
Design and Implementation of Plug-ins for JCrypTool

Visualization of the Cryptographic SPA and DPA Attacks

Diploma Thesis by Biqiang Jiang

March 2010



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Design and Implementation of Plug-ins for JCrypTool
Visualization of the Cryptographic SPA and DPA Attacks

Submitted Diploma Thesis by Biqiang Jiang

Advisor: Prof. Dr. Johannes Buchmann

Tutor: Dr. Vangelis Karatsiolis

Tag der Einreichung:

Ich möchte hier die Leute erwähnen, die mich bei meiner
Diplomarbeit sowie im ganzen Studium unterstützt haben.
Als erstes möchte ich mich bei meinen Eltern bedanken, ohne die
das Studium nicht möglich wäre.

Darüber hinaus bedanke ich mich bei meinem Betreuer Dr. Vangelis Karatiolis,
der mir während des Studiums viel geholfen hat. Er ist immer sympathisch
und hat mir viel geholfen aus meiner Arbeit einen Erfolg zu machen.
Ohne seine Anregungen und Unterstützung konnte diese Arbeit nicht abgeschlossen werden.
Auch Prof. Dr. Johannes Buchmann möchte ich danken, dass ich in seinem
Fachbereich meine Diplomarbeit anfertigen konnte.

Erklärung zur Diploma Thesis

Hiermit versichere ich die vorliegende Diploma Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, on March 26, 2010

(Biqiang Jiang)

Contents

| | |
|--|------------|
| Contents | iii |
| 1. Introduction to the Thesis | 1 |
| 1.1. The Eclipse and Eclipse RCP Platforms | 1 |
| 1.2. The JCrypTool Project | 2 |
| 1.3. Goal and Motivation | 3 |
| 1.4. Thesis Outlines | 3 |
| 2. The Introduction to PA | 6 |
| 2.1. Introduction to SCA and PA | 6 |
| 2.2. The Classification of the PA Attack | 6 |
| 2.3. The Environment of the PA Attack | 7 |
| 2.4. The possible Targets and Countermeasures of PA Attack | 8 |
| 3. The SPA Plug-in | 10 |
| 3.1. The Idea of the SPA Attack | 10 |
| 3.2. The RSA and "Squaring and Multiplication" Algorithms | 11 |
| 3.3. The Vulnerability of the "Squaring and Multiplication" Algorithm | 12 |
| 3.4. The Countermeasure against the SPA Attack | 12 |
| 3.5. The Design of the SPA Plug-in | 15 |
| 3.6. The Functionality of the SPA Plug-in | 17 |
| 3.7. The Implementation of the SPA Plug-in | 20 |
| 4. The Differential Power Analysis Plug-in | 24 |
| 4.1. The Idea of the DPA Attack | 24 |
| 4.2. The Elliptic Curve and the "Doubling and Addition" Algorithm | 25 |
| 4.3. The Vulnerability of the "Doubling and Addition always" Algorithm | 28 |
| 4.4. The Countermeasures against the DPA Attack | 29 |
| 4.5. The Design of the DPA Plug-in | 33 |
| 4.6. The Functionality of the DPA Plug-in | 35 |
| 4.7. The Implementation of the DPA Plug-in | 38 |
| 5. The Summary and Outlook | 41 |
| 5.1. Summary | 41 |
| 5.2. Outlook | 41 |
| List of Figures | 42 |
| List of Tables | 43 |
| A. Bibliography | 44 |

Abstract

The security of cryptographic system is based on difficult mathematical problems. However, since the publication of side channel attacks in 1995 more and more researchers have turned their attention to the hardware environment of the algorithms' implementation. An adversary invades a cryptographic system by skirting the mathematical entities and analyzing the side channel information of the underlying implementation environment. Instead of breaking the cryptographic system at the mathematical level directly, it proves to be easier and more efficient by cracking with side channel information. Power analysis attack is one of the most popular side channel attacks which have already been researched and discussed by many cryptographic scholars for many years. In this thesis the procedure of power analysis attack and its corresponding countermeasures on the JCrypTool platform are introduced and visualized. Two plug-ins, the SPA plug-in against RSA and the DPA plug-in against elliptic curve cryptographic system are implemented. We propose with the visualization plug-ins, the principle and process of power analysis attack and its countermeasures will be more intelligible, understandable, and suitable for the usage of teaching.

1 Introduction to the Thesis

In this thesis we implement and visualize a certain sort of side channel attacks¹ (SCA), namely the power analysis² (PA) attack. The visualization of the PA attack is realized in the form of plug-ins and can be divided into two parts: the simple power analysis (SPA) attack and differential power analysis (DPA) attack. With the plug-ins of visualization the users understand the mechanisms and corresponding countermeasures of PA attack more specifically and concretely. These two plug-ins are developed based on an Eclipse RCP project named JCrypTool³. Details for Eclipse RCP and JCrypTool project are available in Sections 1.1 and 1.2.

1.1 The Eclipse and Eclipse RCP Platforms

Eclipse⁴ is an open-source software development project, which focuses on the development of highly integrated tools and provides a full-featured, commercial-quality industry platform. It was originally developed by IBM⁵ as a Java programming IDE (Integrated Development Environments) project. Since 2001 it has been released as an open-source project. Due to the property of the resource sharing, the users are able to download the source code of Eclipse and develop their own features for Eclipse. More accurately, all new features can be infinitely expanded on the Eclipse platform in the form of plug-ins. In this sense Eclipse is no more an ordinary IDE. Because of its excellent extensibility, it can be regarded as a universal IDE for almost all programming languages: not only programming languages are supported on Eclipse, but also modeling language such as UML⁶ can be developed on it with suitable plug-ins. Currently the latest Eclipse Galileo is released in version 3.5.

Eclipse Rich Client (RCP)⁷ platform can be regarded as an aggregation in which the core functions of Eclipse are included. It can be considered the rest parts of the Eclipse platform after removing the contents of IDE, Ant, Search, Team, Debug, etc.. The remaining parts, including SWT⁸, JFace, OSGI, Runtime and UI, form the core functions of Eclipse platform which provide a basic workbench with the removable window components (editors and views), menus, toolbars, buttons, forms etc.. More exactly, Eclipse RCP realizes the shift from the plug-in development to a desktop application development. Furthermore, different from the original Eclipse platform, Eclipse RCP offers SWT graphics library and tool kits to replace the AWT and Swing tool kits for J2SE. With SWT the developer can call the graphics library on current operation system directly, which guarantees the Look&Feel design principle⁹ of the Java applications and provides the possibility of achieving a fully consistence with the operation system in use; more importantly, such a direct call to the local method improves the running speed of the Java-based applications greatly. Figure 1.1 describes the relationship between Eclipse and Eclipse RCP platforms visually.

¹ http://en.wikipedia.org/wiki/Side_channel_attack

² http://en.wikipedia.org/wiki/Power_analysis

³ <http://JCrypTool.sourceforge.net/JCrypTool/Home.html>

⁴ <http://www.eclipse.org>

⁵ <http://www.ibm.com>

⁶ <http://www.uml.org>

⁷ <http://www.eclipse.org/home/categories/rcp.php>

⁸ <http://www.eclipse.org/swt/>

⁹ http://en.wikipedia.org/wiki/Look_and_feel

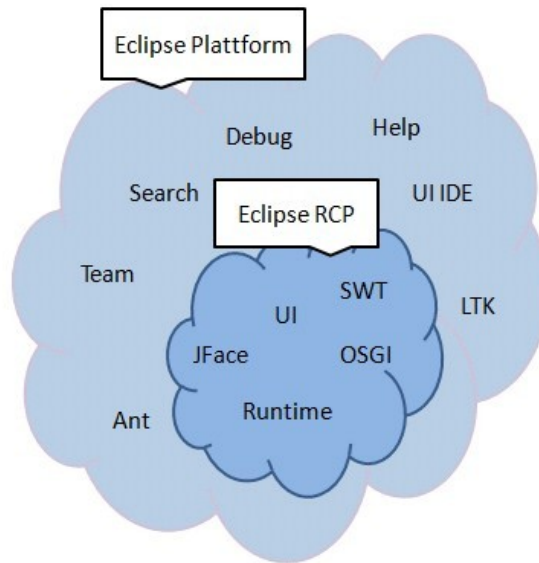


Figure 1.1.: The relation between Eclipse and RCP

1.2 The JCrypTool Project

JCrypTool is a useful open-source e-learning platform which has functions to apply, analyze, and visualize the cryptographic processes, mechanisms, and other related topics. JCrypTool project was launched as the successor to its famous predecessor: the cryptographic e-learning platform CrypTool. CrypTool is a free e-learning application for Windows and is developed in C++. In contrast to CrypTool, JCrypTool is a Java based Eclipse RCP project and has the feature of cross-platform regardless of computer architecture. That means all of the popular operating systems are suitable for running JCrypTool. The plug-ins developed for JCrypTool can also be used on those operating systems as follows: Windows, Linux, MacOS, Solaris, Unix and so on. As the successor of CrypTool, JCrypTool inherits all advantages of CrypTool and has already been used in some schools and universities in Europe as an innovation in the cryptographic teaching area. With JCrypTool the users not only understand the theories, principles and other related issues of cryptography more easily and efficiently, but also extend the JCrypTool plug-ins project in their own way to develop cryptographic plug-ins directly. JCrypTool is a free open-source project, which is available at SourceForge¹⁰ under the following address: <http://JCrypTool.sourceforge.net/JCrypTool/Home.html>.

