Differential Fault Analysis on Smart Cards using Elliptic Curve Cryptography

Spyridon Antakis

Eindhoven University of Technology
Department of Mathematics & Computer Science
Den Dolech 2, P.O Box 513, 5600 MB Eindhoven
Email: s.antakis@student.tue.nl

March 22, 2009

Abstract

Elliptic Curve Cryptography (ECC) can be easily implemented inside a smart card, demonstrating great level of security and computation efficiency. However, there are still possible threats for such tamper-resistant devices, as an attacker can take advantage of the existence of non-evaluated hardware weaknesses. Differential Fault Analysis (DFA), is one of the side channel techniques which can be used for attacking this kind of hardware and successfully reveal useful secret information, such as the private key. This paper, presents an outline of the elliptic curve cryptosystem usage on smart cards and focuses on how DFA attacks can be applied on smart cards using ECC.

Keywords: *Elliptic Curve Cryptography (ECC), Differential Fault Analysis (DFA), elliptic curves, public-key cryptography, tamper-resistant, fault attacks.*

1 Introduction

Nowadays, many commercial applications aim to provide a high-level security, by using tamper-proof devices such as smart cards. Inside a smart card, a designer is able to choose between different kinds of cryptographic systems, in order to provide security features. Nevertheless, a typical smart card has several characteristics, such as low computational processor power and small memory size (e.g. 32 KB), that constitute an important drawback for applying strong cryptographic algorithms.

Elliptic Curve Cryptography (ECC) seems ideal to handle these obstacles, as in comparison to other public-key cryptography schemes (e.g. RSA, DSA) is able to deliver a significantly smaller key size by providing equivalent level of security (A.1 [1] [2]). This results in faster computations, lower power consumption, as well as memory and bandwidth savings. Therefore, the time needed for an elliptic curve cryptosystem to generate a key pair is less, that even a smart card with limited resources can function efficiently. However, no theoretic proof is known for the difficulty of the *Elliptic Curve Discrete Logarithm Problem (ECDLP)* on which EC's security is based (see 2.1), therefore the adoption of such cryptosystems in the industry is not so wide.

Public-key cryptography schemes require each party to have a key pair. A *private key*, which must not be disclosed to another user and a *public key*, which may be made available in a public directory. The two keys are related by a hard one-way function, so it is computationally infeasible to determine the private key from the public key. The secure storage of the private key is critical and vital to the security of the cryptosystem. Private keys are often stored in software with password protection, however a storage in a hardware token such as a smart card, would be ideal for preventing direct access or tampering.

In 1997 [3], a new kind of attack called *Differential Fault Analysis (DFA)* was discovered and successfully applied to tamper-proof devices that were implementing popular cryptosystems (e.g RSA, DES). This paper, performs a deep survey on DFA attacks that can be applied on smart cards which are using *Elliptic Curve Cryptography (ECC)*. In section 2, a brief discussion about the functionality of ECC inside a smart card is made. In section 3, an introduction to DFA is given and several possible methods for creating *faults* are described. After that, a detailed summary of known DFA attacks

on elliptic curve cryptosystems follows and potential countermeasures are reported. Finally, the paper concludes and evaluates the importance of DFA threats and whether they constitute a drawback for the commercial adoption of elliptic curve cryptosystems on smart cards.

2 The EC cryptosystem

An *elliptic curve cryptosystem*, is a system that uses a public key encryption technique based on elliptic curve theory and creates faster, smaller, and more efficient cryptographic keys. This cryptosystem, is able to generate keys from the properties of the elliptic curve equation (A.2 [4]) and perform a cryptographic operation, comparable to all the other traditional public key encryption methods. In the following subsections, the steps that an elliptic curve cryptosytem performs are presented and a certification system design based on elliptic curves is described.

2.1 Cryptographic steps

According to [1], a cryptographic operation that uses elliptic curve theory, includes 5 basic steps:

- *Step 1:* A curve $\mathcal{C}$ that is produced using the simple[1] equation form, is chosen over a finite field $\mathbb{F}_q$.
- *Step 2:* A point P that belongs to $\mathcal{C}$ is found, such as $order^2(P) = t$, where t constitutes a large prime number.
- *Step 3:* The curve $\mathcal{C}$ and the point P are considered known to the public and they can be shared by multiple users.
- *Step 4:* The private key is an integer d, such that $1 < d < t$.
- *Step 5:* The public key P_k is a point in the curve $\mathcal{C}$, where $P_k = dP$.

