of the cells in memory, $[a_0][a_1][a_2]$ or $[a_3]$. For example, a data corruption fault may target the value $[a_3]$, producing an erroneous value $[a_3]'$. The fault can alter either a bit, bits, byte or bytes of $[a_3]$ to produce $[a_3]'$. A corrupt bilinear pairing computation, will be called a faulty pairing. Given a faulty pairing, the ability to derive the secret depends on the effect that the fault injected has caused and the bilinear pairing algorithm itself. In the three candidate bilinear pairings, the consequences of such faults are examined to determine whether the fault injected enables extraction of the secret. Details on fault injection techniques, fault types and effects, were presented in Chapter 2.

6.3.2 Choice of Candidate Pairings

Three bilinear pairing algorithms are selected for examination for vulnerability to a data corruption fault attack. The three pairings chosen are namely the Weil pairing, the η pairing and the Tate pairing. Each of these is chosen based on the varying complexity of its final exponent. The version of the η pairing considered is the version of Galbraith et al. [41], who recently developed a version of the η pairing with no final exponentiation. This version of the η pairing considered here will be referred to as the η_G pairing to

distinguish it from the original η pairing, which has a final exponentiation. The version of Weil pairing considered is also a variant of the original and most widely recognised Weil pairing. The version of the Weil pairing considered employs a simple final exponent of $p-1$, unlike the original which has no final exponentiation [117]. To distinguish the version of the Weil pairing considered here from the original Weil pairing, it will be denoted by ω_D since the version considered employs denominator elimination. The version of the Tate pairing considered is the original Tate pairing computed using the BKLS algorithm. It has the most complex final exponent of all the bilinear pairings considered of $(p^k-1)/r$. The algorithms for each of these bilinear pairings are given in Chapter 3.

As was noted in Chapter 5, the secret parameter can be entered as the first or second parameter in the bilinear pairing $e(P,Q)$. Therefore, there are two potential avenues to attack the bilinear pairing, the P path and the Q path. The situation where the secret elliptic curve point can take either path is addressed in each attack description.

6.3.3 The General Idea

Bilinear pairings consist of two main parts, the Miller loop and the final exponentiation. Therefore, either part of the algorithm can be targeted. To attack the Miller loop, there are a number of locations that a data corruption fault can target. It can affect the Miller variable, the point P (or an intermediate point calculated during the computation of $[r]P$), the point Q (or an intermediate point calculated during computation of $[r]Q$, specifically in the case of the Weil and η_G pairing), or the order of the Miller loop. An attack that alters the order of the Miller loop was examined by [107]. All other possibilities for attack will be discussed in this chapter. To attack the final exponentiation, two data values can be targeted with a data corruption fault, either the Miller variable resulting from the Miller loop or the exponent can be affected. However, the consequences of tampering with either component in the final exponentiation is difficult to exploit, as will be discussed below.

Let $\rho(P,Q)$ denote any of the candidate bilinear pairing algorithms, i.e. $\rho(P,Q) = \eta_G(P,Q)$, $\omega_D(P,Q)$ or $e(P,Q)$ where $\eta_G(P,Q)$ denotes Galbraith et al.'s η pairing, $\omega_D(P,Q)$ denotes the Weil pairing of [117] and $e(P,Q)$ denotes the Tate pairing. Let $\rho(P,Q)'$ denote a pairing algorithm in which a fault has been injected. The type of fault that produces $\rho(P,Q)'$ is discussed in the respective attack descriptions.

Adopting the approach of Page and Vercauteren and considering the Miller loop alone, the pairing $\rho(P,Q)$ can be represented as the product

$$m = \prod_{i=1}^{r} g_i^{2^{r-i}} \tag{6.7}$$

where g_i accounts for all line function contributions. If a fault is injected into one of the line functions, then the following relationship can be exploited,

$$\frac{\rho(P,Q)}{\rho(P,Q)'} = \frac{m}{m'} = \left(\frac{g_i}{g_i'}\right)^{2^{r-i}} \tag{6.8}$$

where g_i is the correct line contribution to the i-th iteration, g_i' is the corrupted line contribution to the i-th iteration, i is the round in the Miller loop where the fault was injected and r denotes the number of rounds in the Miller loop. For example, if a correct pairing[1] is calculated as

$$(((((g_1)^2 \cdot g_2)^2 \cdot g_3)^2 \ldots)^2 \cdot g_r) \tag{6.9}$$

and a faulty pairing is calculated as

$$(((((g_1)^2 \cdot g_2)^2 \cdot g_3')^2 \ldots)^2 \cdot g_r) \tag{6.10}$$

where the fault corrupts the line contribution in the third round of the algorithm, then dividing a correct pairing by a faulty pairing, where input parameters are identical for both

[1] When the pairing involves double and additions, this may also be represented as $(((g_1)^2 \cdot g_2 \cdot a_2)^2 \ldots)^2 \cdot g_r)$ where in some rounds, depending on the binary representation of the loop order, both an addition and double is performed.

pairings, will nullify any parts of the pairing computation common to both executions, and leave the parts of the pairing computations where the two executions diverge, i.e.

$$\frac{\rho(P,Q)}{\rho(P,Q)'} = \left(\frac{g_3}{g_3'}\right)^{2^{r-3}} \tag{6.11}$$

which are the remaining parts of the computation affected by the fault. An obvious objective of the fault attack will be to inject a fault so that the difference between these two pairings $\rho(P,Q)$ and $\rho(P,Q)'$ is minimal, and so dividing $\rho(P,Q)$ by $\rho(P,Q)'$ isolates a part in the computation which is easy to exploit. Injecting a type of fault so that an exploitable part of the computation is isolated, is the key idea behind the attacks proposed in this chapter. Some of the optimal locations for fault injection include the following.

1. The fault should target the last round, or at least later rounds, in the Miller loop. The optimal round to attack is the final round.

$$\frac{\rho(P,Q)}{\rho(P,Q)'} = \left(\frac{g_i}{g_i'}\right)^{2^{r-r}} = \left(\frac{g_i}{g_i'}\right). \tag{6.12}$$

   Any rounds that are targeted before the final round will require calculation of a number of square roots to access this fraction.

2. The fault should target values used in the Miller loop which are required for use in a single round alone. This is particularly relevant for faults which affect rounds preceding the final round. If the fault affects values which are used in multiple rounds, then multiple contributions to the Miller variable will be affected. However, if the fault targets values used in the Miller loop which are only required for use in a single round then the fault can be contained, only affecting the Miller variable for that round. Places for possible injection are the coordinates x_i, y_i or the slope λ_i, which are the intermediate values in the computation of $[r]P$ (or $[r]Q$). If these values are pre-computed and stored on the device and looked up when required

[120], then the fault injected can be contained. However, if the fault affects x_i, y_i or λ_i, where they are computed as part of the pairing, then the subsequent x_{i+1}, y_{i+1} or λ_{i+1} will also be corrupted, leading to contamination of numerous parts of the computation, thus denying access to the single factor.

Generally, as will be described below, once the fault targets these optimal locations a single factor (and by single factor it is meant a single Miller variable evaluation) from the Miller loop can be accessed. Accessing this single factor can greatly facilitate extraction of the secret, as was the case for [107]. However, in certain bilinear pairing algorithms extracting the secret is not this straightforward, since the output of the Miller loop undergoes another step in the pairing computation, the final exponentiation.

If the bilinear pairing requires a final exponentiation, then relationship in Equation (6.8) becomes

$$\frac{\rho(P,Q)}{\rho(P,Q)'} = \left(\frac{g_i}{g_i'}\right)^{2^{(r-i)}f} \tag{6.13}$$

where f is the final exponent. To reclaim the straightforward relationship in (6.8), reversal of the final exponentiation is necessary. However, depending on f this may not be possible.

The complexity of the final exponentiation depends on the bilinear pairing. In the situation where a simple final exponentiation exists, as is the case for the variant of the Weil pairing considered here, access to the output of the Miller loop is in fact possible. This is due to the Frobenius action. For example, let $a+ib$ be an element in the field $\mathbb{F}_{p^2}$, with $a, b \in \mathbb{F}_p$, the Frobenius action is calculated as

$$(a+ib)^{p-1} = \frac{(a-ib)}{(a+ib)} = c+id,$$

thus providing a straightforward relationship between $c+id$ and $a+ib$. This will be demonstrated further in Section 6.4.2. In the situation where a more complex final exponentiation exists, as in the case for the Tate pairing, the output of the Miller loop is

difficult to access and accessing it is equivalent to solving a n-th root problem [54]. What makes the final exponentiation difficult to reverse is that it is a many-to-one relationship [41]. In general, the final exponentiation involves raising the output of the Miller loop to the power of $(q^k - 1)/r$. This can be thought of as

$$(q^d - 1)\left(\frac{(q^d + 1)}{r}\right) \quad \text{or} \quad (q^d - 1)\left(\frac{(q^d + 1)}{\Phi_k(q)}\right)\left(\frac{\Phi_k(q)}{r}\right)$$

where $\Phi_k(q)$ is the cyclotomic polynomial and $k = 2d$. Raising to the power of $(q^d - 1)$ and $((q^d + 1)/\Phi_k(q))$ is easy using the Frobenius action. Raising to the power of $(\Phi_k(q)/r)$ is not as straightforward, requiring usage of an algorithm for fast exponentiation [91, 58, 59].

Page and Vercauteren describe an attack of the Duursma-Lee algorithm for the Tate pairing, which has a final exponent of $q^3 - 1$. In their attack they are able to reverse this final exponentiation and access the single factor from the Miller loop. This is because the single factor that they are accessing has a special form and so is identifiable from all other roots. In the next section, it will be shown that the successful attacks on the η_G and Weil pairing, cannot be applied to the Tate pairing. This is because the factor that the Tate pairing produces from the Miller loop is not in general of special form and so cannot be pinpointed from all other roots.

As mentioned above, the final exponentiation can itself also be the target of a fault attack. However, considering even the most powerful scenario where a fault nullifies the final exponentiation, this means that the output of the bilinear pairing algorithm will be the output of the Miller loop. The output of the Miller loop will consist of all contributions to the Miller variable, which have been multiplicatively accumulated. These can be thought of as a system of multivariate equations similar to those described in [41], which [41] notes is difficult to solve. Therefore corrupting the final exponentiation, will not aid the adversary in finding the secret. If the adversary can inject multiple faults such that

the final exponentiation is nullified in one execution, and in a second execution the final exponentiation is nullified and a fault is injected into the Miller loop to isolate a single factor, then an attack could be launched. This however, is an unrealistic attack scenario.

6.4 Specific Examples of Attack

In this section, concrete instances of a data corruption fault attack on the η_G and Weil pairing are presented. It is then shown how such attacks are not applicable to the Tate pairing. Firstly, a number of assumptions are made about the fault analysis of the candidate pairings.

1. It is first assumed that the adversary has the capability to inject a fault, i.e. the adversary has access to a glitch attack environment or a similar attack apparatus.

2. It is assumed that the adversary will have knowledge of the point in time in which to inject the fault. This is based on the fact that the adversary can use Simple Power Analysis (SPA) to identify each round in the Miller loop via the power trace.

3. It is assumed that the adversary can invoke multiple pairing executions for their choice of elliptic curve point. For example, $\rho(P_i, Q)$ can be executed for various input points P_i or $\rho(P, Q_i)$ can be executed for various input points Q_i, where the secret of interest is in the former case the point Q and in the latter the point P. Note that the adversary also has the power to repeatedly input the same elliptic curve point to the pairing.

4. In most cases it is assumed that the fault is injected into the last round of the Miller loop, for the reasons given above.

6.4.1 Corrupting the η Pairing

The η pairing $\eta(P,Q)$ [13], specialises in pairings over supersingular curves of small characteristic. The main distinction between the η pairing and its siblings is that it chooses the order of the Miller loop r as a multiple of the group order such that it divides $q^k - 1$ to give a small factor, resulting in a simple final exponentiation. For example, consider the η pairing on a supersingular curve of characteristic two, if $q^k - 1 = 2^{4m} - 1$ and $r = 2^{2m} + 1$, which is a multiple of the order $2^m \pm 2^{(m+1)/2} + 1$, then the final exponentiation basically involves a conjugation and division, i.e. $(2^{2m} - 1)$.

Recently Galbraith et al. [41] described a variant of the η pairing, referred to as the η_G pairing, which required no final exponentiation, i.e. the result of the pairing is a unique element and a bilinear map without any final exponentiation. This is enabled by the additional evaluation of vertical line functions (which the original η pairing [13] did not require).

When the authors of [41] presented the η_G pairing with no final exponentiation, they addressed possible security implications. Mathematical attacks such as a multivariate attack and a straight line program (SLP) were considered. However, they conclude that the pairing's security (in the non fault attack sense) is still strong, and breaking such a pairing requires solving the pairing inversion problem, which both [130] and [114] show is difficult. In this section, it is shown how pairings with no final exponentiation can easily succumb to a data corruption fault attack.

Algorithm 7 and 8 given in Chapter 3, describe the η_G pairing. The implementation described considers supersingular elliptic curves over the binary field $\mathbb{F}_{2^m}$ with $k = 4$. The heart of the Miller loop is the calculation of the Miller variable m. In each round the line functions l and v are calculated and then divided to produce g, which is multiplicatively incorporated into the Miller variable m as follows

$$m \leftarrow (m)^2 \cdot g.$$

The lines l and v to produce g, are calculated as

$$l = [y_Q + y_A + \lambda(x_Q + x_A + 1)][\lambda + x_Q + 1][\lambda + x_Q][0]$$

and

$$v = [x_Q + x_C + 1][1][1][0].$$

The output of the pairing is an element in the extension field $\mathbb{F}_{24m}$. The cells of l and v will be denoted by $[l_0]$, $[l_1]$, $[l_2]$, $[l_3]$ and $[v_0]$, $[v_1]$, $[v_2]$, $[v_3]$ respectively.