The structure of JCrypTool platform is quite simple. It is made up of two parts:

- JCrypTool core project¹¹: the JCrypTool core project contains all basic components of the applications which form the running environment of JCrypTool platform.
- JCrypTool plug-in project¹²: the JCrypTool plug-in project is a collection of plug-ins for the JCrypTool platform.

The schematic structures of the individual components of JCrypTool core project and JCrypTool plug-in project are shown in Figures 1.2 and 1.3.

¹⁰ SourceForge is a web-based source code repository.

¹¹ <http://sourceforge.net/projects/JCrypTool/>

¹² <http://sourceforge.net/projects/jctplugins/>

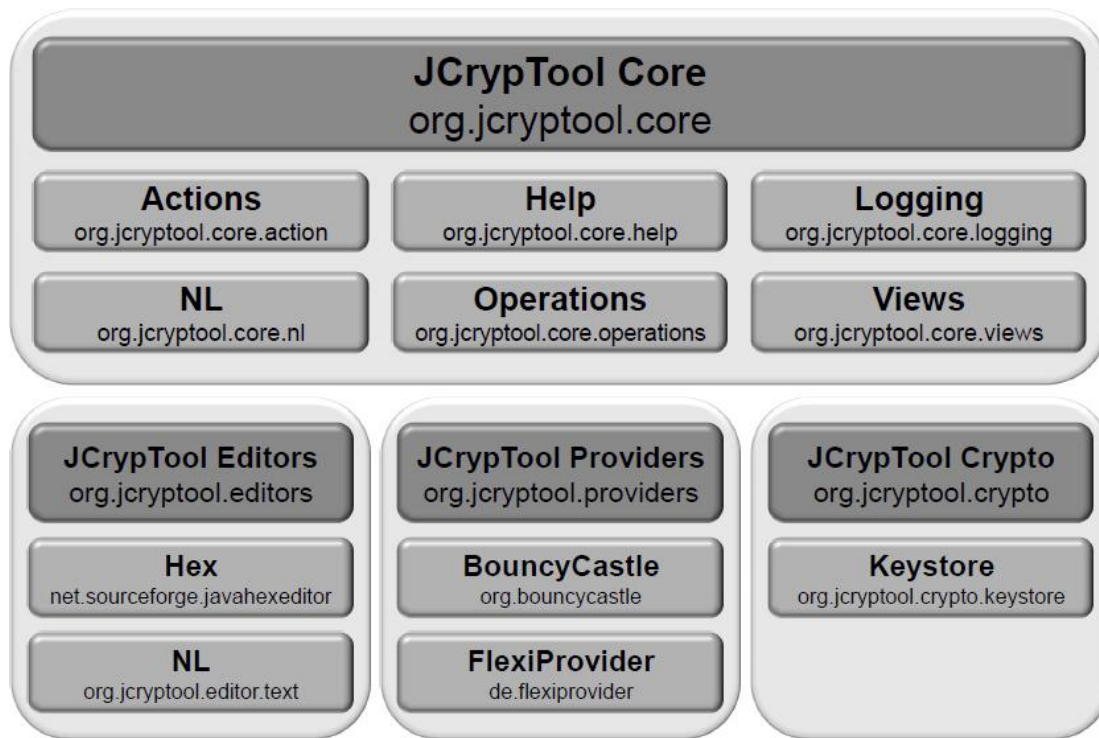


Figure 1.2.: The individual components of JCrypTool core project, taken from [GET09]

1.3 Goal and Motivation

The goal of this thesis is to implement and visualize the SPA and DPA attacks in the form of plug-ins of JCrypTool. The visualization for SPA and DPA attacks are separately based on RSA algorithm and elliptic curve cryptographic system. Simple instances based on these cryptographic systems are implemented and visual examples for the description of the procedure of SPA and DPA attacks are generated automatically.

For the students and others learners who are interested in cryptography, the cryptographic theories, principles, theorems, and formulas are sometimes too abstract to understand. Without visual and concrete examples, the procedure of learning is usually boring and tiresome. Furthermore, it is also sometimes puzzled for learners, how to apply the cryptographic knowledge into practical application. So our goal in this thesis is based on the plug-ins development with a high usability for both teaching and research purposes. Because JCrypTool project is increasingly being used in teaching, it is also relevant to realize the algorithm in a more visual, practicable, and understandable way.

The motivation of this thesis is to introduce and describe the plug-ins development on the cryptographic e-learning platform JCrypTool. It is important for the user to understand the algorithms and theories stepwise instead of just showing a final result. It is also essential to introduce and present the advantages and benefits of JCrypTool. On one side, with this e-learning platform the learners can easily access and use the algorithms' plug-ins designed and implemented by others with which the abstract cryptographic issues are more concrete, visual, and specific for understanding; on the other side, the learners can also apply the earned knowledge into practice through developing their own plug-ins on JCrypTool.

1.4 Thesis Outlines

Chapter 2 introduces some basic issues about side channel attacks. First in Section 2.1 we focus on the issues of side channel, side channel attacks, power analysis attack, etc.. Then in Section 2.2 the classification of PA is addressed and some definitions and theories about SPA and DPA will be explained. In Section 2.3 the hardware environment for a power analysis attack is described in details. The targets of PA attack and possible ways to prevent a cryptographic system from



Figure 1.3.: The individual components of JCryptTool plug-in project, taken from [GET09]

PA attack are discussed in Section 2.4.

Chapter 3 focuses on the development of the SPA plug-in. First of all, in Section 3.1 the idea and principle of an SPA attack is introduced. Then in Section 3.2 the RSA algorithm and the vulnerable method "squaring and multiplication", at which the SPA attack aims, is presented. In Section 3.3 the vulnerability of the "squaring and multiplication" method is explained in details. After that in Section 3.4 a method named "squaring and multiplication always" which is used as the countermeasure of SPA attack is described and examples are offered. In Section 3.5 the design of the SPA plug-in is provided and in Section 3.6 we aim at the functionality of the plug-in in which the structure of the plug-in is presented step by step. The implementation of the plug-in is drawn at the end of this chapter as Section 3.7 in which some useful code segments are picked out and explained.

Chapter 4 concentrates on the development of the DPA plug-in. The structure of this chapter is similar to the structure of chapter 3 except that the target of DPA attack is changed to the elliptic curve cryptographic system and its vulnerable method for point multiplication: the "doubling and addition always" algorithm. After explanation of the vulnerability of this insecure method, three methods are given as the countermeasures of DPA attack.

Chapter 5 summarizes the purpose of this thesis. An outlook for further development is described as the end of this thesis.

2 The Introduction to PA

The security of a cryptographic system is no more equal to a perfect designed cryptographic algorithm. The environment, in which the decryption or signature operations are implemented, is also taken into consideration. Security is no longer just an overall concept. It does not mean that the whole system is secure enough, even if each of the components of a cryptographic system is secure. For the system in practice, the robustness of a cryptographic system should be tested under various invading models from various angles, not only just from the mathematical algorithmic side, but also from the side channels of a cryptographic system which are regarded as a more and more vulnerable and relevant aspect of security.

This chapter introduces the basic knowledge of side channel and power analysis attack. First of all, some definitions are explained in Section 2.1.