The security of such a system relies on the *Elliptic Curve Discrete Logarithm Problem (ECDLP)*, which states the following:

"Let $\mathcal{E}$ be an elliptic curve over a finite field $\mathbb{F}_q$. Suppose $\boldsymbol{P}$ is some point of $\mathcal{E}(\mathbb{F}_q)$ and let $\boldsymbol{Q}$ be a point in group $\boldsymbol{\{O, P, 2P, 3P, \ldots\}}$. Find an integer $\boldsymbol{t}$ such that $\boldsymbol{Q = tP}$. "

[1]Simplified form: $y^2 = x^3 + ax + b$.

[2]order(P) is the smallest t integer that satisfies the equation $tP = \mathcal{O}$, where $\mathcal{O}$ the point at infinity.

It is widely believed that the elliptic curve discrete logarithm problem is computationally hard to solve, when the point P has large prime order. Up to now, most of the known methods for solving the ECDLP, are showing *exponential* or *sub-exponential* complexity. Thus, based on this problem various kind of cryptographic operations can be executed, such as *encryption/decryption*, *key-exchange* and *use of certificates*. In the next subsection, a certificate system design that uses such cryptographic operations is discussed.

2.2 A certification system design

Though this document mainly deals with *fault attacks*, this section briefly presents to the reader a certification system on smart cards, implementing ECC. Through this mention the reader will be able to understand in a better way the sections to follow and also will be given a clear picture of the usage of elliptic curves in real life applications.

In [1], a possible certification system design based on an elliptic curve cryptosystem is described. The system design supposes that there is Certification Authority (CA) and a smart card user. The certification procedure is divided into 2 main parts: i) *Request for a certificate* and ii) *Verification of a user*. Both, are shortly described below.

2.2.1 Request for a certificate

- *Step 1:* Bob who wants to apply for a certificate refers to the CA in person. Then a verification of his identity is performed by the CA.
- *Step 2:* Bob has to type his personal information and password into the computer directly.
- *Step 3:* The CA stores the data into the computer temporary.
- *Step 4:* Smart card randomly generates the key pair, which is constituted from a private key s and a public key $S = sG$. This way, the private key is secured inside the smart card, as it is never left the device.
- *Step 5:* The smart card sends the public key (a EC point) to the CA server.
- *Step 6:* Combining the user's information and the public key, the CA creates the certificate and signs it, using its own private key.
- *Step 7:* Thereafter, CA copies the created certificate inside the smart card.

2.2.2 Verification of a user

- *Step 1:* Bob presents his smart card that contains the certificate to the terminal.
- *Step 2:* The terminal verifies the certificate using CA's public key.
- *Step 3:* The terminal randomly generates an integer x and requests from the smart card to sing it, using Bob's own private key.
- *Step 4:* Bob's smart card signs x and sends it back to the terminal.
- *Step 5:* The terminal verifies it, by using Bob's public key extracted from the certificate.
- *Step 6:* The terminal will only accept Bob if he passes the verification.

The advantage of storing the private key inside the smart card without having the ability to leave the device, is that Bob can now prove that he knows his private key without revealing it. This approach is known as *zero-knowledge-proof* and it can prevent the unauthorized access. Suppose, that Eve creates a fake smart card in order to pretend Bob. This attack will fail because Eve cannot pass the zero-knowledge-proof, as she does not possess Bob's private key. However, as it will be demonstrated in the following sections, there are techniques that Eve could use to disclose the private key. Thus, the security countermeasures of systems based on secure information storage on hardware should be re-evaluated, in order to defend against such techniques.

3 Differential Fault Analysis

Researchers showed that the occurrence of faults is a serious threat to cryptographical devices. In 1997, they introduced a new active side channel attack, called *Differential Fault Analysis (DFA)*. The principle of DFA attack is based on creating faults and unexpected environmental conditions into cryptographic designs, in order to disclose their internal states without focusing on the algorithms weaknesses. [3] [7]

As a *fault attack*, we define a complete approach that if applied to a tamper-resistant device, it is able to return secret data. An attacker may run the device several times while inducing faults into memory cells or other structural elements of the device. In this way, he causes the attacked device to malfunction and to output a faulty result, which is used to derive secret information. More precisely, if an attacker enforces some sort of physical stress on the smart card, he can induce faults into the circuitry or memory. These faults become manifest in the computation as if an error occurs, a faulty result is computed. Thus, if the computation depends on some secret key, a comparison between correct data and faulty data may allow to conclude facts about the secret key.