There are a number of possible locations in which the fault can be injected, and a number of different effects that this fault can have. A fault can be injected into any of the cells of l or v, or any of the coordinates x_A, y_A, x_C, x_Q or y_Q and the fault injected can corrupt a bit or byte or multiple bits or bytes of the target. If a fault is injected into any of the cells of l or v, then the effect will be local, only corrupting the cell in question. If the fault is injected into one of the coordinates, then the resulting erroneous coordinate will have consequences for all subsequent operations in which the erroneous coordinate is used. The possible effects of the faults in these locations, will each be addressed in turn.

Let $\eta_G(P,Q)'$ denote a corrupted pairing, where the fault is injected into the cell l_0 in the last round of the Miller loop. Division of the faulty pairing by the valid pairing, where input elliptic curve points are identical for both executions, will isolate the round in which the fault was injected to yield

$$\frac{\eta_G(P,Q)'}{\eta_G(P,Q)} = \frac{m^{2^r} \cdot g'}{m^{2^r} \cdot g} = \frac{g'}{g} = \frac{(l'/v)}{(l/v)} = \frac{l'}{l} = \frac{[l_0]'[l_1][l_2][l_3]}{[l_0][l_1][l_2][l_3]}.$$

This division will produce an element in $\mathbb{F}_{24m}$, and can be thought of as four different cell

values N_0, N_1, N_2 and N_3, where $N_i \in \mathbb{F}_{2^m}$. Therefore,

$$\frac{[l_0]'[l_1][l_2][l_3]}{[l_0][l_1][l_2][l_3]} = [N_0][N_1][N_2][N_3], \tag{6.14}$$

which can be rewritten as

$$[l_0]'[l_1][l_2][l_3] = ([N_0][N_1][N_2][N_3])([l_0][l_1][l_2][l_3]). \tag{6.15}$$

Given $\eta_G(P,Q)$ and $\eta_G(P,Q)'$, the adversary can compute N_0, N_1, N_2 and N_3.

Using knowledge of how multiplication in $\mathbb{F}_{2^{4m}}$ is performed, it can be determined which cells on the right-hand-side of Equation (6.15) correspond to the cell on the left-hand-side of the equation. For instance, the following three equations can be derived.

$$l_0' = N_0 l_0 + N_1 l_1 + N_2 l_2 + (N_1 + N_3) l_1 \tag{6.16}$$

$$l_1 = N_0 l_0 + 2N_1 l_1 + 2N_2 l_2 + (N_0 + N_1)(l_0 + l_1) + (N_1 + N_3) l_1 + (N_2 + N_3) l_2 \tag{6.17}$$

$$l_2 = N_0 l_0 + N_1 l_1 + 2N_2 l_2 + (N_2 + N_3) l_2 + (N_0 + N_2)(l_0 + l_2) \tag{6.18}$$

The adversary will obviously choose the optimal equation to solve. Equation (6.16), contains l_0', whereas Equations (6.17) and (6.18) do not. Hence either Equations (6.17) or (6.18) contain less unknown information.

In the scenario where P is private, (6.18) can be simplified to

$$A_0 x_A + B_0 y_A + C_0 \lambda + D_0 = 0 \tag{6.19}$$

where

$$\begin{aligned} A_0 &= N_2\lambda \\ B_0 &= N_2 \\ C_0 &= N_0 + N_1 + N_2 + N_3 + N_2x_Q + 1 \\ D_0 &= (N_0 + N_1 + N_3 + 1)x_Q + N_2y_Q + N_1 \end{aligned}$$

In the scenario where Q is private, (6.18) can be simplified to

$$A_0x_Q + B_0y_Q + C_0 = 0 \tag{6.20}$$

where

$$\begin{aligned} A_0 &= N_0 + N_1 + N_2\lambda + N_3 + 1 \\ B_0 &= N_2 \\ C_0 &= N_0\lambda + N_1 + N_1\lambda + N_2(x_A\lambda + y_A + \lambda) + N_3\lambda + \lambda \end{aligned}$$

Note that Equation (6.17) could just as validly have been used. When P is secret, notice that Equation (6.19) is in fact a non-linear equation. However, when Q is secret a linear equation is obtained. To solve for Q, Equation (6.20) has two unknown variables and so two equations are required to solve the system. First however, the required equations must be produced. The main requirement is that the equations are produced with the same unknown variables. This can be achieved with two approaches. Either this can be performed by repeatedly executing the pairing with the objective of injecting various faults into l_0, or the pairing could be executed with other parameters, $\eta_G(P_i, Q)$ for various elliptic curve points P_i, where the secret Q remains static. For example, if the first approach is used

where two different faults are injected into l_0 to produce l_0' and l_0'',

$$\frac{\eta_G(P,Q)'}{\eta_G(P,Q)} = \frac{m^{2^r} \cdot g'}{m^{2^r} \cdot g'} = \frac{g'}{g} = \frac{(l'/v)}{(l/v)} = \frac{l'}{l} = \frac{[l_0]'[l_1][l_2][l_3]}{[l_0][l_1][l_2][l_3]} = [N_0][N_1][N_2][N_3]$$

$$\frac{\eta_G(P,Q)''}{\eta_G(P,Q)} = \frac{m^{2^r} \cdot g''}{m^{2^r} \cdot g''} = \frac{g''}{g} = \frac{(l''/v)}{(l/v)} = \frac{l''}{l} = \frac{[l_0]''[l_1][l_2][l_3]}{[l_0][l_1][l_2][l_3]} = [N_4][N_5][N_6][N_7]$$

then two entirely different sets of equations can be derived in the same variable. If the second approach is used, and the pairing is executed with other parameters, such as $\eta_G(P_i, Q)$ for various elliptic curve points P_i where the secret Q remains static, then the required number of equations in the same variable can be derived in a similar manner.

Given two such equations, modular Gaussian elimination [80] or simple substitution can be used. Alternatively, since x_Q and y_Q can be expressed in terms of each other using the elliptic curve equation, one fault alone will be sufficient to extract Q. A specific example of using the relationship between the coordinates x and y will be given in the next section. A numerical example of the attack described above is given in Appendix B.1.1.

If a fault is injected into any of the other cells of l, then a similar process can be applied to derive similar equations. Hence, fault injection into any of the cells of l, is equivalent to solving a system of modular linear equations.

In the scenario where the fault is injected during the calculation of the line function v, the η_G pairing succumbs to a more straightforward attack. Let $\eta_G(P,Q)'$ denote a corrupted pairing, where the fault is injected into the cell v_0 in the last round of the Miller loop. It is again assumed that the input elliptic curve points are identical for both executions. Division of the valid pairing by the faulty pairing will isolate the round in which the fault was injected to yield

$$\frac{\eta_G(P,Q)}{\eta_G(P,Q)'} = \frac{m^{2^r} \cdot g}{m^{2^r} \cdot g'} = \frac{g}{g'} = \frac{(l/v)}{(l/v')} = \frac{v'}{v} = \frac{[v_0]'[v_1][v_2][v_3]}{[v_0][v_1][v_2][v_3]}$$

This division will produce an element in $\mathbb{F}_{2^{4m}}$, and can be thought of as four different cell values N_0, N_1, N_2 and N_3, where $N_i \in \mathbb{F}_{2^m}$. Therefore,

$$\frac{[v_0]'[v_1][v_2][v_3]}{[v_0][v_1][v_2][v_3]} = [N_0][N_1][N_2][N_3], \tag{6.21}$$

which can be rewritten as

$$[v_0]'[v_1][v_2][v_3] = ([N_0][N_1][N_2][N_3])([v_0][v_1][v_2][v_3]). \tag{6.22}$$

Given $\eta_G(P,Q)$ and $\eta_G(P,Q)'$, the adversary can compute N_0, N_1, N_2 and N_3. Again, using knowledge of how multiplication in $\mathbb{F}_{2^{4m}}$ is performed, it can be determined which cells on the right-hand-side of Equation (6.22) correspond to the cell on the left-hand-side of the equation. For instance, the following three equations can be derived.

$$v_0' = N_0v_0 + N_1v_1 + N_2v_2 + (N_1 + N_3)v_1 \tag{6.23}$$

$$v_1 = N_0v_0 + 2N_1v_1 + 2N_2v_2 + (N_0 + N_1)(v_0 + v_1) + (N_1 + N_3)v_1 + (N_2 + N_3)v_2 \tag{6.24}$$

$$v_2 = N_0v_0 + N_1v_1 + 2N_2v_2 + (N_2 + N_3)v_2 + (N_0 + N_2)(v_0 + v_2) \tag{6.25}$$

Since $v_1 = 1$ and $v_2 = 1$, equations (6.24) and (6.25) can be simplified to

$$v_0 = \frac{N_0 + 4N_1 + 3N_2 + 2N_3 + 1}{2N_0 + N_1} \tag{6.26}$$

or

$$v_0 = \frac{N_0 + N_1 + 4N_2 + N_3 + 1}{2N_0 + N_2}, \tag{6.27}$$

where only one unknown piece of data remains, v_0. Since v_0 is equal to $[x_Q + x_C + 1]$, either the coordinate x_Q or x_C can be extracted depending on whether the secret is P or Q. A numerical example of this attack is given in Appendix B.1.2.

If the target for the fault attack is instead a coordinate used in the calculation of the line function, various consequences can be observed. The main difference between corruption of a coordinate and corruption of a cell as described above, is that a coordinate may be influential in a number of cells. The coordinate x_C is only resident in the cell v_0 and so if corrupted will have similar consequences to the corruption of v_0 as described above. Similarly, if the coordinates x_A, y_A or y_Q are corrupted, then this is equivalent to the corruption of the cell l_0.

If however, the fault attack corrupts either the coordinate x_Q or the slope λ, then the fault will affect numerous cells. If the fault affects λ, the cells l_0, l_1 and l_2 will also be affected, and result in the introduction of an extra unknown variable into the system of linear equations. This results in an increase in the number of unknowns to four variables for P secret and three for Q secret. In addition, this extra variable is very specific in that the solution of the system of equations requires the ability to recreate identical faults. For example, the same λ' must be created in at least three pairings in order to be able to extract Q. This is a difficult task to initiate and detect. If the fault affects x_Q, then the cells l_0, l_1, l_2 and v_0 will be corrupted. Therefore, the division of the valid and faulty pairing will not cancel one of the line functions, l or v, leaving a relationship of the form

$$\frac{\eta_G(P,Q)}{\eta_G(P,Q)'} = \frac{(l/v)}{(l'/v')} = \frac{\frac{[l_0][l_1][l_2][l_3]}{[v_0][v_1][v_2][v_3]}}{\frac{[l_0]'[l_1]'[l_2]'[l_3]}{[v_0]'[v_1][v_2][v_3]}} = [N_0][N_1][N_2][N_3].$$

When expanded, a difficult modular non-linear equation is obtained.

Another consequence of corrupting coordinates, is that the effect of the fault will not be local, as is the case for cell corruption. This is particularly problematic if the fault is injected in a round prior to the final round. For example, if the coordinates x_A or y_A are corrupted in a round preceding the final round, then the subsequent point addition in which x_A and y_A will be involved, will be affected. A common practice to optimise computation time is to pre-compute the intermediate points required in the computation of

$[r]P$ and store them on the device, so only a look-up operation is required as opposed to an elliptic curve point scalar multiplication [120]. In such an implementation, the fault can target the memory cell in which x_A or y_A is stored, and so the effect will remain local. In scenarios where these types of attacks are a threat, this would be a valid argument against the use of pre-calculation.

6.4.2 Corrupting the Weil Pairing

The Weil pairing $\omega(P, Q)$ over a prime characteristic elliptic curve with embedding degree $k = 2$, was originally defined with no final exponentiation. However by eliminating the vertical function evaluations (an optimisation known as denominator elimination), a Weil pairing $\omega_D(P, Q)$ with a simple final exponent can be more efficient [117].

Algorithms 2 and 3 given in Chapter 3, describe the Weil pairing. The particular implementation assessed [117] considers ordinary elliptic curves over the prime field $\mathbb{F}_p$ with $k = 2$ and a final exponent of $p - 1$. The heart of the Miller loop is the calculation of the Miller variable m. In each round the value g, which consists of the evaluation of the line functions u and v are calculated and then multiplied to produce g, which is multiplicatively incorporated into the Miller variable as either

$$m \leftarrow (m)^2 \cdot g$$

or

$$m \leftarrow (m)^2 \cdot g \quad \text{and} \quad m \leftarrow m \cdot g$$

depending on the binary representation of the order of the Miller loop and whether an addition or doubling is being performed. The lines u and v to produce g are calculated as

$$u = [\lambda_1(x_A + x_Q) - y_A][y_Q]$$

and

$$v = [y_P][y_C - \lambda_2(x_C + x_P)].$$

The output of the pairing is an element in $\mathbb{F}_{p^2}$. The cells of u and v will be denoted by $[u_0][u_1]$ and $[v_0][v_1]$ respectively.

Again, a number of different locations can be targeted and various types of faults can be injected to cause different effects. The types of faults which aid in the extraction of the secret are far fewer than in the η_G pairing. This is because direct access to the output of the Miller loop is not available.

Let $\omega_D(P_i, Q_i)'$ denote the Weil pairing in which a fault is injected into the last round in the Miller loop. The fault corrupts data in an unspecified way, i.e. either $[u_0]$, $[u_1]$, $[v_0]$ or $[v_1]$ is corrupted. The case where $[u_0]$ is corrupted, will be described for demonstrative purposes, however the same end result is witnessed for this type of fault regardless of the cell targeted. Division of the valid pairing by the fault pairing will yield

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{(m^{2^r} \cdot g)^{p-1}}{(m^{2^r} \cdot g')^{p-1}} = \frac{(u \cdot v)^{p-1}}{(u' \cdot v)^{p-1}} = \frac{([u_0][u_1])^{p-1}}{([u_0]'[u_1])^{p-1}}.$$

This type of disruption of the execution has the following consequences, where raising an element in $\mathbb{F}_{p^2}$ to the power of $p-1$ involves a conjugation and division.