2.1 Introduction to SCA and PA

Definition 1 *Side channel: side channels, which are capable of leaking information about the secret key, can be defined to be unintended output channels from a system. For example, the power consumption, the clock rate, and the electromagnetic (EM) emissions are considered side channels of a cryptographic system.*

Definition 2 *Side channel attack: in cryptography side channel attacks are attacks which are different from the brute force attacks or the other traditional mathematical attacks. Side channel attacks are based on side channel information such as timing information, electromagnetic emissions, power consumption statistics and so on. Attacks through side channel information are easier, cheaper, and more efficient than those from the traditional attacks. Side channel attacks against the algorithmic secure system that is stored in electronic devices as smart cards or RFID chips are becoming more and more popular and turning to be a major and serious problem in both economic and industrial fields.*

Figure 2.1 shows a real world model of a cryptographic system including the side channels over a traditional mathematical cryptographic system. The two rectangles in the figure can be regarded as the encryption and decryption devices, in which the cryptographic algorithm is processed. In Figure 2.1 Alice and Bob are communicating with each other and Eva is the adversary. The traditional cryptographic attack focuses on the attacks with a mathematical solution. If the cryptographic algorithm is robust enough, it is not easy to solve the encryption algorithm with the traditional mathematical or brute force methods. However, in the real world the adversary Eva invades the cryptographic system not through breaking the algorithm but by using information gathered from the side channels. That means Eva bypasses the complex encryption and decryption algorithms and looks for the other simpler ways to achieve the purpose of cracking. All the information marked with dotted arrow belong to side channel information, with which the adversary Eva may invade indirectly the system more easily.

Side channel attacks can be classified into several kinds: PA attack, timing attack (TA), fault attack (FA), EM attack, etc.. Among them power analysis is one of the most successful attacks, so it has been chosen as the topic of this thesis.

2.2 The Classification of the PA Attack

Power analysis was firstly published by Paul Kocher in 1998 as a novel, powerful, and efficient attack. Since its publication power analysis has been discussed and researched over ten years. Hundreds of research papers in this area have been published. Literature on PA and its corresponding countermeasures is coming out continuously. Actually, PA attack has become the focus of the current research about side channel attacks.

Definition 3 *Power analysis attack: PA attack can be regarded as a form of side channel attack in which the attacker measures the power consumption of special encryption or decryption operations of a cryptographic hardware system (such as smart cards, tamper-resistant integrated circuit, etc.) and through analysis of the tiny difference between operations the adversary seizes security related important information, even recovers the secret keys.*

The PA is divided into two sorts: SPA and DPA.

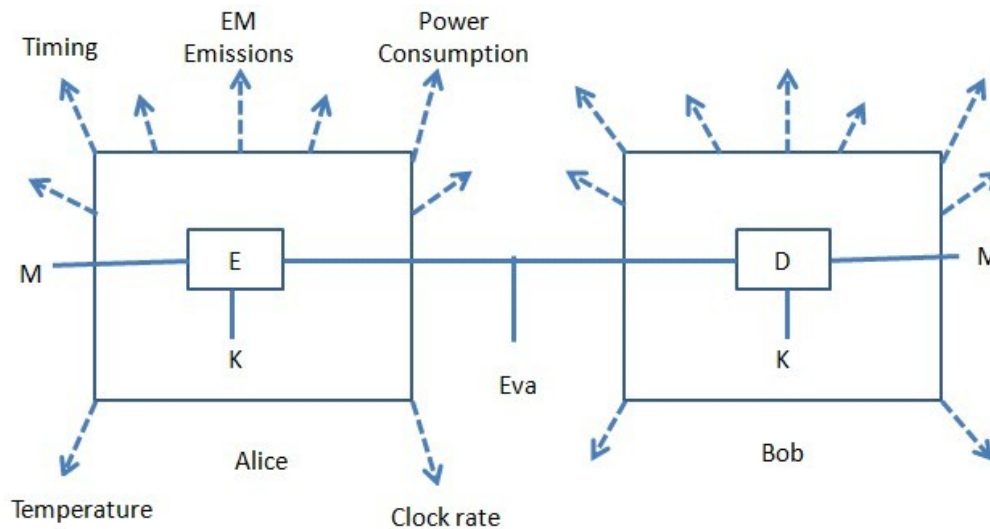


Figure 2.1.: The side channels of a traditional cryptographic system

SPA interprets the private key of a cryptographic algorithm directly through visualization and comparison of the power consumption traces collected during the cryptographic procedure. An adversary recovers the secret key by observing directly the power consumption of a system. The amount of the power consumption depends on which kind of microprocessor instruction is performed. The secret keys of the asymmetric encryption and signature algorithms such as RSA can be identified from a certain operation, which is performed during the decryption or signature procedure. The common countermeasure against an SPA attack in software level is normally realized with appending dummy operations, with which the difference of the power consumption traces between different microprocessor instructions will be reduced. Such a method of protecting a cryptographic system from SPA attack is described in Section 3.4 precisely. However, even though such a method is resistant to SPA attack, it still remains vulnerable against DPA attack.

DPA is the improved SPA attack with statistical analysis and error correction techniques to recognize the tiny differences in power consumption signals of distinct operations. A DPA attacker uses not only visual but also statistical analysis and error-correction statistical methods to filter noise and gains additional information about the process of algorithm performing. In this sense, DPA attack is too powerful and efficient to be prevented. In order to protect a cryptographic system from invading by DPA attack, some algorithmic parameters or computational processes are randomized. In this thesis three kinds of countermeasures against DPA attack are discussed, namely randomizing the scalar multiplier, randomizing the initial point P and randomizing the projective coordinates. Details for DPA attack are described in Chapter 4.

2.3 The Environment of the PA Attack

The equipments required for PA attack can be divided into three basic components, which are listed in Table 2.1.

The functionality of each component is described as follows:

1. The cryptographic devices are the target to be cracked. They are connected to both control devices and measure-record devices.
2. The control device is used to instruct the cryptographic devices to perform a certain procedure or execution. Furthermore, it is also used to analyze the measured power consumption data with special software like power signal analyzer.
3. The measure-record devices is aiming at detecting and recording the power consumption traces and other useful information.

Table 2.1.: The devices for the PA attack

| No. | device | instances |
|-----|------------------------|--|
| 1. | cryptographic devices | smart cards, RFIDs, card reader, etc. |
| 2. | control device | a PC with software like smart card controller, signal analyzer, etc. |
| 3. | measure-record devices | oscilloscope, probes, etc. |

Figure 2.2 presents a typical environment for PA attack.



Figure 2.2.: The environment for PA attack¹

The procedure of PA attack can be divided into four steps, as shown in Figure 2.3.

1. First the control device initializes the measure devices with an oscilloscope control software which is installed in the PC.
2. Then the control device sends a control instruction to cryptographic devices, e.g. the smart card, to perform a decryption or signature operation.
3. The measure device oscilloscope records all power traces during the encryption procedure of the smart card reader, including the trigger signal, measuring signal, etc.
4. After that the oscilloscope sends all recorded data back to the pc, analyzes the measured data with the signal analysis software and tries to crack the secret key step by step.

2.4 The possible Targets and Countermeasures of PA Attack

Because power consumption can be measured and analyzed, PA attack is applicable to the hardware's implementation of a cryptographic system. It is effective and successful in invading smart cards and other embedded systems storing the secret key. The countermeasures against PA attack can be designed and realized in both hardware and software level.

- First, the weakness of hardware design for most embedded systems is obvious. For instance, the power supply of embedded systems like smart cards is always supplied by an external source. An attacker can invade a smart card with relative cheap hardware like a numeric oscilloscope and a PC with special softwares for power analysis. Since smart cards have not any protection to conceal the power consumption in decryption process, the adversary just plugs wires or probes at the right place of a smart card to be able to measure and record the power traces.