Usually, in order to induce faults a direct access and contact with the chip is needed. Depending on the precision that the injections can achieve and the equipment that is used for injecting the faults, the methods can be separated into *invasive* and *non-invasive*. [9]

Non-invasive fault injection methods do not modify the package of the device. Faults are provoked by manipulating the conditions the device runs. This can be done by injecting peaks in to the clock or the power supply, which is called glitch or spike attack, respectively. Another possibility is to change the temperature over the specified conditions. These methods are inexpensive and easy to perform, however only a limited precision can be achieved, as those attacks impact on the whole chip at once.

Invasive methods establish direct electrical contract to the surface of the chip. Thus, the device itself can be modified. These attacks need very expensive equipment like a probe station and a laser cutter.

In fact, we could say that every *fault attack* method can be described by a measure of *invasiveness* indicating the amount of tampering necessary. Moreover, some attacks allow complete control where the fault injection happens, whereas others do not. Therefore, an attacker may need a direct access to the tamper-resistant device and special equipment, in order to apply a fault attack.

3.1 Methods for injecting faults

Errors almost never occur naturally, as software is usually rigorously tested and hardware is expected to be faultless. As a result of that, there is no protection against errors and any malfunction is left uncontrolled. Smart cards function in a hostile environment, thus an attacker is possible to apply techniques in order to generate faults. In practice, there are various methods for inducing faults on tamper-resistant devices like smart cards and the most commonly used are presented below. [10] [11]

3.1.1 Temperature

Typically, electronic equipment only works reliably in a certain range of temperature. If the outside temperature is too low or too high, faults occur.

The circuit manufacturers define upper and lower temperature thresholds within which their circuits will function correctly. The goal here is to vary temperature until the chip exceeds the thresholds boundaries. When conducting temperature attacks on smart cards two effects can be obtained: the random modification of RAM cells due to heating/-cooling and the exploitation of the fact that read and write temperature thresholds do not coincide in most non-volatile memories (NVMs). Thus, by tuning the chips temperature to a value where write operations work but reads do not or the other way around, a number of attacks can be mounted.

3.1.2 Power Spikes

A smartcard is a portable device without its own power supply. Hence, it always requires a smart card reader providing it with power in order to work. This reader can be easily replaced by an adversary with laboratory equipment, capable of tampering with the power supply. It has been specified by standards, that a smart card must tolerate a certain variation on the power supply of 10% of the standard 5V voltage. However, if the variation is significantly higher than 10%, the card does not work properly anymore. In fact, short massive variations of the power supply, which are called *spikes*, can be used to induce errors into the computation of the smart card. Spikes allow to induce both memory faults as well as faults in the execution of a program. Equipment requirements are totally depend on the type of the spike to be generated, but are not necessarily expensive.

3.1.3 Clock Glitches

In a similar way to the power spikes, it is sometimes possible to have the clock speed disturbed. Smart cards do not create their own clock signal and they are usually provided with a 3.5 MHz signal. Although modern high-end smartcards use a randomized clock, they only randomize the clock signal provided by the external card reader. Smart cards are required to tolerate a voltage variation in the clock signal and must also work properly with deviations of clock rise and clock fall times of 9% from the standard period clock cycle. Since the adversary may replace the card reader by laboratory equipment, he may provide the card with a clock signal, which incorporates short massive deviations from the standard signal, which are beyond the cards' tolerance boundaries . Such signals are called *"glitches"*. Clock-signal glitches can be used to both induce memory faults as well to cause a faulty execution behavior and they are simple, cheap and noninvasive.