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{\frac{([\lambda_i(x_A + x_Q) - y_A][-y_Q])}{([\lambda_i(x_A + x_Q) - y_A][y_Q])}}{\frac{([\lambda_i(x_A + x_Q) - y_A]'[-y_Q])}{([\lambda_i(x_A + x_Q) - y_A]'[y_Q])}}$$

which is equivalent to

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \left(\frac{([\lambda_i(x_A + x_Q) - y_A][-y_Q])}{([\lambda_i(x_A + x_Q) - y_A][y_Q])} \cdot \frac{([\lambda_i(x_A + x_Q) - y_A]'[y_Q])}{([\lambda_i(x_A + x_Q) - y_A]'[-y_Q])}\right).$$

If this equation is expanded, a difficult multivariate non-linear equation is derived. The cancelations that were possible on the η_G pairing, which allow access to a simple factor,

are no longer possible. This is due to the final exponentiation.

Since a general data corruption fault of the Weil pairing will not suffice to extract the secret, other fault types are examined. Another more powerful and targeted attack that will facilitate extraction of the secret elliptic curve point, targets the sign of either $[u_1]$ or $[v_0]$ and so the coordinate y_Q or y_P. This type of fault attack is referred to as a sign change fault attack since a single sign bit is flipped [20], assuming that the sign is stored as a single bit.

Whether P or Q is the secret affects the optimal location to alter the sign (or flip the bit) and how the secret is extracted. When Q is secret the optimal location to inject the fault is the sign of the y_Q coordinate, whereas when P is secret the optimal location is the sign of the y_P coordinate. This will be demonstrated below.

Let the fault attack cause a change in sign in the y_Q component of the elliptic curve point Q during the last round of the Miller loop, i.e. u is altered as follows

$$u' = [\lambda_i(x_A + x_Q) - y_A][-y_Q].$$

Again it is assumed that one valid pairing $\omega_D(P,Q)$ and one faulty pairing $\omega_D(P,Q)'$, can be calculated.

Since the implementation of the Weil pairing being assessed requires a final exponentiation, division of the valid pairing by a faulty pairing will yield the following:

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{(m^{2^r} \cdot g)^{p-1}}{(m^{2^r} \cdot g')^{p-1}} = \frac{(u \cdot v)^{p-1}}{(u' \cdot v)^{p-1}} = \frac{(u)^{p-1}}{(u')^{p-1}} = \frac{([\lambda_i(x_A + x_Q) - y_A][y_Q])^{p-1}}{([\lambda_i(x_A + x_Q) - y_A][-y_Q])^{p-1}}$$

Raising an element in $\mathbb{F}_{p^2}$ to the power of $p-1$ is simply a conjugation and division,

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{\frac{([\lambda_i(x_A + x_Q) - y_A][-y_Q])}{([\lambda_i(x_A + x_Q) - y_A][y_Q])}}{\frac{([\lambda_i(x_A + x_Q) - y_A][y_Q])}{([\lambda_i(x_A + x_Q) - y_A][-y_Q])}}$$

which is equivalent to

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \left(\frac{[\lambda_i(x_A + x_Q) - y_A][-y_Q]}{[\lambda_i(x_A + x_Q) - y_A][y_Q]}\right)^2.$$

Therefore,

$$\sqrt{\frac{\omega(P,Q)}{\omega(P,Q)'}} = \pm\left(\frac{[\lambda_i(x_A + x_Q) - y_A][-y_Q]}{[\lambda_i(x_A + x_Q) - y_A][y_Q]}\right) \tag{6.28}$$

This division will produce an element in $\mathbb{F}_{p^2}$, and can be thought of as two different cell values N_R and N_C where $N_i \in \mathbb{F}_p$. Therefore,

$$\pm\left(\frac{[\lambda_i(x_A + x_Q) - y_A][-y_Q]}{[\lambda_i(x_A + x_Q) - y_A][y_Q]}\right) = [N_R][N_C].$$

Exploiting this relation, two linear equations can be derived,

$$(\lambda_i - N_R\lambda_i)x_Q + N_C y_Q + (x_A\lambda_i - y_A - N_R x_A\lambda_i + N_R y_A) = 0 \tag{6.29}$$

$$(N_C\lambda_i)x_Q + (N_R + 1)y_Q + (N_C x_A\lambda_i - N_C y_A) = 0 \tag{6.30}$$

Since there exists two possible square roots, there are two possibilities for each equation.

If P is secret, then three unknown values x_A, y_A and λ_i are present in the equation, and so will require three faults to be injected to solve for P.

If Q is the secret however, two unknown values x_Q and y_Q are present in the equation. With the use of the elliptic curve equation E, only one fault needs to be injected to derive Q. Using the elliptic curve equation E, $\pm\sqrt{x_Q^3 + ax_Q - b}$ can be substituted for y_Q in either of the above equations. This will yield a cubic equation in one variable, which is solvable by Cardano's method [25]. For example Equation (6.29) reduces to

$$x_Q^3 + A_0 x_Q^2 + B_0 x_Q + C_0 = 0 \tag{6.31}$$

where

$$\begin{aligned} A_0 &= -\frac{(\lambda_i - N_R\lambda_i)^2}{N_C^2} \\ B_0 &= \frac{(a - 2(\lambda_i - N_R\lambda_i)(x_A\lambda_i - y_A - N_R x\lambda_i + N_R y_A))}{N_C^2} \\ C_0 &= -b - \frac{(x_A\lambda_i - y_A - N_R x_A\lambda_i + N_R y_A)^2}{N_C^2} \end{aligned}$$

and Equation (6.30) reduces to

$$x_Q^3 + A_0 x_Q^2 + B_0 x_Q + C_0 = 0 \tag{6.32}$$

where

$$\begin{aligned} A_0 &= -\frac{(N_C x_A\lambda_i)^2}{(N_R+1)^2} \\ B_0 &= \frac{(a + 2(N_C x_A\lambda_i)(N_C x_A\lambda_i - N_C y_A))}{(N_R+1)^2} \\ C_0 &= -b + \frac{(N_C x_A\lambda_i)^2}{(N_R+1)^2} \end{aligned}$$

Given x_Q, y_Q can then be found by simply calculating $\pm\sqrt{x_Q^3 + ax_Q - b}$.

When P is secret the optimal location is the sign of the y_P coordinate in the line function v, which will similarly see P being extracted with only one fault. A numerical example of this sign change fault attack is given in Appendix B.2.

As previously explained, the optimal time in which to inject the fault is the final round of the Miller loop. If the fault is injected in earlier rounds in the Miller loop, then the effort to extract the secret is increased. Any round preceding the final round sees the Miller variable m being squared. Therefore, to access equations similar to Equation (6.29) or Equation (6.30), multiple square roots calculations will be required. For example, let the fault corrupt the sign of y_Q in the second last round of the Miller loop. The relationship

between the output of the pairings and the data of interest is now

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{((m^{2^{r-1}} \cdot g)^2)^{p-1}}{((m^{2^{r-1}} \cdot g')^2)^{p-1}} = \left(\frac{(u)^2}{(u')^2}\right)^{p-1}$$

and so

$$\sqrt{\sqrt{\frac{\omega_D(P,Q)}{\omega_D(P,Q)'}}} = \pm\left(\frac{[\lambda_i(x_A + x_Q) - y_A][-y_Q]}{[\lambda_i(x_A + x_Q) - y_A][y_Q]}\right).$$

This requires the computation of two square roots and so potentially four cubic equations (depending on whether the square root exists or not). Hence, the earlier in the loop the fault is injected the greater number of cubic equations there will be to solve and test. It is expected that this problem can be combatted by using SPA to identify the target loop.

6.4.3 The Resistance of Tate Pairing to a Data Corruption Fault

The Tate pairing $e(P,Q)$, has the most complex final exponent of $(q^k - 1)/r$. This not only ensures that the output of the pairing is unique, but it also serves as a defensive mechanism against fault attacks as will now be demonstrated.

Algorithm 4 and 5 given in Chapter 3, describe the BKLS algorithm for the Tate pairing. The implementation assessed considers non-supersingular elliptic curves over the prime field $\mathbb{F}_p$ with $k = 2$ and a final exponent of $(p^2 - 1)/r$. Again, the heart of the Miller loop involves the calculation of the Miller variable m, which consists of the calculation of g. In each round, g is calculated from the evaluation of the line function u, which is calculated as

$$u = [y_A - \lambda(x_Q + x_A)][-y_Q].$$

As before, in each round g is multiplicatively incorporated into the Miller variable as either

$$m \leftarrow (m)^2 \cdot g$$

or

$$m \leftarrow (m)^2 \cdot g \quad \text{and} \quad m \leftarrow m \cdot g$$

depending on the binary representation of the order of the Miller loop. Elements are in the extension field and consequently the output of the pairing is an element in $\mathbb{F}_{p^2}$.

The sign change fault attack of the Weil pairing will be used to examine the strength of the Tate pairing. This is for two main reasons. Firstly, the Tate pairing is most similar to the Weil pairing. Secondly the sign change fault attack is the most powerful of the attacks considered and so if Tate withstands this attack, it is inferred that it will withstand less powerful attacks such as the general data corruption fault.

Once again it is assumed that a fault has been injected into the final round of the Miller loop. The contribution of the line function for that round can thus be isolated as

$$\frac{e(P,Q)}{e(P,Q)'} = \frac{(m^{2^r} \cdot g)^{\frac{p^2-1}{r}}}{(m^{2^r} \cdot g')^{\frac{p^2-1}{r}}} = \frac{(u)^{\frac{p^2-1}{r}}}{(u')^{\frac{p^2-1}{r}}} = \left(\frac{[y_A - \lambda(x_Q + x_A)][-y_Q]}{[y_A - \lambda(x_Q + x_A)][y_Q]} \right)^{\frac{p^2-1}{r}}$$

Breaking the exponent into its factors as was demonstrated in equation (6.28), this can be simplified to

$$\sqrt{\frac{e(P,Q)}{e(P,Q)'}} = \pm \left(\frac{[y_A - \lambda(x_Q + x_A)][y_Q]}{[y_A - \lambda(x_Q + x_A)][-y_Q]} \right)^{\frac{p+1}{r}}. \tag{6.33}$$

However, the reversal of the exponent $(p+1)/r$ is infeasible. To access

$$\frac{[y_A - \lambda(x_Q + x_A)][y_Q]}{[y_A - \lambda(x_Q + x_A)][-y_Q]}, \tag{6.34}$$

and subsequently derive the secret elliptic curve point (whether it be P or Q), a specific n-th root, where $n = (p+1)/r$, must be calculated. Since n divides the group order once, there exists a simple formulae to compute a n-th root, i.e.

$$\left(\sqrt{\frac{e(P,Q)}{e(P,Q)'}} \right)^{n^{-1} \pmod{s}} \tag{6.35}$$

where $s = (p+1)/n$. However, The particular root of interest does not exhibit any special form, unlike the case of [107], and is a full quadratic element. Therefore, no extra information is available to aid in determining which is the root we are interested in. In addition, since there exists n n-th roots, where n is large, the cost of computing and testing all n-th roots, renders such an attack infeasible.

6.4.4 Summary of Findings

Three algorithms for bilinear pairings were examined for vulnerability to a fault attack. Two of these proved to succumb to a fault attack that facilitated the extraction of the secret.

It was shown that Galbraith et al.'s version [41] of the η pairing, the η_G pairing, is susceptible to a data corruption fault attack. However, the ability to extract the secret depends on where the fault is injected and which parameter P or Q is the secret input point to $\eta_G(P,Q)$. If the fault affects any of the cells of the line function l, when Q is secret, a linear equation with two unknown variables is derived. However, if the same fault affects l when P is secret, a non-linear equation is derived. Therefore, for this type of attack the Q path is the less secure path in the pairing $\eta_G(P,Q)$. If the fault affects any of the coordinates x_A, y_A, or y_Q, then a similar conclusion about the vulnerability of the P and Q paths can be reached. If the fault affects the cell v_0 or the coordinate x_C, then a simple congruence only needs solving, and so only requires one fault to be injected. This holds for both paths P and Q. If however the fault affects either the slope λ or the coordinate x_Q, then the steps to extract the secret are not as straightforward. In the case of the corruption of λ the number of unknown variables in the system of linear equations is increased. To generate the required number of equations to solve the system requires an identical fault to be injected into multiple pairing executions. If the fault affects the coordinate x_Q, then a non-linear equation is obtained.

It was shown that the type of data corruption fault, which was successfully applied

to the η_G pairing, was ineffective against the Weil pairing. However, it was shown that another type of fault attack, namely a sign change fault, could be successfully applied to the Weil pairing to reveal the secret. The attack described was demonstrated to be equally powerful against both the P and Q path.

It is not possible to attack the Tate pairing with either a general data corruption fault, or a sign change fault attack. This is on account of the final exponentiation which makes accessing the output of the Miller loop a difficult root finding problem.

In addition to assessing three bilinear pairing algorithms, a number of points were made about the optimal time and location to inject a data corruption fault. In particular, it was noted that the final round is the best round to target and it is more advantageous to inject local faults. It was also discovered that pre-computation can actually aid a data corruption fault attack.

6.5 Countermeasures

Apart from choosing a pairing algorithm which has a complex final exponentiation, a number of fault obfuscation and detection mechanisms can be used to prevent the described attacks. In this section, various techniques will be presented.

6.5.1 Fault Obfuscation Mechanisms

The aim of fault obfuscation techniques is to make the faulty pairings acquired by the adversary, useless. Page and Vercauteren [107] describe a number of techniques that blind the input point known to the adversary. For example, by computing

$$\rho(sP, Q)^{s^{-1}},$$

where P is public, Q is the secret and s is a random value in $\mathbb{F}_q$, the values required for use in Equations (6.16), (6.17) and (6.18) and (6.29) or (6.30), i.e x_P, y_P, x_A and y_A,

will no longer be computable by an adversary.

6.5.2 Fault Detection Mechanisms

Numerous software and hardware mechanisms already exist to detect a fault attack that are not algorithm specific [11]. For example, the simple act of executing the algorithm twice to check if the results are the same can be applied to any algorithm. The property of bilinearity inherent in pairings allows a method of fault detection, which is specific to pairings. By checking whether

$$\rho(P,Q)^{sr} = \rho(sP, rQ), \tag{6.36}$$

any faults injected will be caught. This can be applied to all pairings and should pick up any type of fault. The only drawback of this fault detection mechanism is that it is quite costly, requiring two pairing computations and additional point scalar multiplication of two points and exponentiation of an element in the extension field $\mathbb{F}_{q^k}$.