¹ This figure is picked from veri-smart GmbH, available at: <http://www.veri-smart.cn/upload/products/20098104376.jpg>

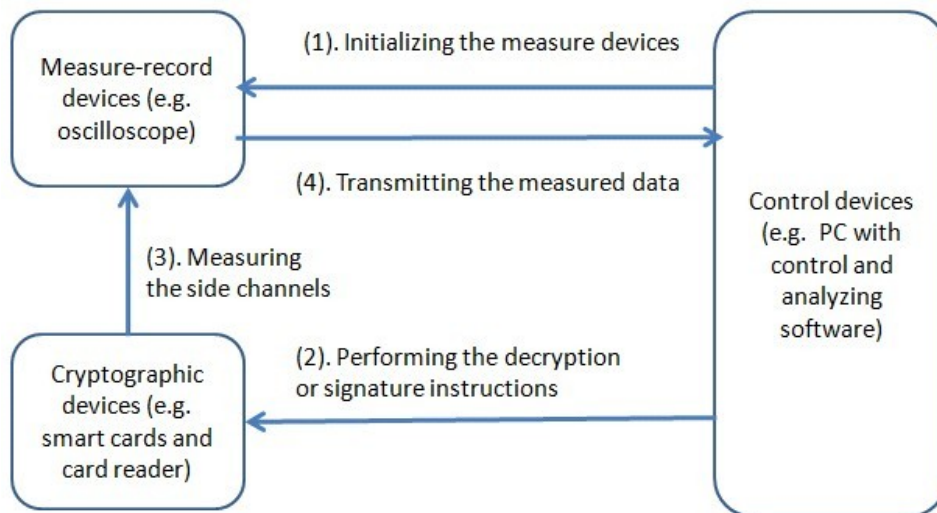


Figure 2.3.: The procedure of PA attack

Although the secret key stored in embedded system is not accessible directly, it is still correlated with the power consumption, which can be easily measured with PA.

As a solution, the countermeasures in hardware level are realized in the form of special dedicated devices like FPGAs. Such special hardware masks or changes the power traces and in this case the PA attack is no more a significant threat. However, to realize such protection in hardware level, the whole original production lines need to be updated, which causes continuously investments for both clients and the constructor of the device. It is normally too expensive to accept. Besides that, the side channel attack develops rapidly, a special designed hardware cannot ensure to be perfect to all kinds of side channel attacks. It might be designed to protect the embedded system from DPA attack, but it might be still vulnerable to other kinds of SCA like EM or fault attacks. So the protection in hardware level is sometimes not the best choice.

- The second way to protect a cryptographic system from PA attack depends on the software level, namely the algorithmic protection. A solution of improved algorithmic level is relatively cheaper, more flexible, and acceptable than updating the hardware directly. The costs for a software updating are usually more affordable and the period for achieving software updating is also shorter than that of hardware updating.

In this thesis, the process of PA attack and its countermeasures in arithmetic level are focused and discussed. The visualization of SPA and DPA attack and the common countermeasures in software level are implemented as plug-ins of JCrypTool.

3 The SPA Plug-in

SPA is a side channel attack which reveals the secret key by observing graphic difference of power consumption traces between distinct operations and instructions. Because different instructions have normally different power consumption profiles, the secret key of a certain cryptographic algorithm is recovered more easily through comparing the power consumption seized by oscilloscope rather than using traditional mathematical methods. A plug-in for JCrypTool concerning SPA attack against the method "squaring and multiplication" is designed and implemented in this chapter.

Firstly the idea of the SPA attack is described in Section 3.1. Then in Section 3.2 the principle of "squaring and multiplication" method is introduced. In Section 3.3 the vulnerability of this method is explained. After that a countermeasure named "squaring and multiplication always" is given in Section 3.4. In Section 3.5 we focus on the design of the plug-in. The functionality of the SPA plug-in is given step by step in Section 3.6. In Section 3.7 some program segments are chosen to explain how the plug-in is implemented.

3.1 The Idea of the SPA Attack

The SPA attack is based on the observation of a single power consumption trace. Some special operations of cryptographic algorithms are related to the secret key and may reveal important information. Typically, for some cryptographic system the condition branches statements depend on the bits of secret key.

For example:

```
1  if (the current bit of the secret key is "1")
2  then
3  do operation A
4  else
5  do operation B
```

If the difference of power traces between operations A and B is obvious enough, the current bit of the secret key, which is used as the condition of the if-statement, can be retrieved through analyzing the difference of power traces between operations A and B. In the same way all bits of the secret key can be recovered step by step. Figure 3.1 illustrates a typical power consumption trace. In this figure the operations A and B are corresponding to the current bits "1" and "0" of the secrete key. It is clear that the power traces reveal the sequence of the secret key as "11010".

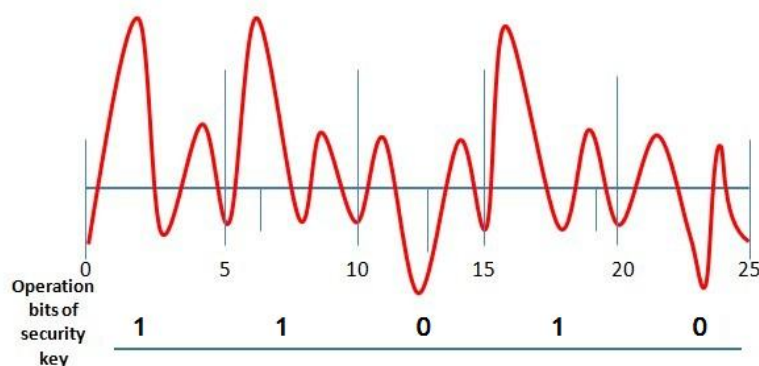


Figure 3.1.: The idea of SPA attack

3.2 The RSA and "Squaring and Multiplication" Algorithms

In this section some basic issues about the target for the SPA attack are discussed firstly.

Definition 1 *RSA algorithm¹: RSA (which stands for Rivest, Shamir and Adleman who first publicly described it) is an algorithm for public-key cryptography. It is the earliest known public key cryptographic system which is suitable for encryption and signature. The security of RSA is related to the mathematical problem of factoring large numbers. During encryption and decryption process the "squaring and multiplication" algorithm is used to accelerate the execution of modular exponentiation.*

The procedure of the RSA algorithm can be divided into three steps:

- Key generation: since the RSA algorithm is a public key cryptographic system, a public key and a private key are necessary for encryption and decryption operations. The key generation is described as follows:
 1. Choose two distinct large enough prime numbers p and q .
 2. Compute $n = pq$, n is used as the modulus in both encryption and decryption operations.
 3. Compute the Euler's totient function $\varphi(pq) = (p - 1)(q - 1)$.
 4. Choose an integer e which satisfies $1 < e < \varphi(pq)$, and e is co-prime with $\varphi(pq)$, e is chosen as the public key which is shared for encryption operation.
 5. Determine d with $de \equiv 1(\text{mod}(\varphi(pq)))$, d is the inverse of e and is used as the private key which must be kept secret for decryption operation.
- Encryption: the encryption function is: $m^e \equiv c(\text{mod}n)$, m is the message which should be encrypted, e is the public key, n is the product of p and q , c is the encrypted cipher text.
- Decryption: the decryption function is: $c^d \equiv m(\text{mod}n)$, c is the message which should be decrypted, d is the private key, n is the product of p and q , m is the decrypted plain text.

For more details about RSA algorithm see [RSA09]. For security in the practical applications the public and private keys are often chosen long enough, typically 1024-2048 bits long. In order to accelerate the exponentiation operation in encryption and decryption, the algorithm "squaring and multiplication" is used.

The "squaring and multiplication" algorithm²: "squaring and multiplication" is a general way for fast exponentiation of large powers of a number. The basic procedure for exponentiation in the decryption operation of the RSA algorithm is presented as the following pseudo code:

```
1  algorithm "squaring and multiplication" method:
2
3  input:
4      cipher text = c;
5      private key = d;
6      d in binary form = b with m+1 digits;
7      (digit m: first significant bit; digit 0: last bit)
8      modulus n = p*q;
9
10 output:
11     Res = c^d mod n;
12
13 process:
14     int Res = 1; //initial value
15     for(int i = m; i>-1; i--) {
```

¹ <http://en.wikipedia.org/wiki/RSA>

² http://en.wikipedia.org/wiki/Exponentiation_by_squaring

```

16 Res = Res^2 mod n;           //"squaring "
17 if (bi == 1) {
18     Res = (Res * c) mod n;   //"multiplication "
19 }
20 else {
21 }

```

The code segments in the loop implicates the vulnerability of the "squaring and multiplication" algorithm which is explained in Section 3.3 in details.

3.3 The Vulnerability of the "Squaring and Multiplication" Algorithm

Just as the pseudo code described above, the squaring operation is processed always. However, only if the current bit of the secret key is "1", the additional multiplication operation in the if-statement is carried out. So it is concluded that the secret key of RSA is related with the operation of "squaring and multiplication". In this case the power consumption traces of "squaring and multiplication" operations is clearly different from the power consumption traces of "squaring", which means the secret key is also correlated with the power consumption traces of different operations. That can be regarded as the vulnerable point of the "squaring and multiplication" method. As described in Figure 3.2, if the power consumption traces are obviously distinct, the private key is recovered step by step through comparison of the power traces between two branches of condition statement and this kind of method is called an SPA attack.