3.1.4 Optical Attacks

Using focused light with specific wavelengths, we can change the flip-flops in a memory cell. By doing this, it is thus possible to change or modify the memory through the photoelectric effect. These attacks require light to be able to reach the chip, and therefore any protective coating needs to be removed. If a smart card is unpacked, in a way that the silicon layer is visible, it is easy to use a *laser cutter* or focused *UV light* in order to destroy individual structures of the chip. This allows to induce a great variety of destructive faults. Modern green or red lasers can be focused on relatively small regions of a chip, such that faults can be targeted fairly well. However, such *light attacks* are usually not usable for a systematic attack, since they cannot be targeted in a high precision in order to change selected bits.

In 2002, a new attack called *optical attack* was introduced. This attack allows to set or unset individual chosen bits of an SRAM memory cell, if the chip is unpacked in order to allow visual contact with the memory cells. In general, optical attacks can be done relatively cheaply with simple equipment, and also they can be very precise, effecting only a single bit.

3.1.5 Electromagnetic Attacks

By creating a strong electromagnetic source near memory the ions representing the states in the memory are moved around, and thereby the memory is disturbed. It is claimed that this gives a fine control of exactly what bit needs to be controlled. Moreover, this attack can be performed relatively cheap and it is a non-invasive attack. It only requires to be near the processor and there is no need for visual contact with the chip as it can be applied from outside.

4 Fault attacks on elliptic curve cryptosystems

In 2000, I. Biehl et. al. extended the ideas for differential attacks used on the RSA cryptosystems. A proposal for a similar way of attacking elliptic curve cryptosystesms was made and an investigation using DFA techniques for revealing the secret key of an ECC smart card implementation was performed. [7] [12]

In all the attacks, the authors assumed that the

cryptographically strong[3] elliptic curve is public and known to the attacker as part of the *public key*. Moreover, they considered that the *secret key* d is stored inside a tamper-proof device, unreadable for outside users and that on an input of some point P which belongs to the chosen elliptic curve, the device computes and outputs the point $d \cdot P$. Finally, they supposed that an attacker can have access to the smart card and is able to compute $d \cdot P$ for any arbitrary input point P'.

The main idea of the attacks, is focused on the fact that possible faults at the begging of the multiplication could leak secret information. In practice, if an attacker inserts a point P' that lies on a different curve from the one being used by the smart card, then he is able to decrease the *elliptic curve discrete logarithm problem* into smaller problems.

In this section, two attacks are discussed in detail. An in depth analysis of each attack is given and the steps that must be followed in order to reveal the secret private key are presented.

4.1 No correctness check

This attack does not depend on the creation of a fault, but on the assumption that there are no *input/output* checks. If the smart card does not explicitly check whether an input point P or the result of the computation is really a point on the cryptographical strong elliptic curve $\mathcal{E}$, then an attack is feasible. Usually, a well-designed system has defensive mechanisms for invalid points. Nevertheless, such a *bad implementation* is possible to occur in practice.

The expected functionality of an elliptic curve cryptosystem, is to take as an input a point $P \in \mathcal{E}$ and output the product $d \cdot P \in \mathcal{E}$ (fig. 1). In this way, an attacker is infeasible to disclose the secret key d as the computations inside the smart card are based on a *strong* elliptic curve.

Figure 1: *Expected functionality*

Supposing now, that the designers of a smart card forgot to implement *input/output* point validation mechanisms, it would be possible for an attacker to insert as input a point P' that does not belong to the strong elliptic curve $\mathcal{E}$ and take as an output the product $d \cdot P'$, which also does not belong to $\mathcal{E}$ (fig. 2). Taking advantage of this lack of *input/output* check, someone can determine the *secret key* d by using another *non-strong* elliptic curve $\mathcal{E}'$.

Let $\mathcal{E} = \{a_1, a_2, a_3, a_4, a_5\}$ the cryptographical strong elliptic curve, that the smart card implements. We choose to give as an input a point P' that belongs to a different elliptic curve $\mathcal{E}'$.