Other more cost effective methods of fault detection could involve random checking of whether the intermediate points used in the computation are still points on the curve, i.e check if $x_i, y_i \in y^2 = x^3 + ax + b$ or $y^2 + y = x^3 + x + b$. A check could be carried out in every round, however this could be expensive. There is also a possibility that this check will not be reliable. For example, the sign change fault attack of the Weil pairing returns a point that is still on the curve and so will pass this check.

6.5.3 Hiding the n-th Root

In the case of the Tate pairing where the difficulty of extracting the secret is based on a n-th root problem, one approach to further harden the Tate pairing against such fault attacks is to ensure that the n-th root of interest is difficult to obtain. In particular, in the event of the exact n-th root being identified, to combat data corruption fault attacks it is

possible to set up the elliptic curve parameters such that n^2 divides the group order once (i.e. p and r are specially chosen to meet this requirement). Now the problem of finding just one n-th root, i.e solving Equation (6.35), is equivalent to solving a Discrete Log Problem [54].

6.6 Conclusion

In this chapter, it was shown that bilinear pairings incorporating of a simple or no final exponentiation are vulnerable to data corruption fault attacks. Specifically, attacks on the η_G pairing, which reduced extracting the secret elliptic curve point to solving in one case a system of linear equations, and in another a simple congruence. A sign change fault attack of the Weil pairing was also demonstrated, which reduced extracting the secret elliptic curve point to solving a cubic equation. The attacks of the η_G and Weil are not possible on the Tate pairing, due to its final exponentiation.

In summary, the success of these attacks depends on the difficulty of reversing the final exponentiation. It is the nature of the final exponentiation to produce a unique value, however in doing this it also destroys information, making it difficult to recover the value which was exponentiated.

Chapter 7

Conclusion

7.1 Review

First-order power analysis is based on exploiting operations involving data known to the adversary and data related to the secret. In this thesis, three bilinear pairings, namely the Tate, Ate and η_T pairing, were assessed for vulnerability to first-order power analysis. The places where known data comes in contact with secret data were identified. These points of contact between known and secret data occurs during various finite field operations such as finite field addition, multiplication, square root and reduction. Attacks on the finite field operations, multiplication, square root and reduction, were proposed. These attacks, which focus on analysing the structure of an operation, are based on the first-order power analysis attack Correlation Power Analysis (CPA). The implications of an attack on these finite field operations were then examined in the context of the candidate bilinear pairings.

Each of the bilinear pairings assessed proved to be theoretical susceptible to first-order power analysis. However, it was shown that certain algorithms are more susceptible than others. In particular, it was shown that the number of options for first-order power analysis varies depending on the bilinear pairing, and the path that the secret takes. It was

shown that in the BKLS algorithm for the Tate pairing $e(P, Q)$ when the secret is entered as the P parameter, and so takes the path relating to P, then the number of operations involving known and secret data is less than if the secret is entered as the Q parameter and takes the path relating to Q. In fact, when the secret takes the P path there is only one operation per iteration of the Miller loop involving the secret and known data. In contrast to this when the secret takes the Q path there are three operations per iteration of the Miller loop involving the secret and known data. In the Ate pairing, a similar difference between the two paths P and Q was witnessed. However, in the η_T pairing, the number of options for using first-order power analysis was equivalent for both paths.

One reason why the Tate and Ate pairings present two varying choices for attack and the η_T pairing presents two equally vulnerable choices for attack, can be explained by looking at the original line function from Miller's algorithm

$$l_{A,B}(C) = (y_C - y_i) - \lambda_i(x_C - x_i), \tag{7.1}$$

where the line function is being calculated at the point C. Note that C will relate to Q when the Miller loop is calculating $[r]P$, and will relate to P when the Miller loop is calculating $[r]Q$. If the secret relates to (x_C, y_C) then the focus of the attack can be on either $(x_C - x_i)$, $\lambda_i(x_C - x_i)$, $y_i - \lambda_i(x_C - x_i)$ or $(y_C - y_i)$, depending on what field the coordinates are part of. Whereas if the secret relates to (x_i, y_i) then the focus of the attack can only be on either of the operations $(y_C - y_i)$ or $(x_C - x_i)$, since the operation of $\lambda_i(x_C - x_i)$ will involve two unknown values x_i and λ_i. Also, if the secret relates to (x_C, y_C) then the focal operations where known data comes in contact with secret data are the subtraction/addition and multiplication operation. While if the secret relates to (x_i, y_i) then there is only one focal operation where known data comes in contact with secret data, namely the subtraction/addition operation.

When only one application of Miller's algorithm is required, as is the case with the

Tate and Ate pairing, then the secret will present at most two or up to four avenues of attack depending on whether the secret is (x_C, y_C) or (x_i, y_i). However, if two applications of Miller's algorithm are required, then an equal number of attack options will exist for the two paths since in either the first or second application of Miller's algorithm the secret will take the place of the (x_C, y_C) coordinate in the line function $l_{A,B}(C)$ and so will present the maximum options for attack.

This ultimately forms the argument that in the context of first-order power analysis, bilinear pairings which require two applications of Miller's algorithm will exhibit two equally vulnerable paths of attack via first-order power analysis. Furthermore, if the line function is calculated as in Equation (7.1) and only one application of Miller's algorithm is required, then the path that provides the fewest avenues of attack is the P path.

Other findings presented in this thesis further consolidate the argument that the Q path is less secure than the P when only one application of Miller's algorithm is required. From Equation (7.1), it can be seen that the coordinates (x_C, y_C), which will relate to Q, and so the coordinates (x_Q, y_Q), are used without change in every round of the Miller loop. Therefore, the various hypotheses that are made for the value of Q, can be tested in a number of rounds. For example, if a prediction is made for the value of x_Q (or even a part of x_Q) then these predictions can be tested in the rounds $i-1, i, i+1$;

$$\begin{aligned} l(Q) &= (y_Q - y_{i-1}) - \lambda_{i-1}(x_Q - x_{i-1}) \quad \ldots \text{ Round } i-1 \\ l(Q) &= (y_Q - y_i) - \lambda_i(x_Q - x_i) \quad \ldots \text{ Round } i \\ l(Q) &= (y_Q - y_{i+1}) - \lambda_{i+1}(x_Q - x_{i+1}) \quad \ldots \text{ Round } i+1 \end{aligned}$$

This describes a form of second-order power analysis which combines multiple samples from within one power trace.

These findings leads us to the first two recommendations relating to the secure implementation of bilinear pairings when considering power analysis. The first recommenda-

tion has additional benefits in terms of efficiency.

Recommendation 1. *Bilinear pairings that require one application of Miller's algorithm are preferable to bilinear pairings which require two.*

Recommendation 2. *Bilinear pairings that require one application of Miller's algorithm, and execute the line function $l_{A,B}(C)$ in its original form, should accept the secret elliptic curve point as the first input parameter to the pairing.*

Now, in the case with the η_T pairing, which presents two equally vulnerable paths to attack, this option for secret parameter placement will have no bearing on the number of access points for first-order power analysis. Therefore, in order to protect the η_T pairing against first-order power analysis, countermeasures to deter power analysis must be implemented. One approach to deter first-order power analysis is to limit or effectively hide operations involving known and secret data. In Chapter 5, a number of countermeasures to achieve this effect were proposed. Such countermeasures should be used not only on the η_T pairing but in all bilinear pairings. This leads us to the next recommendation for secure bilinear pairing implementation.

Recommendation 3. *All bilinear pairings should be implemented with countermeasures to deter power analysis.*

Fault attacks seek to exploit purposely induced faults in a computation. In this thesis, three bilinear pairings, namely the Weil, Tate and Galbraith et al.'s η_G pairing, were assessed for vulnerability to fault attacks. In particular, these candidate pairings were assessed for vulnerability to data corruption fault attacks. It was shown that the distinguishing factor between whether a bilinear pairing succumbs to a data corruption fault

attack or not could be attributed to the complexity of the final exponentiation employed. A simple final exponentiation, or even worse no final exponentiation, allows a data corruption fault to derive the secret, whereas a complex final exponentiation disables a data corruption fault and prevents access to the secret. This was demonstrated in the description of two types of data corruption fault attacks on the Weil and η_G pairing, which could not be applied to the Tate pairing. In the implementations examined the η_G pairing did not require any final exponentiation, the Weil pairing employed a simple final exponentiation and the Tate pairing employed a complex final exponentiation. The reason why a complex final exponentiation, and in particular a final exponent of $(q^k - 1)/r$, is prohibitive to fault analysis is the fact that reversing it is difficult.

Furthermore, it was shown that certain measures could be used to further complicate the reversal of final exponentiation. In particular, it was found that utilising the full field representation is important. Since reversing the final exponentiation is an instance of a n-th root problem, failing to use the full field representation will provide an adversary with additional information that may be used to simplify root finding.

It was also found that the intermediate points that are calculated as part of Miller's algorithm should be dynamically computed as part of the computation, as opposed to pre-computing the required points and slopes and storing them on the device. Pre-computation, which is a common measure to increase efficiency, can actually aid data corruption fault attacks since corrupted data can be contained instead of contaminating other information.

These findings lead us to the following recommendations for secure bilinear pairing implementation when considering fault attacks.

Recommendation 4. *The final exponentiation should be included and chosen such that reversal of it is difficult. Specifically, a final exponent of the form* $(q^k - 1)/r$ *is recommended.*

Recommendation 5. *The Miller variable calculated at each round of the Miller loop will be an element in the extension field $\mathbb{F}_{q^k}$. It is recommended that the full field representation of this element is utilised. For example, if $k = 6$, then all six coefficients of the element should be active.*

Recommendation 6. *It is recommended that the intermediate points required for the computation of $[r]P$ (or $[r]Q$) are not pre-computed, stored and looked up as part of the pairing algorithm. Instead, it recommended that they are dynamically computed. This may be at a cost to the computation time of the pairing.*

Recommendation 7. *All bilinear pairings should be implemented with countermeasures to deter fault analysis.*

In this thesis, some of the key characteristics which compromise the security of bilinear pairings have been identified. Most of these characteristics, which have weakened the bilinear pairing, have generally been incorporated to make the bilinear pairing more efficient. However, as is the case with all cryptosystems, there reaches a point where efficiency will compromise security. It is the hope that with the identification and recognition of such compromising characteristics a secure and robust definition of a bilinear pairing can be achieved in all security settings.

7.2 Future Work & Open Questions

Since bilinear pairings possess such a complex structure, and both fields of implementation attacks and pairing based cryptography are constantly evolving, there are lots of aspects yet to be investigated. The open questions that have arisen as a result of this thesis will be examined here.

Firstly, with respect to passive side channel attacks on bilinear pairings, there are many choices of parameters for the pairing. The elliptic curve can be supersingular or non-supersingular. It can be elliptic (genus 1) or hyper-elliptic (genus $\geq$ 2) [49]. The finite field over which the elliptic curve lies can be over the large or small prime field. The embedding degree can range from 2 to 24, which will consequently affect the size of the underlying field. The coordinate representation can be affine, projective or Jacobian. To date, the main bilinear pairings are the Weil, Tate, Ate, η and η_T pairing. Such parameter choices can affect how these bilinear pairings are implemented, and so may expose further implementation vulnerabilities. In addition, there are various passive side channel attacks to consider. Chapter 2 touched on some of the most potent attacks, such as second-order power analysis, template attacks, EM attacks and timing attacks. In this thesis, specific pairings where examined for susceptibility to first-order power analysis. However, the choices for pairing algorithms, pairing parameters and side channel attack present many open problems.

Another key aspect which deserves further research is the execution of an implementation attack on a bilinear pairing algorithm in practice. The work presented in this thesis was theoretical. The work of Scott et al. [120], who presented the first timings of bilinear pairings on a smart card, actually performed their analysis using an FPGA based emulator, in which the smart card core is embedded. To perform these attacks in practice, access to a smart card implementing pairings is required, which to date has been unavailable.

In the context of fault analysis of bilinear pairings, some of the main issues mentioned above, will also affect the potential for fault attacks. There are also various types of fault attacks with different affects to consider [11]. In addition, the main type of fault attack that was concentrated on in this thesis was Simple Fault Analysis (SFA). One possible avenue for further investigation is Differential Fault Analysis (DFA), where a collection of multiple valid-faulty pairing pairs can aid in determining the secret key. For example,

given the set

$$\{\rho(P,Q), \rho(P,Q)' \rho(P,Q)'', \rho(P,Q)''' \ldots\}$$

where multiple faults have been repeatedly injected into a pairing calculation for a given set of points P and Q, or the set

$$\{\rho(P_0,Q), \rho(P_1,Q)', \rho(P_2,Q)', \rho(P_3,Q)' \ldots\}$$

where a fault is injected into different pairing calculations for different points P_i, is any information about the secret revealed?

The recent development of Stange [126], who proposes a new technique based on elliptic nets to compute pairings, also raises some questions pertaining to implementation attacks. The research performed in this thesis, analysed bilinear pairings based on Miller's algorithm. Hence, analysis of elliptic nets for vulnerability to both power and fault analysis is another pertinent question that needs to be addressed.

Bibliography

[1] Side Channel Cryptanalysis Lounge. URL: http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html.

[2] Workshop on Fault Diagnosis and Tolerance in Cryptography - (FDTC). URL: http://www.elet.polimi.it/conferences/FDTC07/.

[3] O. Aciicmez, Ç. K. Koç, and J.P. Seifert. On the Power of Simple Branch Prediction Analysis. Cryptology ePrint Archive: Report 2006/351. URL: http://eprint.iacr.org/2006/351.

[4] L. Adams, E. J. Daly, and R. Harboe-Sorensen. A Verified Proton Induced Latchup in Space. In *IEEE Transactions on Nuclear Science*, volume 39, pages 1804–1808, 1992.

[5] D. Agrawal, J.R. Rao, P. Rohatgi, and K. Schramm. Templates as Master Keys. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 15–29. Springer Verlag, 2005.

[6] M. L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer Verlag, 2001.