In general, SPA distinguishes the sequence of microprocessor instructions executed and it is always used to break the cryptographic implementations with condition statements. In these algorithms the execution path depends strongly on the data being processed, for different data different operations are chosen by condition-statement to execute. For instance, the "doubling and addition" method in EC cryptographic system, the sequential key generation in the DES algorithm, both of them are vulnerable to an SPA attack.

```

Part 2. Square and Multiply Algorithm:
modul = p*q = n;

Output:
Res = c^d mod n;

Process:
int Res = 1; //initial vaule: Res
for(int i = m; i>-1; i--) {
    Res = Res^2 mod n;    //"Square"
    if (bi == 1) {
        Res = (Res * c) mod n;    //"Multiply"
    }
    else {
        //do nothing; } }

```

Figure 3.2.: The vulnerability of the "squaring and multiplication" algorithm

3.4 The Countermeasure against the SPA Attack

A countermeasure against the SPA attack based on dummy operations was first published by Coron in 1999 [COR99]. Such a countermeasure is called Coron's method. For the application in RSA, it is also named as "squaring and multiplication always". This method against SPA attack is simple, efficient but a little costly.

Definition 2 *Coron's method: the idea of this method is to add an additional dummy operation independently from the fact that the current bit for the conditional statement is "0" or "1". The resulting operation is replaced by a constant pattern and the conditional statement is avoided. In this way the difference between the patterns related to the bit "0" and "1" only include the pointer and memory operations, and they are in general cases not detectable [COR99]. However, this method is a little time-wasting. Instead of the original algorithm with $O(n/2)$ time complexity, the dummy method costs $O(n)$. Because of its additional operations, an additional register is always necessary to save the additional temporary result.*

Similar to the Coron's method, the weakness of "squaring and multiplication" can be improved with the modification of the condition statement. In the modified algorithm "square and multiply always", no matter what is the value of the current bit, the multiplication operation is always executed after the doubling operation. And the temporary result after squaring and multiplication operations is separately saved in registers. The selection of the result is determined by the current bit of the secret key. In this way, the if-statement in the process is avoided and it is hard enough for the adversary to find out the tiny difference or to recover the private key just through the comparison of the difference between power traces with one single sample. The following pseudo code describes the idea of "squaring and multiplication always".

```

1
2 algorithm "squaring and multiplication always" method:
3
4 input:
5     cipher text = c;
6     private key = d;
7     d in binary form = b with m+1 digits;
8     (digit m: first significant bit; digit 0: last bit)
9     modulus n = p*q;
10
11 output:
12     Res = c^d mod n;
13
14 process:
15     int Res = 1; //initial value
16     for(int i = m; i > -1; i--) {
17     {Res_0 = Res^2 mod n; // "squaring"}
18     {Res_1 = Res_0 * c mod n; // "multiplication"}
19     Res = Res_(bi)}

```

For better understanding of the countermeasure "squaring and multiplication always", two visual examples for the original "squaring and multiplication" and the corresponding countermeasure are presented as follows:

- In order to explain the functionality of the countermeasure, an example for the original vulnerable algorithm "squaring and multiplication" will be given below.

Figure 3.3 is an example for the determination of the parameters of RSA:

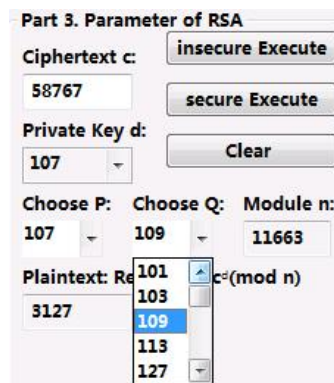


Figure 3.3.: The third part of the SPA plug-in

1. The cipher text is given as the number "58767".
2. The "private key" is selected as the number "107".
3. The two distinct primes of "Module n" are "109" and "107". After both of "P" and "Q" are defined, the execution of "Module n = P*Q" is carried out and the "Module n" is output in the corresponding text field.
4. The module n is automatically executed as the number "11663", together with the execution of the module, both of the buttons "insecure execute" and "secure execute" are enabled automatically.

After clicking either of the buttons, the process of the "squaring and multiplication" or "squaring and multiplication always" is animated, the temporary results is listed in the table as in Figure 3.4 and the final result after the execution is displayed in the text field "Plaintext: Result $R = C^d(mod n)$ " below.

| Part 5. Square and Multiply (from left highest to right lowest value bit) | | |
|---|--|-------------------------------------|
| Round Counter (left to right) | Result after Squaring | Result after Multiplication |
| Input: | Res = Res² mod n | Res = Res * c mod n |
| Ciphertext(c): 58767; Res: 1: | | |
| private key(d): 107 (dec.) | | |
| Binary: 1101011 (bin.) | | |
| Modul n = p*q = 11663 | | |
| Process: | | |
| 1. highest bit = 1 | Res = 1 ² mod 11663 = 1 | Res = 1 * 58767 mod 11663 = 452 |
| 2. highest bit = 1 | Res = 452 ² mod 11663 = 6033 | Res = 6033 * 58767 mod 11663 = 9437 |
| 3. highest bit = 0 | Res = 9437 ² mod 11663 = 9964 | |
| 4. highest bit = 1 | Res = 9964 ² mod 11663 = 5840 | Res = 5840 * 58767 mod 11663 = 3842 |
| 5. highest bit = 0 | Res = 3842 ² mod 11663 = 7269 | |
| 6. highest bit = 1 | Res = 7269 ² mod 11663 = 4971 | Res = 4971 * 58767 mod 11663 = 7596 |
| 7. highest bit = 1 | Res = 7596 ² mod 11663 = 2355 | Res = 2355 * 58767 mod 11663 = 3127 |

Figure 3.4.: The listed data for the "squaring and multiplication"

- In the first five lines of the table all inputs are listed, including the cipher text (cipher text: 58767), the initial result (Res: 1), the private key in decimal (private key(d): 107(dec.)) and binary (private key(b): 1101011(bin.)) forms, and the module (n: 11663).
 - Then all temporary results are displayed in the process: including the round of the loop, the result after squaring, and the result after multiplication.
- As described in Figure 3.4, the private key k in binary form is "1101011" with seven digits, the loop for "squaring and multiplication" will be carried out seven times and the multiplication operation will be executed five times because there are five digits "1" included in the secret key.
- In the last two lines the output will be presented (Final Result: 3127).

And below the table, the visualization of the power consumption traces is illustrated automatically as shown in Figure 3.5.

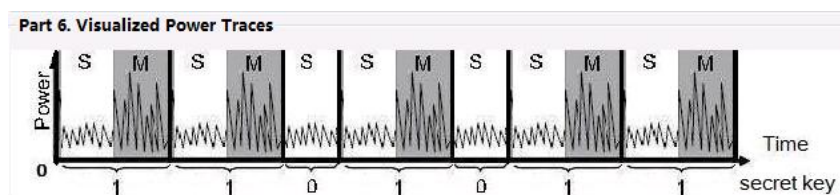


Figure 3.5.: An example of the visualization of the "squaring and multiplication"

The difference of power traces between the distinct operations "squaring" and "squaring and multiplication" is obvious. From the power traces it is not difficult to read out that the sequence of the secret key is "1101011". So this instance illustrates the idea and process of the SPA attack.