Figure 2: *No point validation*

The input pair $P'(x, y)$ is carefully chosen such that $a'_5 = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x$. Then, the new tuple $(a_1, a_2, a_3, a_4, a'_5)$ defines the elliptic curve $\mathcal{E}'$ which is selected in such a way, that its order has a small divisor $r = ord(P')$. Based on *Theorem A* (B.1), it holds that the output of the smart card with input P' would be $d \cdot P' \in \mathcal{E}'$ (fig. 3). [12]

Figure 3: *First attack*

Therefore, we end up with a discrete logarithm problem in the subgroup of order r generated by $P' \in \mathcal{E}'$, namely given points P' and $d \in P'$ that belong to $\mathcal{E}'$, find $d \ mod(ord(P'))$. Repeating the procedure with a different choice of P' we can create a system of equations, like the following one:

$$d \equiv d_1 \ mod(r_1)$$
$$d \equiv d_2 \ mod(r_2)$$
$$d \equiv d_3 \ mod(r_3)$$

The above system, can be easily solved using the *Chinese Remainder Theorem* (see B.2) and thus the value of the secret key d is computable. This technique, is even more efficient if we do not choose in advance the point P', but the curve $\mathcal{E}'$ and then compute the point P'. [12] There are several methods for constructing such a *non-strong* elliptic curve $\mathcal{E}'$, but such a description is out of the scope of this paper and therefore the reader should refer to [13] for more details.

[3] An elliptic curve is called cryptographically strong if the underlying ECDLP is considered to be computationally intractable for the application in use.

4.2 Fault at the beginning

If the smart card is designed in such a way that checks whether the given input point is a point within the group of points of the *cryptographically strong* elliptic curve $\mathcal{E}$, then the attack that was described in subsection 4.1, is no more applicable (fig. 4).

However, if the smart card does not check the output point, we are still able to enforce a fault inside the smart card at some precise moment at the beginning of the multiplication procedure. In order for someone to apply a fault on the smart card, one of the several methods that were presented in section 3 can be used, depending on the needed result.

Figure 4: *Performing input check.*

In this attack, the main objective is to generate *a single bit* fault exactly after the input check and before the start of the multiplication process (fig. 5). Such a fault, is possible to be generated by applying an *optical fault attack*, which as already mentioned in section 3.1.4, it can be very precise and capable of effecting only a single bit.

Figure 5: *Second attack*

Lets suppose that a single bit fault is produced exactly after the input test is finished. Then, the smart card computes internally with a pair P' which differers in exactly one bit from the input point P and therefore if it does not check whether the output is a point on $\mathcal{E}$, it outputs $d \cdot P'$. From theorem A (see B.1) we deduce that $d \cdot P'$ lies on the same elliptic curve $\mathcal{E}'$ as P'. Thus, we can determine a_5' such that the output $d \cdot P'$ satisfies the curve equation with coefficients $(a_1, a_2, a_3, a_4, a_5')$. If these coefficients define an elliptic curve $\mathcal{E}'$, we have reduced the original discrete logarithm problem on $\mathcal{E}$ to a discrete logarithm problem on $\mathcal{E}'$. [12]

The point P', will differ only one bit from the point P. An attacker can check all the possible candidates P' and whether they belong to the elliptic curve $\mathcal{E}$. If the point P' indeed lies on $\mathcal{E}'$ then he has only to solve the discrete logarithm problem on $\mathcal{E}'$. First, he will compute $ord(\mathcal{E}')$, which is the number of points on $\mathcal{E}'$ and it can be easily be computed by the use of algorithms for point counting. If $ord(\mathcal{E}')$ has a small divisor r, then the attacker can solve the discrete logarithm for the points $(ord(\mathcal{E}')/r) \cdot P'_{\mathcal{E}'}$ and $(d \cdot (ord(\mathcal{E}')/r)) \cdot P'_{\mathcal{E}'}$. This will give an equation $d \equiv c \ mod(r)$ for some value of c. Repeating the above procedure with different divisors r, the attacker can create an equation system again and compute the *secret key* d using the *Chinese Remainder Theorem* (see B.2).

4.3 Countermeasures

The attacks that were described in this paper, depend on the ability to disturb a point P on the curve $\mathcal{E}$, in order to generate an ordinary pair. Thus, a smart card which is not designed to make *input* and *output checks* is left unprotected and vulnerable. Usually, most of the elliptic curve cryptosystems that are implemented on tamper-resistant devices check the input points for correctness, but ignore the output points. However, it is very important that smart cards check also the computed and output points and in case these do not satisfy the proper conditions, not to let them leave the device. In this way, the possibility of secret information leakage is decreased and the attacks that were presented in the previous sections, are ineffective.