[7] L. Anghel and M. Niclolaidis. Cost Reduction and Evaluation of Temporary Faults Detecting Techniques. In *Design, Automation and Test in Europe - DATE 2000*, pages 591–597, 2000.

[8] D. Asonov and R. Agrawal. Keyboard acoustic emanations. IBM Almaden Research Center. URL: `http://www.almaden.ibm.com/software/quest/Publications/papers/ssp04.pdf`, 2004.

[9] R. Avanzi. Side Channel Attacks on Implementations of Curve-Based Cryptographic Primitives. Cryptology ePrint Archive: Report 2005/017. URL: `http://eprint.iacr.org/2005/017`.

[10] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The Sorcerers Apprentice Guide to Fault Attacks. In *Workshop on Fault Detection and Tolerance in Cryptography - FDTC 2004*, 2004.

[11] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The Sorcerers Apprentice Guide to Fault Attacks. In *Proceedings of the IEEE, Special Issue on Cryptography and Security*, volume 96 of *2*, pages 370–382, February 2006.

[12] P. Barreto. Pairing Based Crypto Lounge. URL: `http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html`.

[13] P. Barreto, S. Galbraith, C. O'hEigeartaigh, and M. Scott. Efficient Pairing Computation on Supersingular Abelian Varieties. Cryptology ePrint Archive: Report 2004/375. URL: `http://eprint.iacr.org/2004/375`.

[14] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing Based Cryptosystems. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002*, vol-

ume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer Verlag, 2002.

[15] P. Barreto, B. Lynn, and M. Scott. On the Selection of Pairing Friendly Groups. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography - SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25, Ottawa, Canada, 2003.

[16] P. Barreto and M. Naehrig. Pairing Friendly Elliptic Curves of Prime Order. In *Selected Areas in Cryptography - SAC 2006*, volume 3897 of *Lecture Notes in Computer Science*, pages 391–331. Springer Verlag, 2006.

[17] D. J. Bernstein. Cache-timing attacks on AES. URL: `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`.

[18] I. Biehl, B. Meyer, and V. Muller. Differential Fault Attacks of Elliptic Curve Cryptosystems. In M. Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 131–146. Springer Verlag, 2000.

[19] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology - CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525, 1997.

[20] J. Blomer, M. Otto, and J-P. Seifert. Sign Change Fault Attacks on Elliptic Curve Cryptosystems. In Luca Breveglieri and Israel Koren, editors, *Workshop on Fault Detection and Tolerance in Cryptography - FDTC 2005*, pages 25–40, 2005.

[21] D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Journal of Cryptology*, volume 14, pages 101–119. SV, 2001.

[22] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Verlag, 2001.

[23] D. Boneh and R. Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 1996.

[24] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer Verlag, 2004.

[25] Gerolamo Cardano. Cardano's Formula for Solving a Cubic Equation. Wikipedia, URL: `http://en.wikipedia.org/wiki/CubicEquation`.

[26] Ph. Cazenave, P. Fouillata, X. Montagner, H. Barnaby, R. D. Schrimpf, L. Bonora, J. P. David, A. Touboul, M. C. Calvet, and P. Calvel. Total Dose Effects on Gate Controlled Lateral PNP Bipolar Junction Transistors. In *IEEE Transactions on Nuclear Science*, volume 45, pages 2577–2583, 1998.

[27] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. A Cautionary Note Regarding Evaluation of AES Candidates on Smartcards. In *Second Advanced Encryption Standard (AES) Candidate Conference*, 1999. URL: `http://csrc.nist.gov/encryption/aes/round1/Conf2/aes2conf.htm`.

[28] S. Chari, J. R. Rao, and P. Rohatgi. Template Attacks. In Kaliski Jr. B. S, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer Verlag, 2002.

[29] Z. Chen. *Java Card Technology for Smart Cards.* Number ISBN: 0201703297. Addison Wesley, 2000.

[30] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* Chapman and Hell/CRC, 2006.

[31] J. S. Coron. Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer Verlag, 1999.

[32] J. S. Coron and L. Goubin. On Boolean and Arithmetic Masking against Differential Power Analysis. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 231–237. Springer Verlag, 2000.

[33] J. S. Coron and A. Tchulkine. A New Algorithm for Switching from Arithmetic and Boolean Masking. In C. D. Walter, editor, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 89–97. Springer Verlag, 2003.

[34] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - Advanced Encryption Standard.* Number ISBN: 3540425802. Springer-Verlag, 2002.

[35] R. Dahab, A.J. Devegili, C. O hEigeartaigh, and M. Scott. Multiplication and Squaring on Pairing Friendly Fields. Cryptology ePrint Archive: Report 2006/, URL: `http://eprint.iacr.org/2006/`.

[36] R. Dutta, R. Barua, and P. Sarkar. Pairing Based Cryptographic Protocols: A Survey. Cryptology ePrint Archive: Report 2004/064, URL: `http://eprint.iacr.org/2004/064`.

[37] I. M. Duursma and H. S. Lee. Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$. In Chi Sung Laih, editor, *Advances in Cryptology - Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 111–123. Springer Verlag, 2003.

[38] K. Fong, D. Hankerson, J. Lopez, and A. Menezes. Field Inversion and Point Halving Revisited. CACR Technical Report, CORR 2003-18. URL: `http://www.cacr.math.uwaterloo.ca/`, 2003.

[39] D. Freeman, M. Scott, and E. Teske. A Taxonomy of Pairing Friendly Elliptic Curves. Cryptology ePrint Archive: Report 2006/372, URL: `http://eprint.iacr.org/2006/372`.

[40] G. Frey and H.G. Ruck. A Remark Concerning m-Divisibility and the Discrete Logarithm Problem in the Divisor Class Group of Curves. In *Mathematics of Computation*, volume 62(206), pages 865–874, 1994.

[41] S. Galbraith, C. OhEigeartaigh, and C. Sheedy. Simplified Pairing Computation and Security Implication. Cryptology ePrint Archive: Report 2006/, URL: `http://eprint.iacr.org/2006/`.

[42] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2126 of *LNCS*, pages 251–261. Springer-Verlag, 2001.

[43] J. D. Goli and C. Tymen. Multiplicative Masking and Power Analysis of AES. In Kaliski Jr. B. S, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer Verlag, 2002.

[44] L. Goubin. A Sound Method for Switching between Boolean and Arithmetic Masking. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hard-*

ware and Embedded Systems - CHES 2001, volume 2162 of *Lecture Notes in Computer Science*, pages 3–15. Springer Verlag, 2001.

[45] L. Goubin. A Refined Power Analysis Attack on Elliptic Curve Cryptosystems. In *Public Key Cryptography - PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 199–210. Springer Verlag, 2003.

[46] L. Goubin and J. Patarin. Des and Differential Power Analysis, the Duplication Method. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer Verlag, 1999.

[47] D. Han, J. Lim, and K. Sakurai. On Security of XTR Public Key Cryptosystems against Side Channel Attacks. Cryptology ePrint Archive: Report 2004/123. URL: `http://eprint.iacr.org/2004/123`.

[48] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Professional Computing, 2003.

[49] C. O hEigeartaigh. *Pairing Computation on Hyperelliptic Curves of Genus 2*. PhD thesis, Dublin City University, 2007.

[50] F. Hess, N. Smart, and F. Vercauteren. The Eta Pairing Revisited. Cryptology ePrint Archive: Report 2006/110. URL: `http://eprint.iacr.org/2006/110`.

[51] N. A. Howgrave-Graham and N. P. Smart. Lattice Attacks on Digital Signature Schemes. In *Design, Codes and Cryptography*, volume 23, pages 283–290, 2001.

[52] Gemplus Card International. Applied Research & Security Centre, Avenue des Jujubiers, La Ciotat, F-13705, France. URL: `http://http://www.gemplus.com/`.

[53] ISO7816. Identification Cards-Itegrated Circuits Cards with Contacts. International Organization for Standardization.

[54] A. Johnston. *On the Difficulty of Prime Root Computation in Certain Finite Cyclic Groups*. PhD thesis, Royal Holloway University of London, 2006. URL: `http://www.ma.rhul.ac.uk/techreports/`.

[55] A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In *Algorithmic Number Theory Symposium - ANTS IV 2000*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer Verlag, 2000.

[56] M. Joye, A. Lenstra, and J.-J. Quisquater. Chinese remaindering cryptosystems in the presence of faults. In *Journal of Cryptology*, volume 12, pages 241–245, 1999.

[57] M. Joye, P. Pallier, and B. Schoenmakers. On Second-Order Differential Power Analysis. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer Verlag, 2005.

[58] M. Joye and J.J. Quisquater. Efficient Computation of Full Lucas Sequences. In *Electronic Letters*, volume 32(6), pages 537–538, 1996.

[59] M. Joye and S-M. Yen. The Montgomery Powering Ladder. In Kaliski Jr. B. S, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer Verlag, 2002.

[60] J. P. Kaps. Workshop on Cryptographic Hardware and Embedded Systems - (CHES). URL: `http://islab.oregonstate.edu/ches/former.html`.

[61] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side Channel Cryptanalysis of Product Ciphers. In *Journal of Computer Security*, volume 8, pages 141–158, 2000.

[62] T.H. Kim, T. Takagi, D.G. Han, H. W. Kim, and J. Lim. Side Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields. In D. Pointcheval, Y. Mu, and K. Chen, editors, *Crypotgraphy and Network Security - CANS 2006*, volume 4301 of *Lecture Notes in Computer Science*, pages 168–181. Springer Verlag, December 2006.

[63] V. Klima and T. Rosa. Further Results and Considerations on Side Channel Attacks on RSA. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2162 of *Lecture Notes in Computer Science*, pages 244–259. Springer Verlag, 2002.

[64] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In *Advances in Cryptology - CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer Verlag, 1996.

[65] P. Kocher, J. Jaffe, and B. Jun. Overview of Differential Power Analysis. URL: `www.cryptography.com/resources/whitepapers/DPATechInfo.pdf`, 1998.

[66] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer Verlag, 1999.

[67] R. Koga and W. A. Kolasinski. Heavy Ion Induced Snapback in CMOS Devices. In *IEEE Transactions on Nuclear Science*, volume 36, pages 2367–2374, 1989.

[68] R. Koga, M. D. Looper, S. D. Pinkerton, W. J. Stapor, and P. T. McDonald. Low Dose Rate Proton Irradiation of Quartz Crystal Resonators. In *IEEE Transactions on Nuclear Science*, volume 43, pages 3174–3181, 1996.

[69] O. Kommerling and M. Kuhn. Design Principles for Tamper Resistant Smart Card Processors. In *Proceedings of the USENIX Workshop on Smartcard Technology*, pages 9–20, 1999.

[70] S. Kuboyama, S. Matsuda, T. Kanno, and T. Ishii. Mechanism for Single-Event Burnout of Power MOSFETs and its Characterization Technique. In *IEEE Transactions On Nuclear Science*, volume 39, pages 1698–1703, 1992.

[71] S. Kwon. Efficient Tate Pairing Computation for Supersingular Elliptic Curves over Binary Fields. Cryptology ePrint Archive: Report 2004/303. URL: `http://eprint.iacr.org/2004/303`.

[72] T. Le, J. Clédière, C. Canovas, B. Robisson, C. Servière, and J. Lacoume. A Proposition for Correlation Power Analysis Enhancement. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 174–186. Springer Verlag, 2006.

[73] K. Lemke, K. Schramm, and C. Paar. Dpa on n-Bit Sized Booloean Arithmetic Operations and its applications to IDEA, RC6 and the HMAC-Conctruction. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 174–186. Springer Verlag, 2006.

[74] R. Lidl and H. Niederreiter. *Encyclopedia of Mathematics 20: Finite Fields*. Cambridge University Press, 2003.

[75] C. H. Lim and H. S. Hwang. Fast Implementation of Elliptic Curve Arithmetic in GF(p^n). In *International Workshop on Theory and Practice in Public Key Cryptography - PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 405–421. Springer Verlag, 2000.

[76] J. Losq. Influence of Fault Detection and Switching Mechanisms on Reliability of Stand-by Systems. In *Digest 5th International Symp. Fault Tolerence Computing*, pages 81–86, 1975.

[77] W. Mao. *Modern Cryptography, Theory and Practice*. Prentice Hall PTR, 2003.

[78] T. May and M. Woods. A New Physical Mechanism for Soft Errors in Dynamic Memories. In *16th International Reliability Physics Symposium*, 1978.

[79] A. Menezes. *Elliptic Curve Public Key Cryptosystems*. The International Series in Engineering and Computer Science. Springer, 1993.

[80] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[81] A.J. Menezes, T. Okamoto, , and S. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. In *IEEE Transactions on Information Theory*, volume 39, pages 1639–1646, 1993.

[82] T. Messerges. *Power Analysis Attacks and their Countermeasures*. PhD thesis, University of Illinois Chicago, 2000.

[83] T. Messerges. Securing the AES Finalists against Power Analysis Attacks. In B. Schneier, editor, *Fast Software Encryption - FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer Verlag, 2000.

[84] T. Messerges. Using Second Order Power Analysis to attack DPA Resistant Software. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded*

Systems - CHES 2000, volume 1965 of *Lecture Notes in Computer Science*, pages 238–252, 2000.

[85] T. Messerges, E. Dabbish, and R. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *USENIX Workshop on Smartcard Technology*, pages 151–161, 1999.

[86] T. Messerges, E. Dabbish, and R. Sloan. Power Analysis Attacks of Modular Exponentiation in Smartcards. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 144–157. Springer Verlag, 1999.

[87] T. Messerges, E. Dabbish, and R. Sloan. Examining Smartcard Security Under the Threat of Power Analysis Attacks. In *IEEE Transactions on Computers*, volume 51, 2002.

[88] V. Miller. The Weil Pairing, and its Efficient Calculation. In *Journal of Cryptology*, volume 17, pages 235–261, 2004.

[89] A. Miyaji, M. Nakabayashi, and S. Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. In *IEICE Transactions on Fundamentals*, volume E84-A(5), pages 1234–1243, 2001.

[90] D. Montgomery and G. Runger. *Applied Statistic and Probability for Engineers*. John Wiley & Sons, Inc., 2002.