- Now an example for the countermeasure "squaring and multiplication always" is given as follows:
 - all parameters for RSA algorithm remain the same and after enabling both of the buttons for execution, the button "secure Execute" will be clicked.
 - As seen in Figure 3.6, no matter whether the current bit of the secret key is "0" or "1", the multiplication operation will always be carried out. The current bit functions now as the selection determiner of the initial value for the next round loop. After the execution, the final result of the method "squaring and multiplication always" stays the same result of the original method as 3127.

| Part 5. Square and Multiply (from left highest to right lowest value bit) | | |
|---|-------------------------------------|---|
| Round Counter (left to ri... | Result after Squaring | Result after Multiplication |
| Input: | $Res_0 = Res^2 \bmod n$ | $Res_1 = Res_0 * c \bmod n$ |
| Ciphertext(c): 58767; R... | | |
| private key(d): 107 (dec.) | | |
| Binary: 1101011 (bin.) | | |
| Modul n = p*q = 11663 | | |
| Process: | | |
| 1. highest bit = 1 | $Res_0 = 1^2 \bmod 11663 = 1$ | $Res_1 = 1 * 58767 \bmod 11663 = 452$ |
| 2. highest bit = 1 | $Res_0 = 452^2 \bmod 11663 = 6033$ | $Res_1 = 6033 * 58767 \bmod 11663 = 9437$ |
| 3. highest bit = 0 | $Res_0 = 9437^2 \bmod 11663 = 9964$ | $Res_1 = 9964 * 58767 \bmod 11663 = 1810$ |
| 4. highest bit = 1 | $Res_0 = 9964^2 \bmod 11663 = 5840$ | $Res_1 = 5840 * 58767 \bmod 11663 = 3842$ |
| 5. highest bit = 0 | $Res_0 = 3842^2 \bmod 11663 = 7269$ | $Res_1 = 7269 * 58767 \bmod 11663 = 8285$ |
| 6. highest bit = 1 | $Res_0 = 7269^2 \bmod 11663 = 4971$ | $Res_1 = 4971 * 58767 \bmod 11663 = 7596$ |
| 7. highest bit = 1 | $Res_0 = 7596^2 \bmod 11663 = 2355$ | $Res_1 = 2355 * 58767 \bmod 11663 = 3127$ |

Figure 3.6.: The listed data for the "squaring and multiplication always"

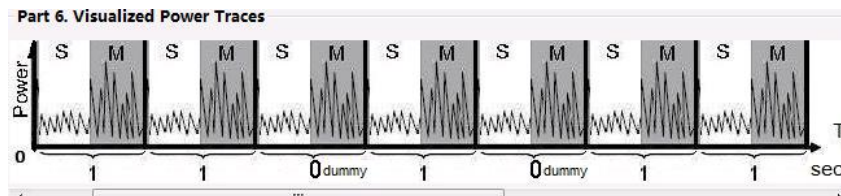


Figure 3.7.: An example of visualization of the "squaring and multiplication always"

- The visualization of the power traces for "squaring and multiplication always" is given in Figure 3.7.

As described in the figure, because of the dummy multiplication operation added, the power traces between the bit "1" and "0" are identical. So it is almost impossible for an adversary with the SPA attack to recover the secret key through the comparison of the difference between the power consumption traces. However, such countermeasure remains insecure against DPA attack, which is introduced in Chapter 4.

3.5 The Design of the SPA Plug-in

The SPA plug-in is designed as a view, in which the Eclipse framework is used. As a part of the Eclipse framework, the view inherits ViewPart from the abstract class. The design and implementation for both SPA- and DPA plug-ins are under the environment of SWT Designer^{TM3}, which is a powerful GUI designer for Eclipse SWT and RCP development. The latest version is 7.3.0 and a trial version for 14 days activation is available at: <http://www.instantiations.com/gwt designer/download-trial.html>.

Figure 3.8 describes the class structure of the SPA plug-in.

The SPA Plug-in is divided into four packages:

- org.jcryptool.algorithm
- org.jcryptool.visual.sidechannelattack
- org.jcryptool.visual.sidechannelattack.spa.views
- com.swtdesigner

The functionality for all classes in these four packages is described as follows:

- The "SWTResourceManager.java" is an automatic generated class which handles SWT widgets creation and disposition.

³ available at: <http://marketplace.eclipse.org/content/swt-designer>

Table 3.1.: The behavior description of SPA plug-in

| No. | type | initial condition | text | description |
|------|-------------------|-------------------|------------------|--|
| 1. | text field | enabled | see Figure 3.1 | description of RSA algorithm |
| 2. | text field | enabled | see Figure 3.1 | description of "squaring and multiplication" |
| 3. | input field | enabled | empty | input of the ciphertext c |
| 4. | drop-down button | enabled | empty | selection of private key |
| 5. | drop-down button | enabled | empty | selection of parameter p |
| 6. | drop-down button | enabled | empty | selection of parameter q |
| 7. | output text field | disabled | empty | output of the module n |
| 8. | output text field | disabled | empty | output of the result R |
| 9. | button | disabled | insecure Execute | activation of "squaring and multiplication" |
| 10. | button | disabled | secure Execute | activation of "squaring and multiplication" always |
| 11. | button | enabled | Clear | elimination of inputs and outputs |
| 12. | button | enabled | see Figure 3.1 | description of "countermeasures" |
| 13. | table column | disabled | empty | round counter |
| 14. | table column | disabled | empty | result after Square |
| 15. | table column | disabled | empty | result after Multiplication |
| 16. | text field | disabled | empty | display of visual power traces |
| I. | group | enabled | see Figure 3.1 | red marked main group |
| II. | group | enabled | see Figure 3.1 | green marked RSA algorithm group |
| III. | group | enabled | see Figure 3.1 | yellow marked "squaring and multiplication" group |
| IV. | group | enabled | see Figure 3.1 | blue marked "parameter of RSA" group |
| V. | group | enabled | see Figure 3.1 | purple marked "countermeasures" group |
| VI. | group | disabled | see Figure 3.1 | black marked "table of intermediates" group |
| VII. | group | disabled | see Figure 3.1 | pink marked "visual power traces" group |

- The third column "initial condition" states whether the initial condition of the object is enabled, disabled, or initially invisible. For example all output fields and some buttons are initially disabled and will be enabled after a certain step.
- The column "text" specifies the stated text on each object and the last column "description" represents briefly the function of SWT object.

A UML activity diagram⁴ is offered below in Figure 3.10 which is used to display the transformation between the different states of workflow. An activity diagram shows also the flow of control and the relations between different components.

3.6 The Functionality of the SPA Plug-in

The view of SPA plug-in is visualized and maximized in Figure 3.9. The view can be divided into 6 parts:

- In the upper right the first part is located, see Figure 3.11 of the view: "RSA Algorithm", which is defined in the "rsaAlgorithmGroup". The group of "RSA Algorithm" describes briefly seven steps of RSA algorithm. The two with yellow background color marked lines of steps are the encryption and decryption functions of RSA, which are usually realized with "squaring and multiplication" algorithm to execute the exponentiation operation. However, "squaring and multiplication" algorithm is insecure against SPA attack.
- Under the first part is the second part of the view (Figure 3.12), namely the "squaring and multiplication" algorithm, which is defined in the "squareAndMultiAlgGroup". In this part we present the process of "squaring and multiplication" in form of pseudo code and explain the vulnerability of this algorithm (the if-statement with red color marked), with the analysis and comparison of the condition statement every bit of the private key can be recovered step by step.

⁴ This UML diagram is drawn with Microsoft Visio 2007 and the introduction of UML diagrams is available at: http://en.wikipedia.org/wiki/Unified_Modeling_Language.