An alternative way for protecting smart cards for fault attacks is to include *active protection*. Active protection encompasses mechanisms that check whether tampering occurs and take countermeasures like having the smart card locked. There are several different protection mechanisms that can be applied to a smart card, such as *light detectors*, *supply voltage detectors*, *frequency detectors* and *hardware redundancy*. Light detectors are able to detect changes in the gradient of light and therefore defend against optical attacks. Supply voltage detectors can react to variations in the supply voltage and can ascertain that only a tolerable voltage is used, protecting this way the smart card from spike attacks. Frequency detectors can ensure that the operation speed is constant and thus can prevent glitch attacks. Hardware redundancy is performing recalculations a number of times by splitting the calculation, using checksums or by doing the calculation in different ways and afterwards verifying the correctness. A great variation of options exist here, depending on the allowed performance hit and the transistors used. A protection against injected faults is accomplished by trying to ensure that only valid data are output.

It is obvious, that various measures can be implemented by the industry in order to provide tamper-proof chips. Efficiently designed actions can be taken to either prevent or at least make fault injection harder. Nevertheless, security is not the only concern of the industry, as factors like *computation performance* and *implementation cost*, are also under consideration. Therefore, sometimes the adoption and the enforcement of protection measurements is sacrificed, leading to feasible fault attacks and on the disclosure of the secret information.

5 Conclusions

Elliptic Curve Cryptography (ECC) is an efficient and attractive technique that provides a high level of security. Although the particular theory is more than one decade old, cryptosystems based on elliptic curves are not yet widely adopted. However, ECC offers the computational and bandwidth advantages at comparable security and this is the reason that the industry has come to realize the potentials of elliptic curves alternative. Elliptic curve cryptosystems have already been introduced on tamper-resistant devices, such as smart cards and as it was expected, this has brought new security concerns.

Fault attacks can be applied through several methods by an attacker and a disclosure of secret information that are stored inside a smart card, is possible. Nevertheless, if security countermeasures are implemented during the design phase of a smart card, then DFA attacks can be ineffective and bound. The security features that a smart card includes would determine the level of security that the device provides against fault attacks. Usually, depending on the final usage of the smart card, different factors are taken into account before implementation. Factors such as cost, performance impact, computation complexity and security mechanisms are evaluated from the designer and decisions are made concerning the features provided to the card.

In conclusion, we can say that fault attacks constitute a possible threat for smart cards that are implementing elliptic curve cryptosystems. However, the existing countermeasures are capable enough for defending against such attacks, but are not always applied. The different priorities that the industry defines as design objectives, could easily neglect the need for security countermeasures.

References

[1] *Joseph K. Liu, Victor K. Wei, C. Siu Roy L. Chan and T. Choi* "Multi-Application Smart Card with Elliptic Curve Cryptosystem Certificate", EUROCON'2001, Trends in Communications, International Conference on, vol.2, pag. 381-384, (2001).

[2] *RSA Laboratories* "A Cost-Based Security Analysis of Symmetric and Assymetric Key Lengths", `http://www.rsa.com/rsalabs/node.asp?id=2088`, Accessed on: 24/02/2009.

[3] *E. Biham and A. Shamir* "Differential Fault Analysis of Secret Key Cryptosystems", Proceedings of CRYPTO'97, Springer, pag. 513-525, (1997).

[4] *Henk C.A. van Tilborg* "Fundamentals of Cryptology", Eindhoven University of Technology, The Netherlands, Kluwer, pag. 213-236, (2000).

[5] *V. Miller* "Use of Elliptic Curve Cryptosystems in Cryptography", CRYPTO'85 Proceedings, Springer-Verlag, pag. 417-426, (1986).

[6] *N. Koblitz* "Elliptic Curve Cryptosystems", Mathematics of Computation, pag. 203-209, (1987).

[7] *D. Boneh, R.A. DeMillo and R.J. Lipton* 'On the Importance of Checking Cryptographic Protocols for Faults', Proceedings of EUROCRYPT'97, Springer, pag. 37-51, (1996).

[8] *M. Otto* "Fault Attacks and Countermeasures", University of Paderborn, Department of Informatics and Mathematics, (2004).

[9] *J.M. Schmidt* "Differential Fault Analysis", Secure Business Austria and A-SIT, (2008).

[10] *H. Bar-El, H. Choukri, D. Naccache M. Tunstall and C. Whelan* "The Sorcerers Apprentice Guide to Fault Attacks", Proceedings of the IEEE, vol. 94, pag. 370-382, (2006).