[91] P.L. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. In *Mathematics of Computation*, volume 48(177), pages 243–264, 1987.

[92] David Naccache, Michael Tunstall, and Claire Whelan. Computational Improvements to Differential Side Channel Analysis. In *NATO Security through Science Series D: Information and Communication Security*, volume 2, pages 26–35, January 2006.

[93] P. Q. Nguyen and J. Stern. The Two Faces of Lattices in Cryptology. In *Cryptography and Lattices -CALC '01)*, volume 2146 of *Lecture Notes in Computer Science*, pages 146–180. Springer Verlag, 2001.

[94] P. Q. Nguyễn and I. E. Shparlinski. The Insecurity of the Digital Signature Algorithm with Partially Known Nonces. In *Journal of Cryptology*, volume 15, pages 151–176. Springer, 2002.

[95] Phong Nyugen, David Naccache, Michael Tunstall, and Claire Whelan. Experimenting with Faults, Lattices and the DSA. In Serge Vaudenay, editor, *8th International Workshop on Theory and Practice in Public Key Cryptography - PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 16–28. Springer, January 2005.

[96] National Bureau of Standards. *FIPS PUB 46: The Data Encryption Standard.* 1977.

[97] National Institute of Standards and Technology. *FIPS PUB 186-2: Digital Signature Standard.* 2000.

[98] National Institute of Standards and Technology. *FIPS PUB 197: The Advanced Encryption Standard.* 2001.

[99] National Institute of Standards and Technology. *FIPS PUB 186-2: Security Requirements for Cryptographic Modules.* 2002.

[100] T. J. O'Gorman. The Effect of Cosmic Rays on Soft Error Rate of a DRAM at Ground Level. In *IEEE Transactions On Electronics Devices*, volume 41, pages 553–337, 1994.

[101] S. B. Ors, E. Oswald, and B. Preneel. Power Analysis Attacks on an FPGA - First Experimental Results. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *Crypto-*

graphic Hardware and Embedded Systems - CHES 2003, volume 2779 of *Lecture Notes in Computer Science*, pages 35–50. Springer Verlag, 2003.

[102] S.B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel. Power Analysis Attack on an ASIC AES Implementation. In *International Conference on Information Technology - ITCC 2004*, pages 546–552, 2004.

[103] E. Oswald and S. Mangard. Template Attacks on Masking - Resistance is Futile. In *CCT-RSA 2007*, Lecture Notes in Computer Science, pages 243–257. Springer Verlag, 2007.

[104] E. Oswald and S. Mangard. Template Attacks on Masking - Resistance is Futile. In *RSA Conference 2007 Cryptographers' Track (- CT-RSA 2007*, Lecture Notes in Computer Science, pages 243–256. Springer Verlag, 2007.

[105] D. Page and F. Vercauteren. Fault and Side-Channel Attacks on Pairing Based Cryptography. Cryptology ePrint Archive: Report 2004/283. URL: `http://eprint.iacr.org/2004/283`.

[106] D. Page and F. Vercauteren. A Fault Attack on Pairing Based Cryptography. In Luca Breveglieri and Israel Koren, editors, *Fault Detection and Tolerance in Cryptography - FDTC 2005*, pages 56–66, 2005.

[107] D. Page and F. Vercauteren. A Fault Attack on Pairing Based Cryptography. In *IEEE Transactions on Computers*, volume 55(9), pages 1075–1080, July 2006.

[108] J. H. Patel and L. Y. Fung. Concurrent Error Detectionin ALUs by Recomputing with Shifted operands. In *IEEE Transactions on Computers*, volume C-31, pages 589–595, 1982.

[109] C. Percival. Cache Missing for Fun and Profit. URL: `http://www.daemonology.net/papers/htt.pdf`.

[110] J. C. Pickel and J. T. Blandford. Cosmic Ray Induced Errors in MOS Memory Circuits. In *IEEE Transactions On Nuclear Science*, volume 25, pages 1166–1171, 1978.

[111] J.J. Quisquater and D. Samyde. A New Tool for Non-Intrusive Analysis of Smart Cards based on Electromagnetic Emissions: the SEMA and DEMA Methods. Eurocrypt Rump Session, 2000.

[112] C. Rechberger and E. Oswald. Practical template attacks. In *WISA 2004*, Lecture Notes in Computer Science, pages 440–456. Springer Verlag, 2004.

[113] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairings. In *Symposium on Cryptography and Information Security*, pages 26–28, 2000.

[114] T. Satoh. On Polynomial Interpolations related to Verheul Homomorphisms. In *London Mathematics Society, Journal of Computation and Mathematics*, volume 6, pages 135–158, 2006.

[115] J. Groschädl and E. Savaş. Instruction Set Extensions for Fast Arithmetic in Finite Fields GF(p) and GF(2^m). In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 133–147. Springer Verlag, 2005.

[116] C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. In *Math. Programming*, volume 66, pages 181–199, 1994.

[117] M. Scott. Multiprecision Integer and Rational Arithmetic C/C++ Library - MIRACL. URL: `http://www.shamus.ie`.

[118] M. Scott. Computing the Tate Pairing. In *Topics in Cryptology - CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer Verlag, 2005.

[119] M. Scott and P. Barreto. Generating more MNT elliptic curves. Cryptology ePrint Archive: Report 2004/058, URL: `http://eprint.iacr.org/2004/058.pdf`.

[120] M. Scott, N. Costigan, and W. Abdulwahab. Implementing Cryptographic Pairings on Smart Cards. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 134–147. Springer Verlag, 2006.

[121] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology - CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1985.

[122] Adi Shamir and Eran Tromer. Acoustic Cryptanalysis: On Nosy People and Noisy Machines. URL: `http://theory.csail.mit.edu/ tromer/acoustic/`.

[123] V. Shoup. Number Theory C++ Library - NTL. URL: `http://www.shoup.net/ntl/`.

[124] S. Skorogogatov and R. Anderson. Optical Fault Induction Attacks. In Kaliski Jr. B. S, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer Verlag, 2002.

[125] J. Solinas. Generalized Mersenne Numbers. Technical Report CORR-39, URL: `http://www.cacr.math.uwaterloo.ca/`.

[126] K. E. Stange. The Tate Pairing via Elliptic Nets. Cryptology ePrint Archive: Report 2006/392, URL: http://eprint.iacr.org/2006/392.

[127] E. G. Stassinopoulos, G.J. Brucker, P. Calvel, A. Baiget, C. Peyrotte, and R. Gaillard. Charge Generation by Heavy Ions in Power MOSFETs, Burnout Space Predictions and Dynamic SEB Sensitivity. In *IEEE Transactions on Nuclear Science*, volume 39, pages 1704–1711, 1992.

[128] D. Stebila and N. Theriault. Unified Point Addition Formulae and Side Channel Attacks. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 354–368. Springer Verlag, 2006.

[129] E. Trichina, D. De Seta, and L. Germani. Simplified Adaptive Multiplicative Masking for AES. In Kaliski Jr. B. S, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 187–197. Springer Verlag, 2002.

[130] E. Verheul. Evidence that XTR is more Secure than Supersingular Elliptic Curve Cryptosystems. In B Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210, 2001.

[131] J. Waddle and D. Wagner. Towards Efficient Second-Order Power Analysis. In M. Joye and J. J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 1–15. Springer Verlag, 2004.

[132] C. Walter. Sliding Windows Succumbs to Big Mac Attack. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 286–299. Springer Verlag, 2001.

[133] C. Walter. Mist: An Efficient Randomized Exponentiation Algorithm for Resisting Power Analysis. In B. Preneel, editor, *Topics in Cryptology - CT-RSA 2002. The Cryptographer's Track at the RSA Conference*, volume 2271 of *Lecture Notes in Computer Science*, pages 53–66. Springer Verlag, 2002.

[134] C. Walter. Simple Power Analysis of Unified Code for ECC Double and Add. In M. Joye and J. J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 191–204. Springer Verlag, 2004.

[135] C. Walter and S. Thomson. Distinguishing Exponent Digits by Observing Modular Subtractions. In D. Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference*, volume 2020 of *Lecture Notes in Computer Science*, pages 192–207. Springer Verlag, 2001.

[136] L. Washington. *Elliptic Curves, Number Theory and Cryptography*. Chapman and Hell/CRC, 2003.

[137] C. Whelan and M. Scott. The Importance of the Final Exponentiation in Pairings when Considering Fault Attacks. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *First International Conference on Pairing-Based Cryptography - Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 225–246. Springer, 2007.

[138] C. Whelan and M. Scott. Side Channel Analysis of Practical Pairings: Which Path is more Secure? In Phong Q. Nguyen, editor, *International Conference on Cryptology in Vietnam - VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 99–114. Springer Verlag, September 2006.

[139] P. Wright. *Spy Catcher: The Candid Autobiography of a Senior Intelligence Officer*. Viking Press, 1987.

Appendix A

Numerical Examples of Power Analysis

The purpose of this appendix is to demonstrate how the various finite field operations can be used as eligible selection functions, and so are suitable for use in a first-order power analysis attack, as was described in Chapter 5. Software was written in both Java and C to simulate the following examples.

A.1 The Square Root Operation

Let k relate to the secret value of interest. At some stage in the computation, an operation of the form $\alpha_i + \sqrt{k}$ is calculated involving known information α_i. In order to be able to make predictions for the output of $\alpha_i + \sqrt{k}$, the adversary must be able to calculate $\sqrt{k}$ given guesses for k. How analysing the structure of the square root operation gives insight into the relationship between k and $\sqrt{k}$, will be demonstrated in this section.

A.1.1 Data Values

Field parameters:

The implementation assessed is over the binary field $\mathbb{F}_{2^{379}}$, with reduction polynomial $f(z) = z^{379} + z^{315} + z^{301} + z^{287} + 1$. From [38], $\sqrt{z} = z^{190} + z^{158} + z^{151} + z^{144}$.

A.1.2 Example

The secret of interest k is

$k =$ 007B0C7AFE0A33D7A270D8A35B995B3546D77D6E14C1DDC6
32159FA6E2AC3E204F6961869342DC99BC2684EDD71C57AB

where the most significant bit is on the left-hand-side and the least significant bit is on the right-hand-side.The square root of k is

$\sqrt{k} =$ 01C9FCD674E126852CC10E97724E1F5405BC1DA5509ED26C
3097661F515ED5D7AFFA69FA47728260B99258E5622BF6F1.

It is assumed that the target device has a 32-bit processor, and so is similar to the smart card of Scott et al. [120]. The least significant word of k,

$k =$ 007B0C7AFE0A33D7A270D8A35B995B3546D77D6E14C1DDC6
32159FA6E2AC3E204F6961869342DC99BC2684EDD71C57AB

affects the the least significant 16 bits of $\sqrt{k}$,

$\sqrt{k} =$ 01C9FCD674E126852CC10E97724E1F5405BC1DA5509ED26C
3097661F515ED5D7AFFA69FA47728260B99258E5622BF6F1

as follows. Firstly, the least significant word of k will be split into odd and even parts. The odd part of the least significant word of k, will then be multiplied by $\sqrt{z}$, reduced by $f(z)$, then added to the even part of the least significant word of k. The splitting of

D71C57AB into its odd and even parts is depicted in figure A.1.

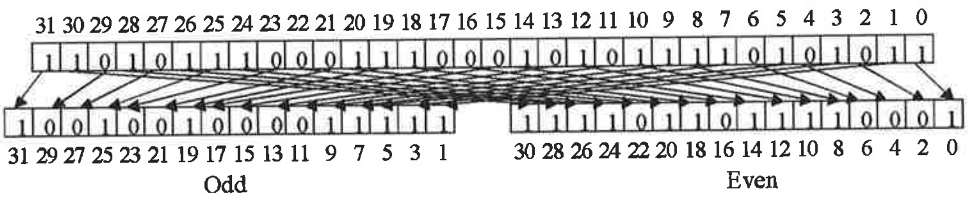

Figure A.1: Splitting the least significant word of k, `D71C57AB`, into its odd and even parts.

The multiplication of the odd part of the least significant word of k by $\sqrt{z}$ can be calculated since least significant word of k will be predicted in the hypothesis test and $\sqrt{z}$ is public information. The subsequent reduction can also be calculated since $f(z)$ is also known, however a reduction may not be required since odd $\cdot\sqrt{z}$ may be less than $f(z)$.

odd $\cdot\sqrt{z} \pmod{f(z)} =$

```
100100100001111100000000000000001001001100111001011101100111
110000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000
```

The addition of the resultant value, odd $\cdot\sqrt{z} \pmod{f(z)}$, to the even least significant word of k can also be calculated given the guess for the least significant word of k.

even $+(\text{odd}\cdot\sqrt{z} \pmod{f(z)}) =$

```
100100100001111100000000000000001001001100111001011101100111
110000000000000000000000000000000000000000000000000000000000
```

0000000000000000000000000000000000000000000000000000000000000000
0000000000111110111110001

When converted back to hexadecimal, it can be seen that the least significant 16 bits of the result corresponds to the least significant 16 bits of the $\sqrt{k}$.

$\sqrt{k}$ = 0000000000000000000000000000000000000002487C000
24CE5D9F0000000000000000000000000000000F6F1

As mentioned in Section 5.4.6, making guesses for 32 bits is infeasible. The technique described above will also work when considering smaller portions of k. For example, the last byte of k,

k = 007B0C7AFE0A33D7A270D8A35B995B3546D77D6E14C1DDC6
32159FA6E2AC3E204F6961869342DC99BC2684EDD71C57AB

will affect the least significant 4 bits of $\sqrt{k}$,

$\sqrt{k}$ = 01C9FCD674E126852CC10E97724E1F5405BC1DA5509ED26C
3097661F515ED5D7AFFA69FA47728260B99258E5622BF6F1

the second last byte of k,

k = 007B0C7AFE0A33D7A270D8A35B995B3546D77D6E14C1DDC6
32159FA6E2AC3E204F6961869342DC99BC2684EDD71C57AB

will affect the next least significant 4 bits of $\sqrt{k}$,

$\sqrt{k}$ = 01C9FCD674E126852CC10E97724E1F5405BC1DA5509ED26C
3097661F515ED5D7AFFA69FA47728260B99258E5622BF6F1

and so the last 16 bits of k,

$k =$ 007B0C7AFE0A33D7A270D8A35B995B3546D77D6E14C1DDC6
32159FA6E2AC3E204F6961869342DC99BC2684EDD71C57AB

will affect the the least significant 4 bits of $\sqrt{k}$,

$\sqrt{k} =$ 01C9FCD674E126852CC10E97724E1F5405BC1DA5509ED26C
3097661F515ED5D7AFFA69FA47728260B99258E5622BF6F1.