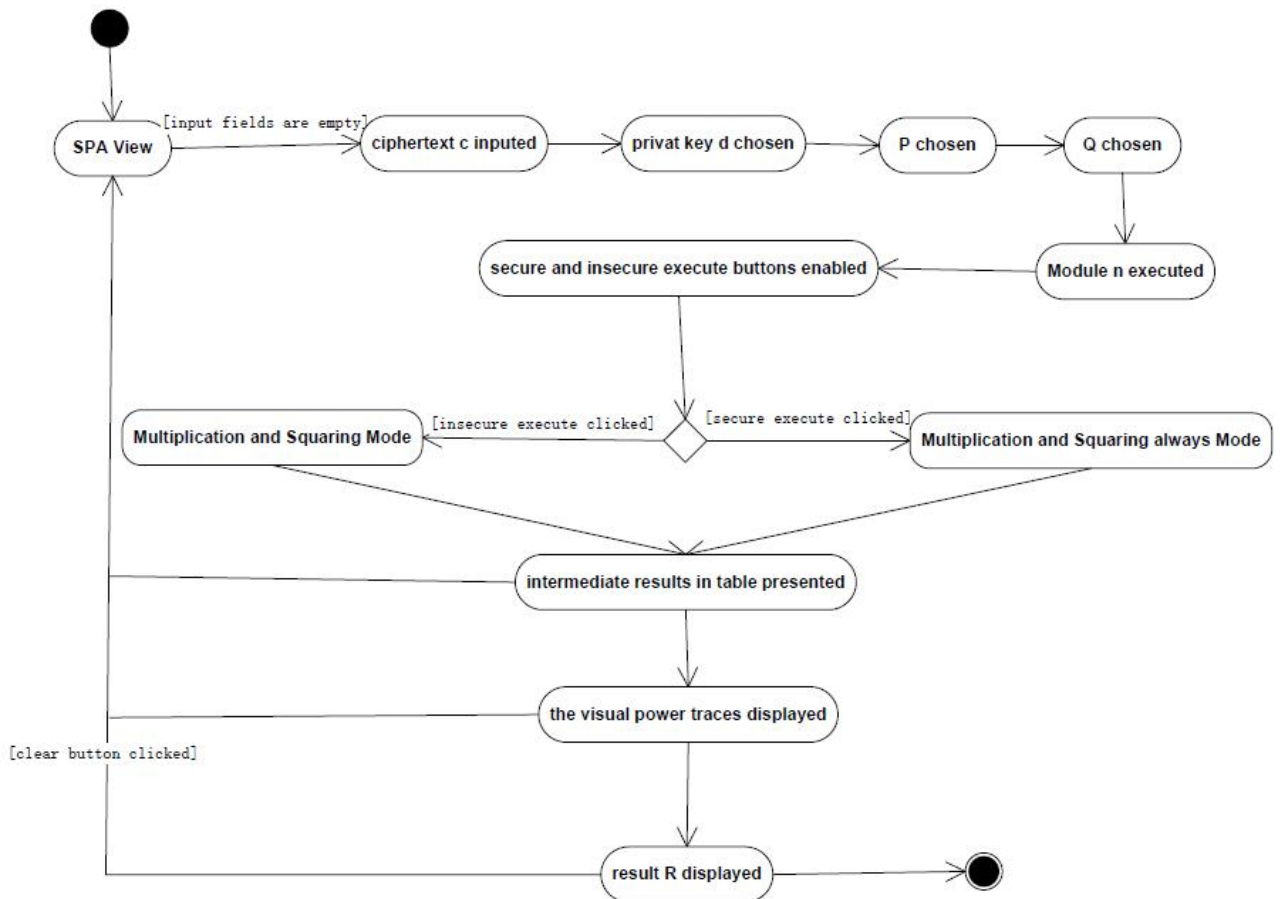


Figure 3.10.: The UML activity diagram of the SPA plug-in

Part 1. RSA Algorithm:
 and decryption functions(yellow marked) is normally realised with Square and Multiply algorithm, which is proved vulnerable to SPA.

1. Choose two distinct prime numbers p and q .
2. Compute the modulus $n = p \cdot q$.
3. Compute the totient: $\phi(n) = (p-1) \cdot (q-1)$.
4. Choose a public key 'e' coprime with $\phi(n)$.
5. Determine private key 'd': $d \cdot e = 1 \pmod{\phi(n)}$.
6. Encryption Function: $c \equiv m^e \pmod{n}$.
7. Decryption Function: $m \equiv c^d \pmod{n}$.

Figure 3.11.: The first part of the SPA plug-in

```

Part 2. Square and Multiply Algorithm:
Output:
Res = cd mod n;
Process:
int Res = 1; //initial vaule: Res
for(int i = m; i > -1; i--) {
Res = Res2 mod n; // "Square"
if (bi == 1) {
Res = (Res * c) mod n; // "Multiply"
}
else {
//do nothing; }
}

```

Figure 3.12.: The second part of the SPA plug-in

- In the upper middle the third part (Figure 3.3) of the view is located, which is defined in "parameterOfRSAGroup", named "Parameter of RSA". In this part the user can input the "ciphertext c", and choose the private key d, as well as two distinct prime parameters "P and Q" of "the Module n". After all above mentioned parameter of "RSA" are specified, the two buttons "insecure Execute" and "secure Execute" will be enabled. By clicking the "insecure Execute" button triggers the process of the insecure "squaring and multiplication" algorithm, and by clicking "secure Execute" button the secure "squaring and multiplication always" algorithm is carried out. Each step of the process is displayed in the upper right table and the visual diagram of power traces is painted in the lower right of the view. An example of this part has already been given in Section 3.4.
- The fourth part of the view "Countermeasures" is located in lower middle. The "squaring and multiplication always" algorithm is introduced here in the form of pseudo code. Five with green background color marked lines of codes are the improvement of the original code. As shown in Figure 3.13, whichever the current bit is "0" or "1", the "squaring" and the "multiplication" operations are always executed. In this way the adversary cannot easily recover the private key through analyzing and comparing the difference of power traces between the bit "0" and "1".

```

Part 4. Countermeasures
Note: in the modified algorithm
"Square and Multiply Always", no
matter the current bit is "0" or "1",
the multiplication operation will be
always executed. So it is impossible
to recover the private key through
comparison of the difference
between power traces. So this
"Square and Multiply Always"
Algorithm is resistant to SPA.
Process:
int Res = 1; //Initial vaule: Res
for(int i = m; i > -1; i--) {
Res_0 = Res2 mod n;
// "Square"
Res_1 = Res_0 * c mod n;
// "Multiply"
Res = Res_(bi);
}

```

Figure 3.13.: The fourth part of the SPA plug-in

- The next part of the SPA view: the table of "Square and Multiply" is one of the most important parts of SPA plug-in. Together with the visual diagram displayed below, the visualization of the original algorithm and its countermeasure is realized. In this table group all temporary results and relevant data in the process of the both algorithms are listed, including the initial inputs of parameter, each step of the execution of "squaring and multiplication" operations, and the final output result of RSA algorithm. A visual example has been given in Section 3.4.
- The last part of view is called " Visualized Power Traces", which is located in the lower right of the main view. This is the most important part of the SPA plug-in. The power traces are displayed visually according to the recorded

data in the table above. With the visualization the users realize and understand specifically the weak point of "squaring and multiplication", the process of SPA attack against RSA, and the principle of the countermeasure "squaring and multiplication always". An example of vulnerable "insecure execute" visualization and its corresponding specification is given in Figures 3.14 and 3.15 as follows:

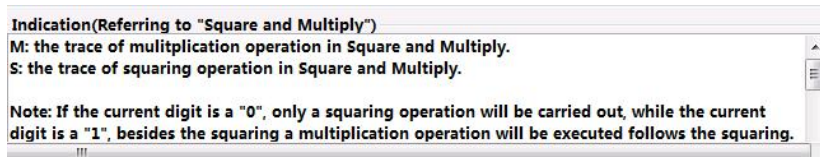


Figure 3.14.: The specification of the "squaring and multiplication"

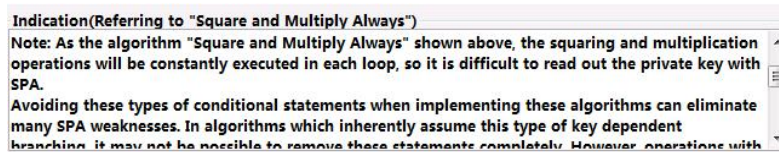


Figure 3.15.: The specification of the "squaring and multiplication always"

For the visualization of "secure Execute" and its corresponding specification see Figures 3.7, 3.15, and Section 3.4.

3.7 The Implementation of the SPA Plug-in

The "SPAView" is the most important class of this plug-in, in which the view of visual representation for SPA attack is implemented. The functionalities and visualization are capsuled in this class. The "SPAView" realizes also the interface "Constants.java" in the same package, which is aimed to list all constant strings like unicode characters, pseudo codes of algorithm, and string paragraphs, which is used in the class "SPAView.java". In such a design approach any change of the constants is easier to locate and perform. Besides that it is also simpler to transfer the original contents into other languages.