[11] *K.O. Gadella* "Fault Attacks on Java Card", Eindhoven University of Technology, Department of Mathematics and Computer Science, (2005).

[12] *I. Biehl, B. Meyer and V. Mller* "Differential Fault Attacks on Elliptic Curve Cryptosystems", Advances in Cryptology - CRYPTO'00, Springer, pag. 131-146, (2000).

[13] *IEEE P1363 Draft* "Standard Specifications for Public Key Cryptography", IEEE Group, Version D13, (1999).

Appendices

A Elliptic Curve Cryptosystems

In the 1980's , V. Miller and N. Koblitz proposed a new public key cryptosystem using a group of points on an elliptic curve. The points on an elliptic curve $\mathcal{E}$ over a finite field $\mathcal{K}$ form a commutative group. The addition operation is really easy to implement and the the discrete logarithm problem in this group is believed to be very difficult. [5, 6]

A.1 Comparison of cryptographic schemes

MIPS years to break	RSA/DSA Key size (bits)	ECC Key size (bits)	RSA/ECC Key size ratio
10^4	512	106	5:1
10^8	768	132	6:1
10^{11}	1.024	160	7:1
10^{20}	2.048	210	10:1
10^{78}	21.000	600	35:1

ECC Key size (bits)	RSA Key size (bits)	Time to break	Machines	Memory
112	430	Less than 5 min.	105	Trivial
160	760	600 months	4300	4 GB
192	1020	3 million years	114	170 GB
256	1620	10^{16} years	16	120 TB

Figure 6: *"Time needed for breaking different public key schemes. [1, 2]"*

MIPS = Millions Instructions Per Second.

A.2 Introduction to Elliptic Curve

The elliptic curve equation (*Weierstrass equation*), is defined as follows [4] :

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where *a, b, c, d* and *e* are real numbers satisfy some conditions which depends on the field it belongs to, such as *real number* or *finite field*.

Furthermore, there is a point $\mathcal{O}$ called the *point at infinity* or *zero point*. The basic operation of the elliptic curve is addition. To add points P_1 and P_2, which are both not at infinity, the following two steps must be executed:

- a) Compute the line $\mathcal{L}$ through P_1 and P_2 (or tangent, if $P_1 = P_2$) and find the third point of intersection with $\mathcal{E}$. Let this be *Q*.
- b) The sum $P_1 + P_2$ is defined as $P_3 :=$ - Q.

Definition of EC addition:

Let P be a point on an elliptic curve $\mathcal{E}$, with $\mathcal{O}$ as point at infinity. Then, sum is defined as follows:

$$P + \mathcal{O} = \mathcal{O} + P = P$$

Further, let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on $\mathcal{E}$, both not $\mathcal{O}$. Then the sum $P_1 + P_2$ is defined by,

- i) $\mathbf{P_3 = - Q}$, if $x_1 \neq x_2$. Here Q is the third point of intersection of $\mathcal{E}$ with of the line $\mathcal{L}$ through (x_1, y_1) and (x_2, y_2). (Figure 2)

- ii) $\mathbf{P_3 = - Q}$, if $P_1 = P_2$ and the tangent line through P is a single tangent. Here Q is the third point of intersection of $\mathcal{E}$ with of the tangent $\mathcal{L}$ through P. (Figure 3)
- iii) $\mathbf{P_3 = - P_1}$, if $P_1 = P_2$ and the tangent line through P is a double tangent.
- iv) $\mathbf{P_3} = \mathcal{O}$, if $P_1 = - P_2$.