This enables hypotheses which are made for portions of k to be subsequently tested in operations such as $\alpha_i + \sqrt{k}$ involving known data.

A.2 The Multiplication Operation

Let x_Q relate to the secret value of interest. At some stage in the computation, an operation of the form $\lambda_j(x_j + x_Q)$ is calculated involving known information λ_j and x_j. In order to perform first-order power analysis, it must be possible to make predictions for the output of $\lambda_j(x_j + x_Q)$. Analysing the structure of the multiplication operation, aids in determining which parts of λ_j, x_j and x_Q, affect $\lambda_j(x_j + x_Q)$. This will be demonstrated in this section. Note that the bilinear pairing in question is the Tate pairing as described in Chapter 5.

A.2.1 Data Values

Field parameters:

The implementation assessed is over the prime field $\mathbb{F}_p$. The elliptic curve is $E(\mathbb{F}_p)$: $y^2 = x^3 + ax + b$ where

$p =$
8D5006492B424C09D2FEBE717EE382A57EBE3A352FC383E1AC79F21DDB43
706CFB192333A7E9CF644636332E83D90A1E56EFBAE8715AA07883483F82

67E80ED3

$a =$

8D5006492B424C09D2FEBE717EE382A57EBE3A352FC383E1AC79F21DDB43
706CFB192333A7E9CF644636332E83D90A1E56EFBAE8715AA07883483F82
67E80ED0

$b =$

609993837367998001C95B87A6BA872135E26906DB4C192D6E038486177A3
EDF6C50B9BB20DF881F2BD05842F598F3E037B362DBF89F0A62E5871D41D9
51BF8E

and the order $r =$

800000000000000000000000000000020001

Input Points to $e(P, Q)$:

P:

(24D466C77758F51374CC63BAF7055563B11D11399561914B45B937C02AD5
3318F8413E912324BA19797A817E1D5D9F9DCF2FBBFB2A005782D79B489FD
CFD542E,3C020B6B2170E1A586C5D3E08B41D16FA415C623DB7D5DA1B3FB9
E7E1AB7FB812F90A476953D14C11206B63BC31A7F5393D4049B357E307550
742246A5B07677)

Q:

(69B083576981F45F3D4E8E3EF12B1E37C35A1AA17F8A79591EF868AB34E1
CF19982EAC7E460C560AFAB962997D2E25F8359B3AA4BBAABCDAFD18C523F
1F6F8F5,7FB6FFB5BF46DC0EB17B7B720C283B5A7C224C1E1177C2D23570B
F8C758528CCDCE9781DC273D092B98C86F74EF9B4299FE73657F7E9BFB58C

F7B981B8087150)

A.2.2 Example

Let round two of the Miller loop be the target round in the power trace. The data values relating to this round are.

x_j:

18F45F42ED73EB0AA1199916336F88915A69643CBAB84EB65D1BF0CB15200
F5DBE900A664FCBF96102792069B15C0F2F8AE2000859DEAD1193C558A372
954825

λ_j:

5129212473B5BAC5A13C0F131C86DABB35C0FB99D0DA7638EB78C13FDF9CE
3FFBCCB184F4F32FF01AB031B01FAF6555FAFFE938174F929765D968DC6DA
3BA67D

x_Q:

69B083576981F45F3D4E8E3EF12B1E37C35A1AA17F8A79591EF868AB34E1C
F19982EAC7E460C560AFAB962997D2E25F8359B3AA4BBAABCDAFD18C523F1
F6F8F5

Focusing on the least significant word $x_j[0]$, $x_Q[0]$ and $\lambda_j[0]$,

$x_j[0] = 72954825 = 01110010100101010100100000100101$

$x_Q[0] = F1F6F8F5 = 11110001111101101111100011110101$

$\lambda_j[0] = DA3BA67D = 11011010001110111010011001111101$

In this example no reduction by the modulus p is required. If it was required, this could also be calculated by adding the two's complement of the least significant word of p, $p'[0]$, to $x_j[0] + x_Q[0]$. Now to predict the output of the operation $\lambda_j(x_j + x_Q)$, the least significant words are again focused on, i.e. $\lambda_j[0](x_j[0] + x_Q[0])$,

$x_j[0] + x_Q[0] =$ 648C411A

01100100100011000100000100011010

$\lambda_j[0](x_j[0] + x_Q[0]) =$ 55B6DD272FB0A5B2

1010101101101101101110100100111 0

0101111101100001010010110110010

Note that the least significant word of the result of $\lambda_j[0](x_j[0] + x_Q[0])$, 2FB0A5B2, is the least significant word in the entire produce $\lambda_j(x_j + x_Q)$ pre-reduction mod p.

296B2904FBAE4BAB67B26FAF9C29E9008E68E2E4AE4B86878F4C554412F4E

D11FB14AE2661A63750A71623D4380C4EFDDC340850C5EC2E6FD76DBD4B9A

96D80E70AAFD0656D7D6DF1FF9624F9E848C58F909B5EA1D8A4A3DAF14B42

C9EB9D50D58019BCCCDEAF768013F5926EFBB2F15AFCB681AA9958B6907E9

1D6E2FB0A5B2

This demonstrates that given known x_j, λ_j and unknown x_Q, output portions of the operation $\lambda_j(x_j + x_Q)$ can be deterministically calculated.

Appendix B

Numerical Examples of Fault Attacks

The purpose of this appendix is to present concrete examples of the fault attacks described in Chapter 6. Software was written in both Java and C to simulate the following examples.

B.1 Data Corruption Fault of η_G Pairing

This section describes a data corruption fault of the η_G pairing [41]. The fault attacks described here were detailed in Section 6.4.1. In each of the attack descriptions it is assumed that the point Q is the secret elliptic curve point and so the coordinates (x_Q, y_Q) are of interest.

B.1.1 Target Data: Cell l_0

The fault targets the cell l_0 in the last round of the Miller loop.

B.1.1.1 Data Values

Field parameters:

The implementation assessed is over the binary field $\mathbb{F}_{2^{271}}$, with reduction polynomial $f(z) = z^{271} + z^{58} + 1$ and elliptic curve $E(\mathbb{F}_{2^{271}}) : y^2 + y = x^3 + x$.

Input Points to $\eta_G(P_1, Q)$:

P:

(69E714DA0E9BF21C4E9D400F4ED644F8C80E983D60026CD2AEC39AB010CE
E129B34D,50882E348FD0133A0120F005BF65CB8C21A8837DAD9E5BD58FA9
CF1BBFC24AEDD0E4)

Q:

(7B92FE9FBAB4ED6F8C8902D159020B9BA8C7C01A9DF27FC05FA9036C9727
AB203CE9,7C71F818ABD7D6F408AB5935D6A114476B837BE1ADA14C49DD13
C071D1E044C15F44)

Input Points to $\eta_G(P_2, Q)$:

P:

(7C73C5C86B07887EFA2BB2DFE22C93195AD5F0FC34BAA9C84989F8A7BD0D
6645D795,2C45894D01BD49EE187D1BC0FF54C11097121FF1FC8EF6FE979F
01DC1A4344F03744)

Q:

(7B92FE9FBAB4ED6F8C8902D159020B9BA8C7C01A9DF27FC05FA9036C9727
AB203CE9,7C71F818ABD7D6F408AB5935D6A114476B837BE1ADA14C49DD13
C071D1E044C15F44)

Valid Pairing $\eta_G(P_1, Q)$:

[25113BDCDD9200B876451861074C193BB570A864612A13079C93D3D8F1D9
4166A70C,5339132DF97AEEFE6401F2C56B28DB23D467501C4EA6FB1DBDE4
7E63181B4FAB1043,54689B2C103FD9AD37E24464D37C43D4C869F7562D06
4195D3DEED7258C9DE8547D2,20BE058D44CFFA54C008EDED71D776D2AF63
EB1A7CAA41B0C6B101E4DE444DA4E01F]

Faulty Pairing $\eta_G(P_1, Q)'$:

[3259D5598613D936436A69A41E3A2AA039A08F5847C61EA5F869DA50EFF1
2396166,50A17A7FA301950ECE64F8DB3604FF5B5EA6959986F2BC1C904D5
965DA662AA5ED03,6236CC14DC63BCD5F3C238043CD0EEA2E35AAF0F91BE2
48DC415245A3D7979787D59,3D3C6FEE0A288253F9C26A0BA9586D14074EF
31DDA543D25BA0C9C3EF8885DA37444]

Valid Pairing $\eta_G(P_2, Q)$:

[48C59B1D5CC04E01F067D1A9E2132A1A311A9A8AE0C886E669B92D7DB8BB
8ACB0045,7424B7E7E4522C09B2791C021567009CE198C516D8535CC462BC
A95116DFE716BCF2,7CB5B888DFC3F8FA5CE11F914178D554EC3E227E952F
D3DF04E59613163373295EB6,4953CBB7EC080F64A0C0F4B804BDD43ACB1B
05DD598EEFE045E80810385E68857D01]

Faulty Pairing $\eta_G(P_2, Q)'$:

[5EE40B6945E2DCAEB19282AEF3CB2ED1A6C42CCC0C78AE0A3185F73DD9F0
8873D05F,558C27BA29E0A25497FA0ACCFC8B26ADC2BCBE5DE5255896FA89
B48903E1F5E2A98F,4486368D6D8DD1EA55AFF99C73DEAEC464CB058A2FDA
DB528AF85C51E03148F3C9B0,766E6FCEE9CD75B400017697750F8E3F0443

24CEE3F939E8E5DD2DC34D12219F5E2E]

B.1.1.2 Steps to the Attack

Step 1: Division of first faulty pairing by valid pairing.

$\eta_G(P_1, Q)'/\eta_G(P_1, Q)$:

[526A26DF60A7D8CEAF9C7F481559105A501F81052D761EA314E1463FD1E3
DB634A75,1B82B9CDE1182467ABDF2E66B6766FDAA3F006476F2ADED635A3
315198A8C52F19E3,6E798561BB4238F3F2ECF02657F31C172B391A85E229
D01A19D723A2C6261B75F6C6,32734DF92C0B70847B8771BE800E674C5D72
A797E92B79BD3FEFE59E7383BD30CAC7]

Step 2: Using the values generated from $\eta_G(P_1, Q)'/\eta_G(P_1, Q)$, namely N_0, N_1, N_2 and N_3, calculate data values relating to Equation (B.1). Note the derivation of this equation was discussed in Section 6.4.1.

$$A_1 x_Q + B_1 y_Q + C_1 = 0 \tag{B.1}$$

where

$$\begin{aligned} A_1 &= N_0 + N_1 + N_2\lambda + N_3 + 1 \\ B_1 &= N_2 \\ C_1 &= N_0\lambda + N_1 + N_1\lambda + N_2(x_A\lambda + y_A + \lambda) + N_3\lambda + \lambda \end{aligned}$$

$A_1 =$

360FF51525E2CFACEEDE9DE8CAFCEA2181FF04291A66008EEE9D6D29D241
C0DFDD57

$B_1 =$

6E798561BB4238F3F2ECF02657F31C172B391A85E229D01A19D723A2C6261
B75F6C6

$C_1 =$

68EB066FEF1ED42B48822A8D253B639F960DED9F29471AB9EA8861A351E14
19C2F93

Step 3: Division of second faulty pairing by valid pairing.

$\eta_G(P_2, Q)'/\eta_G(P_2, Q)$:

[EE53C25890FC56404EE992B7E4FB7B81D4318417D1F58C43C1B12B1046DD
D8A9D83,5E93DCE69B6B2A73B8C6520008C4E61DE68CF348EB61CEB73D5FA
A7D029CD7326E40,46B10926ACF3647E966F67D26E7E6FA5FCDE507236614
B9995C5783EA4FB8A1F5DF5,7C4D465B25B8D47EF4A4E0F3629EF8F9A1587
29F917810363F60B5769D80BF37D994]

Step 4: Using the values generated from $\eta_G(P_2, Q)'/\eta_G(P_2, Q)$, namely N_4, N_5, N_6 and N_7, calculate data values relating to Equation (B.2).

$$A_2 x_Q + B_2 y_Q + C_2 = 0 \tag{B.2}$$

where

$$\begin{aligned} A_2 &= N_4 + N_5 + N_6\lambda + N_7 + 1 \\ B_2 &= N_5 \\ C_2 &= N_4\lambda + N_5 + N_5\lambda + N_6(x_A\lambda + y_A + \lambda) + N_7\lambda + \lambda \end{aligned}$$

$A_2 =$

7F19B0BC5DCC269C04C746D042EF76F6E5A2E681805C9421197BA1549543C
6918C68

$B_2 =$

46B10926ACF3647E966F67D26E7E6FA5FCDE507236614B9995C5783EA4FB8
A1F5DF5

$C_2 =$

65584AA6420DEB696BC843F292ED2737EF0124BD14DC155FE6E6E835D1D57
9EA769D

Step 5: Solve for x_Q and y_Q. This can be performed using substitution as follows.

(i) Multiply Equation (B.1) by A_2 and Equation (B.2) by A_1 to get,

$$A_2A_1x_Q + A_2B_1y_Q + A_2C_1 = 0 \tag{B.3}$$

$$A_1A_2x_Q + A_1B_2y_Q + A_1C_2 = 0 \tag{B.4}$$

(ii) Add Equation (B.3) to Equation (B.4) to get,

$$A_2A_1x_Q + A_1A_2x_Q + A_2B_1y_Q + A_1B_2y_Q + A_2C_1 + A_1C_2 = 0$$

Therefore,

$$y_Q = \frac{(A_2C_1 + A_1C_2)}{(A_2B_1 + A_1B_2)}$$

and so

$y_Q =$

7C71F818ABD7D6F408AB5935D6A114476B837BE1ADA14C49DD13C071D1E04
4C15F44

(iii) Substitute y_Q into either of the original two equations to find x_Q. Therefore,

$x_Q =$

7B92FE9FBAB4ED6F8C8902D159020B9BA8C7C01A9DF27FC05FA9036C9727A
B203CE9

B.1.2 Target Data: Cell v_0

The fault targets the cell v_0 in the last round of the Miller loop.