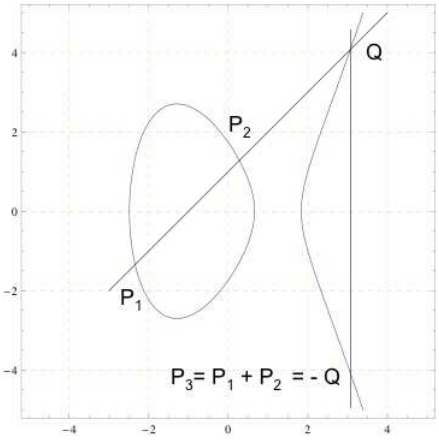

Figure 7: *"Addition for two different points P_1 and P_2."*

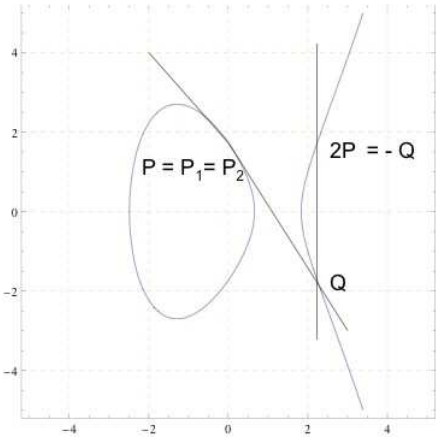

Figure 8: *"Addition for a tangent, where $P_1 = P_2 = P$. "*

B Theorems

In this section, some important theorems needed for the understanding of the main content of the paper are presented. The subsection B.1, refers to a theorem that it is needed for the two attacks described and the subsection B.2, presents the definition of the *Chinese Remainder Theorem*.

B.1 Theorems from the elliptic curve theory

Theorem A: Given a pair $P = (x, y) \in \mathcal{P}$ and a positive integer m. Assume that the tuple $(a_1, a_2, a_3, a_4, y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x)$ defines an elliptic curve $\mathcal{E}'$ over K. Then any fast multiplication type algorithm with input $(m, P, a_1, a_2, a_3, a_4)$ computes the result $m \bigotimes P$. Moreover, we have the equality $m \bigotimes P = m \cdot P_{\mathcal{E}'}$, where $P_{\mathcal{E}'} = P$ and $m \cdot P_{\mathcal{E}'}$, are points on $\mathcal{E}'$ and the latter is computed with "ordinary" point additions.

B.2 Chinese Remainder Theorem

Let m, $1 \leq i \leq k$, be k pairwise coprime integers. Further, let a_i, $1 \leq i \leq k$, be integers with $gcd(a_i, m_i) = 1$. Then, the system of k simultaneous congruence relations,

$$a_i x \equiv b_i \ mod(m_i), \qquad 1 \leq i \leq k,$$

has a unique solution modulo $\prod_{i=1}^{k} m_i$ for all possible k-tuples of integers $b_1, b_2, ... b_k$.

B.2.1 A simplified example

Suppose, that we want to solve using *Chinese Remainder Theorem* the following system,

$$d \equiv \ 6 \, mod(11), \quad d \equiv \ 11 \, mod(13), \quad d \equiv \ 16 \, mod(17)$$

i) Using the extended version of *Euclid's algorithm* [4] for **11** and **13 x 17 = 221**, we have:

$gcd(221, 11) = gcd(11, 1) = gcd(1, 0) = 1$

$221 = (1) \cdot 221 + (0) \cdot 11$
$11 = (0) \cdot 221 + (1) \cdot 11$
$1 = (1) \cdot 221 + (-20) \cdot 11$

ii) For **13** and **11x17 = 187**, we have:

$gcd(187, 13) = gcd(13, 5) = gcd(5, 3) = gcd(3, 2) = gcd(2, 1) = gcd(1, 0) = 1$

$187 = (1) \cdot 187 + (0) \cdot 13$
$13 = (0) \cdot 187 + (1) \cdot 13$
$5 = (1) \cdot 187 + (-14) \cdot 13$
$3 = (-2) \cdot 187 + (29) \cdot 13$
$2 = (3) \cdot 187 + (-43) \cdot 13$
$1 = (-5) \cdot 187 + (72) \cdot 13$

iii) For **17** and **11x13 = 143**, we have:

$gcd(143, 17) = gcd(17, 7) = gcd(7, 3) = gcd(3, 1) = gcd(1, 0) = 1$

$143 = (1) \cdot 143 + (0) \cdot 17$
$17 = (0) \cdot 143 + (1) \cdot 17$
$7 = (1) \cdot 143 + (-8) \cdot 17$
$3 = (-2) \cdot 143 + (17) \cdot 17$
$1 = (5) \cdot 143 + (-42) \cdot 17$

So, $d \equiv \ 6 \cdot (1) \cdot 221 + 11 \cdot (-5) \cdot 187 + 16 \cdot (5) \cdot 143 \equiv \ 50 \ mod(11 \cdot 13 \cdot 17)$.