B.1.2.1 Data Values

Field parameters:

The implementation assessed is over the binary field $\mathbb{F}_{2^{271}}$, with reduction polynomial $f(z) = z^{271} + z^{58} + 1$ and elliptic curve $E(\mathbb{F}_{2^{271}}) : y^2 + y = x^3 + x$.

Input Points to $\eta_G(P, Q)$:

P:

(69E714DA0E9BF21C4E9D400F4ED644F8C80E983D60026CD2AEC39AB010CE
E129B34D,50882E348FD0133A0120F005BF65CB8C21A8837DAD9E5BD58FA9
CF1BBFC24AEDD0E4)

Q:

(7B92FE9FBAB4ED6F8C8902D159020B9BA8C7C01A9DF27FC05FA9036C9727
AB203CE9,7C71F818ABD7D6F408AB5935D6A114476B837BE1ADA14C49DD13
C071D1E044C15F44)

Data used in round of Miller loop targeted by the fault:

Intermediate point $[i]P$ used in last round of line function.

$x_A =$

38F3F251F96093C28B2B82D9EB12D93BD4E5D0C123D851E1AC5A3BF6BD998DD40BA

$y_A =$

3EF89128D586F7F0082C10BD03D081549F29AC42A51C67B5D628271359CD30594C85

$x_C =$

69E714DA0E9BF21C4E9D400F4ED644F8C80E983D60026CD2AEC39AB010CEE129B34C

$y_C =$

396F3AEE814BE1264FBDB00AF1B38F74E9A61B40CD9C3707216A55ABAF0CABC463A8

$\lambda =$

18543BF3AAF8010C87D9C01123FC0BC6D29B6C25C6A783AEAC8247420DF6C25F9FA6

Output from two (valid, faulty) pairing executions:

Valid Pairing $\eta_G(P, Q)$:

[25113BDCDD9200B876451861074C193BB570A864612A13079C93D3D8F1D94166A70C,5339132DF97AEEFE6401F2C56B28DB23D467501C4EA6FB1DBDE47E63181B4FAB1043,54689B2C103FD9AD37E24464D37C43D4C869F7562D064195D3DEED7258C9DE8547D2,20BE058D44CFFA54C008EDED71D776D2AF63EB1A7CAA41B0C6B101E4DE444DA4E01F]

Faulty Pairing $\eta_G(P, Q)'$:

[4F38B1694502C41ADE928596BCA3CB37A65BEDAE81B3F3195986732B16EF

F50F39BC,1B72EBFAB9953ACFA9D15025F7C8A6CDFC0122F5B650B4F3555A
C7747EB25B4DE344,726CD21B5F6A060EBDB9DB744D603B1EB57C6E6A0EE9
5760DF4CBB863ACD238C1882,62FCBFD09FC2700B456C2EFC1E8F24389C52
8C43511A7E9CE4831CE0B155DC73DF9F]

B.1.2.2 Steps to the Attack

Step 1: Division of the valid pairing by the faulty pairing.

$\eta_G(P,Q)/\eta_G(P,Q)'$:
[53DD742B0B855A7233697CEB0A0E8C2C1433834EA6608F4F67FB26BEFB4E
AB27BC98,47C7B8A0729BA78C76FA41B95198B8BB24404B0CB14D17EFBA54
C18189167FAFB099,47C7B8A0729BA78C76FA41B95198B8BB24404B0CB14D
17EFBA54C18189167FAFB099,0]

Step 2: Solving the congruence.

$$v_0 = \frac{N_0 + 4N_1 + 3N_2 + 2N_3 + 1}{2N_0 + N_1}$$

Since we are dealing with binary fields, this means that any even multiples of elements in $\mathbb{F}_{2^m}$ will dissolve to zero. Therefore

$$v_0 = \frac{N_0 + N_2 + 1}{N_1}$$

which is evaluated to
v_0 =
1275EA45B42F1F73C21442DE17D44F6360C95827FDF01312F16A99DC87E94
A098FA4

By simply adding/subtracting the known value $x_C + 1$, the secret coordinate x_Q is found,

$v_0 + x_C + 1 = x_Q =$

7B92FE9FBAB4ED6F8C8902D159020B9BA8C7C01A9DF27FC05FA9036C9727A
B203CE9

B.2 Sign Change Fault Attack the of Weil Pairing

This section describes a data corruption fault of the Weil pairing [117]. The fault attacks described here were detailed in Section 6.4.2.

B.2.1 Data Values

Field parameters:

The implementation assessed is over the prime field $\mathbb{F}_p$. The elliptic curve is $E(\mathbb{F}_p)$: $y^2 = x^3 + ax + b$ where

$p =$

C8DF90FD89D7A9827ED3A42409D36AED13BA334D9E716169FC674B6CB2F98
E407A50E9E0FBDC15512D1F74A679C08D3AC42F292C8A95DF30758293235E
04739B

$a =$

C8DF90FD89D7A9827ED3A42409D36AED13BA334D9E716169FC674B6CB2F98
E407A50E9E0FBDC15512D1F74A679C08D3AC42F292C8A95DF30758293235E
047398

$b =$

491224746BC3C29BAB0157FE84580BB3E53912605012B88BCCEFE240FB679
E4605EABBBBA9A642BEA71A3840107915913F998E71662C2501A2E35122C5
69BF68

and the order r =

800000000000000000000000000000020001

Input Points:

P:

(C27938C41E38949CA9177135C99374D27659D9489955F65F312EEF10375C
78CAE2348E0E8E7B32D5021E181CABC42EFE30D44038950F23E7F49980D26
54C09B8,59F5C3910EB567CDB4B16BB2A0B2A8DFEA3ED92A64A4524789057
6568B779A846ECBC9FEA150CD492223836EE4796B23751D547B4DC06CD48E
230DCAE43632DA)

Q:

(B1F0D753A62BC0458E230371B2E663321CEDC5DC0BBBFA88B16BB4A54931
0D7297AE3F90F87876C63D898891085FDB29C61005E94C85DB372879FA953
420F48B,A747D64825676DB1176A4A2CF79EAD209525FBFFE008749CEB045
8C8433501979524635D3436CE5D232FF1CF122F01B659E36CA0EFB1AED331
8EE4B32FD8423C)

Data used in round of Miller loop targeted by the fault:

Intermediate point $[i]P$ used in last round of the line function.

x_A =

3A385AA26F93FC9279B2A69C326E3845C91302AD818122EA21480D53034A7
A81A60C46778707619423A3D0299A18F29C35692AC0A4CE1923235FD5A0DB
1E0584

y_A =

71BD2AB938A12B43B67C6A39C9D0134558B6E7CBA14A0328C0C238D3172F6
EC5C36E53E79EA5EF348C67DE4452AF353F9DCE41DE2CC362414DEA4E6621
100602

$\lambda_i =$
A4D9B86BCB52F6896909EF178128D2EF55AFA2DCDD9035C90ABD58C0A88AC
3A0E16A910DA759664F9BBE7BCF396842A134ECA15D5390EFFE2B9BD52D29
AAF28C

Output from two (valid, faulty) pairing executions:

Valid Pairing $\omega_D(P,Q)$:
[66192B7F5D59DF1CB6238052468D262D3EEE879E056FD7A7B52B4181C58A
FAD1486AC1DAEA498CC73943159C0F161CD3426DBE36FDA84D0DD520BF237
CBD326E,9804ED2E2F5FEF2CD13165A2FAA44040BE921F98C1E7036EE9CCB
57F324313B60C4D8B903646F32607F13B4F31C6AB519E0EA18926B8B39E3A
3DD93861E9D781]
Faulty Pairing $\omega_D(P,Q)'$:
[7414AEDA4113AF4A59A41591901EEA09C81A620153CAE069730F5616A14A
484BD9D4D5FE4515A9E19E11F8D6F8CDFC1AD5FA2A7C59B1D3C7F68D887C1
8A9DC2,79AF68D3AB50ED5D9628A79EDC59EC9FF820AD7D3FB38C6713576F
74AED35C28454C677119A4D170E4700AEA9F4F8293EAF531E770FE25BDF19
E08E8780C630D]

B.2.2 Steps to the Attack

Step 1: Division of the valid pairing by the faulty pairing.

$\omega_D(P,Q)/\omega_D(P,Q)'$:

[52002F3A6C90C9AE9BB0D7B6FB0C5B56C4DFD86DE24C6A6F95BC40E86560
77631CD6BD666596A20D82063685940CC77BBA9CCCFAD034283AA8F04F57C
A379062,7412C550674512E2E168850B67D36E60EA9C2B053FF19985868FC
F4E8DFCAC40A7B97329A7AE2A11EECCE357DAF09D8C260AFCE63346022955
CEBD2147135E9]

Step 2: Calculate the square root of this value.

[1D7B3A07709ACC56F84A24ABB88EBE62FD4580542E612739DF8BC408F2A9
6075D1C2689F2C8CB86F0AB7F794EBADB0A28AF982B437206E38AE4C66D49
924F682,A1FECD9DFEAE289E71CA3017312EA1EC55B4E0E816978002ADB6A
E8483888D51C83A2BED128C1341F044EF0D228A592D68F829AF820C40F7F1
04227CE9BB3AF5]

This exploits the relationship

$$\sqrt{\frac{\omega(P,Q)}{\omega(P,Q)'}} = \pm\left(\frac{[\lambda_i(x_A+x_Q)-y_A][-y_Q]}{[\lambda_i(x_A+x_Q)-y_A][y_Q]}\right)$$

where two roots ($\pm\sqrt{\ }$) will be produced. Here we obtain the correct root. Let

$N_R =$

1D7B3A07709ACC56F84A24ABB88EBE62FD4580542E612739DF8BC408F2A96
075D1C2689F2C8CB86F0AB7F794EBADB0A28AF982B437206E38AE4C66D499
24F682

and

$N_C =$

A1FECD9DFEAE289E71CA3017312EA1EC55B4E0E816978002ADB6AE8483888
D51C83A2BED128C1341F044EF0D228A592D68F829AF820C40F7F104227CE9

BB3AF5

Step 3: From this relationship we can generate two equations which relate to the grouping of real and complex values. We will just show the real here:

$$(\lambda_i - N_R\lambda_i)x_Q + N_C y_Q + \lambda_i x_A - y_A - N_R\lambda_i x_A + N_R y_A = 0$$

We can calculate the coefficients of this equation since they are composed of known/computable values.

$(\lambda - N_R\lambda) =$

64DCD4C10F7816EF8D7C389BA1709DA605EE81C5F67339764546C2C4B0975
6DAFC5119AA299957AE05AE8FEB5A64DFBE801E0073DB161053319B6B6CD5
256A1

$N_C =$

A1FECD9DFEAE289E71CA3017312EA1EC55B4E0E816978002ADB6AE8483888D51C83A
2BED128C1341F044EF0D228A592D68F829AF820C40F7F104227CE9BB3AF5

$\lambda_i x_A - y_A - N_R\lambda_i x_A + N_R y_A =$

895341DAD464B48011914A33021B77238FDEF077FBF59E124C007043AA5AB
53C223D7E0383761712E4ED4A87AF08242A31A94AC7D7EFB4E7F2F1AC3504
924DC9

To solve this we will need another equation with the same unknowns. However since there exists a relationship between x_Q and y_Q, one equation will suffice.

Step 4: Substitute $\sqrt{x_Q^3 + ax_Q - b}$ for y_Q. This gives a cubic equation:

$$x_Q^3 + A_0 x_Q^2 + B_0 x_Q + C_0 = 0$$

where

$$
\begin{aligned}
A_0 &= -\frac{(\lambda - N_R\lambda)^2}{N_C^2} \\
B_0 &= \frac{(a - 2(\lambda - N_R\lambda)(x\lambda - y - N_R x\lambda + N_R y))}{N_C^2} \\
C_0 &= -b - \frac{(x\lambda - y - N_R x\lambda + N_R y)^2}{N_C^2}
\end{aligned}
$$

and so

$A_0 =$

6802A674045A31776FFB8CFBB67AD446CE7AA807B8FC47B3DFA77C4FA8BC5
879C0E05F5C00806413631186314B64E930BC5D05EF35D81B095AAB80B16D
25C6E2

$B_0 =$

9960E747477925F838D6E65391C97CC2986BED3699CC28748FB80414AB1B7
09E47E5267C0DE54B5084B4EAE935C88F09741189C3AF1B9AE7DE82B31D88
E72D88

$C_0 =$

AF3B578B224A5C538C6E21DAA2A0319C890377BCB93057B0D2BEF9965F94A
DB1276D26C9151C4A7FEE8BC76CBDEF24D0CF80F32EB1F04BCF1AD4B3D149
59BE48

Step 5: Solve with Cardano's Method. Given A_0, B_0, C_0 and the modulus p (which are all computable values by an adversary), solve for x_Q. The roots that our Cardano method produces are

root 1 =

B1F0D753A62BC0458E230371B2E663321CEDC5DC0BBBFA88B16BB4A549310
D7297AE3F90F87876C63D898891085FDB29C61005E94C85DB372879FA9534

20F48B

root 2 =

75C70FD2C28F0367D32F173281F4A611FF2DB79FAE53C328A1AC4B6379AD2

11708183462E800A5B7AB5AA5BD6A3D2597356877BA46BFD955EB8462D0CE

4B565B

root 3 =

2049460A69A5DE02C59A0A82850F84F3CDE4117C9D6BD6EC60F1A80FA5895

7D93FB007216BEAA110E4934CD357F3083D088CEC64C0DEECA7C5B482F4C7

6D56E

where the first root corresponds to our secret coordinate x_Q.