The following are some typical program segments of "SPAView":

- Just as in Section 3.5 introduced the first two parts of "SPAView" are the description of RSA and "squaring and multiplication" algorithms, in which some sub- and superscripts, just like $c \equiv m^e \pmod n$ and $Res = Res^2 \pmod n$ are defined. We implement the representation of sub- and superscripts in text field with the statement: "styles[0].rise = fontData[0].getHeight()/2;", which realizes the superscript by raising half height of the certain character, and its position is defined with the statement "int[] ranges = new int[]{452, 1, 493, 1};". Analog to realization of superscript by raising the position of certain character, we can also represent the subscript by lowering the position with the statement as "newStyles[2].rise = -fontData[0].getHeight()/2;". In this way the subscript in text like $if(b_i = 1)$ can also be shown correctly.

```

1      Font newFont = new Font(parent.getDisplay(), fontData[0].getName(),
2                                fontData[0].getHeight(), fontData[0].getStyle());
3      StyleRange[] styles = new StyleRange[2];
4      styles[0] = new StyleRange();
5      styles[0].fontStyle = SWT.BOLD;
6      styles[0].font = newFont;
7      styles[0].rise = fontData[0].getHeight()/2;
8      styles[1] = new StyleRange();
9      styles[1].fontStyle = SWT.BOLD;
10     styles[1].font = newFont;
11     styles[1].rise = fontData[0].getHeight()/2;

```

```
12 | int[] ranges = new int[] {452, 1, 493 ,1};
```

- Some segments in text fields are marked in colors. For example, in the text field of part 2 ("squaring and multiplication" algorithm) the vulnerable if-statement is marked in red, which means these condition statements are insecure to SPA attack. RGB is the abbreviation of the RGB (red green blue) color model and (252, 99, 103) presents the value of the chosen color, which can be previewed and defined in drawing board. The first and fourth parts of "SPAView" are also marked in yellow and green, which states separately warning and security.

```
1 | RGB c = new RGB(252,99,103);
2 |     squareAndMultiAlgText.setLineBackground(22, 5,
3 |         new Color(parent.getDisplay(),c));
4 |     squareAndMultiAlgText.setStyleRanges(newRanges,newStyles);
5 |
```

- Three drop-down lists are defined in the group "Parameter of RSA" and the following code is used to define the range of the selectable elements in drop-down lists. The available value of "Private Key" is limit in a range of all prime numbers between 100 and 200. And with the while-statement all primes are added into the list.

```
1 | final int[] primeDataExponent = { 101,103,107,109,
2 | 113,127,131,137,139,149,151,157,163,167,173,179,
3 | 181,191,193,197,199 };
4 | int dataLengthExponent = primeDataExponent.length;
5 | int dataElementIndexExponent = 0;
6 | while(dataLengthExponent>0){ exponent.add(String.
7 |     valueOf(primeDataExponent[dataElementIndexExponent]));
8 |     dataLengthExponent--;
9 |     dataElementIndexExponent++;
10 | }
11 |
```

- According to the RSA algorithm, the prime P and Q are distinct integer numbers. So no matter which is selected, the same number should be unavailable in the other drop-down list of Q or P. This function is implemented as follows, a "selectionListener" is appended on "qSelectCombo", if a prime number in "qSelectCombo" is selected, the same prime number will be automatically deleted in "pSelectCombo". Similarly there is also a "selectionListener" added on the "pSelectCombo" to achieve the same function.

```
1 | qSelectCombo.addSelectionListener(new SelectionAdapter() {
2 |     public void widgetSelected(final SelectionEvent e) {
3 |         q_selected = Integer.parseInt(qSelectCombo.
4 |             getItem(qSelectCombo.getSelectionIndex()));
5 |         for(int i = 0; i<pSelectCombo.getItemCount();i++){
6 |             if(pSelectCombo.getItem(i).equals(String.
7 |                 valueOf(q_selected))){
8 |                 pSelectCombo.remove(i);
9 |                 break;
10 |             }
11 |         }
12 |         if(p_selected!=0&&q_selected!=0){
13 |             cue_label.setText("");
14 |             mod.setText(String.valueOf((p_selected)*
15 |                 (q_selected)));
16 |         }
17 |     }
18 | });
```

- In the group of "Parameter of RSA" all text fields (input tags and drop-down lists) are appended with event listeners which are used to check whether the inputs and selection satisfy the corresponding requirements. For instance, the following code is used to set a "ModifyListener" on the input text field "Cipher text". When the condition in the if-statement is satisfied, both buttons of "insecure Execute" and "secure Execute" will be enabled. A "ModifyListener" is appended to the drop-down lists of the "Private key", "P", and "Q". Besides the "ModifyListener", a "VerifyListener" is attached to the "basis", too. The functionality of "VerifyListener" ensures that only numbers within 9 digits are allowed as the value of inputs.

```

1      basis = new Text(parameterOfRSAGroup, SWT.BORDER);
2      basis.setTextLimit(9);
3      basis.setToolTipText(TOOL_TIP_TEXT_BASIS);
4      basis.setBounds(10, 45, 75, 25);
5      basis.addModifyListener(new ModifyListener() {
6          public void modifyText(final ModifyEvent e) {
7              if (!basis.getText().equals("") && (exp_selected != 0)
8                  && (!mod.getText().equals(""))) {
9                  executeButton.setEnabled(true);
10                 samaExecuteButton.setEnabled(true);
11             }
12         }
13     });
14
15     basis.addVerifyListener(new VerifyListener() {
16         public void verifyText(VerifyEvent e) {
17             Pattern pattern = Pattern.compile("[0-9]{1,9}$");
18             Matcher matcher = pattern.matcher(e.text);
19             if (matcher.matches())
20                 e.doit = true;
21             else if (e.text.length() > 0)
22                 e.doit = false;
23             else
24                 e.doit = true;
25         }
26     });
27

```

- After all parameters of the "squaring and multiplication" are specified, both the buttons for "insecure Execute" and "secure Execute" are enabled. Each execution button is appended with a "MouseListener", after clicking down and raising up the mouse from the button, the execution will be fired. All temporary results are saved in the table in part 5 and a visual representation of the power traces for the process will be created in part 6.

```

1      executeButton.addMouseListener(new MouseAdapter() {
2
3          public void mouseUp(final MouseEvent e) {
4              ...
5              ...
6          }})
7
8      samaExecuteButton.addMouseListener(new MouseAdapter() {
9          public void mouseUp(final MouseEvent e) {
10             ...
11             ...
12         }})
13

```

- In order to guarantee a good understanding of the algorithms: "squaring and multiplication" and "squaring and multiplication always", all temporary results are saved in the table of part 5. This table is made of three columns: all inputs, the step of process, and outputs are presented in the first column; in the second column the temporary result after squaring for each turn is saved and the temporary result after multiplication is saved in the last column.

```
1      recordTable = new Table(calculateTableGroup , SWT.BORDER);
2      recordTable.setBounds(5, 20, 650, 320);
3      recordTable.setLinesVisible(true);
4      recordTable.setHeaderVisible(true);
5      final TableColumn roundCol = new TableColumn(
6          recordTable , SWT.NONE);
7      roundCol.setWidth(180);
8      roundCol.setText(FIRST_COLUMN_IN_TABLE);
9      final TableColumn resSquareCol = new TableColumn(
10         recordTable , SWT.NONE);
11     resSquareCol.setWidth(210);
12     resSquareCol.setText(SECOND_COLUMN_IN_TABLE);
13     final TableColumn resMultiCol = new TableColumn(
14         recordTable , SWT.NONE);
15     resMultiCol.setWidth(260);
16     resMultiCol.setText(THIRD_COLUMN_IN_TABLE);
17     new TableCursor(recordTable , SWT.NONE);
```

The realization of "squaring and multiplication" algorithm is implemented in class "SquareandMultiply.java", which is used to calculate and return the result of the vulnerable algorithm.

4 The Differential Power Analysis Plug-in

"Differential power analysis" is one kind of differential side channel attacks which exploit the correlation between the data and the power consumption of the cryptographic algorithm. Because the correlation between data and power consumption is usually unclear and disturbed, statistical methods are regarded an efficient way to filter the noise and enlarge the correlation needed for signal analyzing.

In this thesis the DPA plug-in is implemented to present the procedure of DPA attack against the vulnerable elliptic curve cryptographic system. Three countermeasures are introduced.

The outline of the chapter is as follows: first of all, idea of DPA attack is described in Section 4.1. Then some basic definitions of elliptic curve are demonstrated in Section 4.2, including descriptions of prime field, point addition, point doubling, and other issues of elliptic curve. After that the vulnerability of "doubling and addition" algorithm is shown in Section 4.3. In Section 4.4 three countermeasures against DPA attack are introduced. After introduction of countermeasures in Section 4.5 the design of DPA plug-in is described. In Section 4.6 the functionality of the DPA plug-in is explained step by step. And in Section 4.7 some program segments are chosen to explain how the implementation is realized.

4.1 The Idea of the DPA Attack

In contrast to SPA attack, DPA attack depends on lots of samples of power consumption traces. With the help of statistical methods the DPA attack can extract the tiny details in power traces which are normally unable to be observed with SPA. With statistical methods like averaging technique, the adversary may filter or decrease the noise with different inputs of samples to the cryptographic system. The basic idea of DPA attack is to suppose the value of a certain bit of the secret key and then per statistical averaging methods to verify the validity of the assumption which might correlate to the power traces.

With Figure 4.1 the idea of DPA attack is explained more precisely.

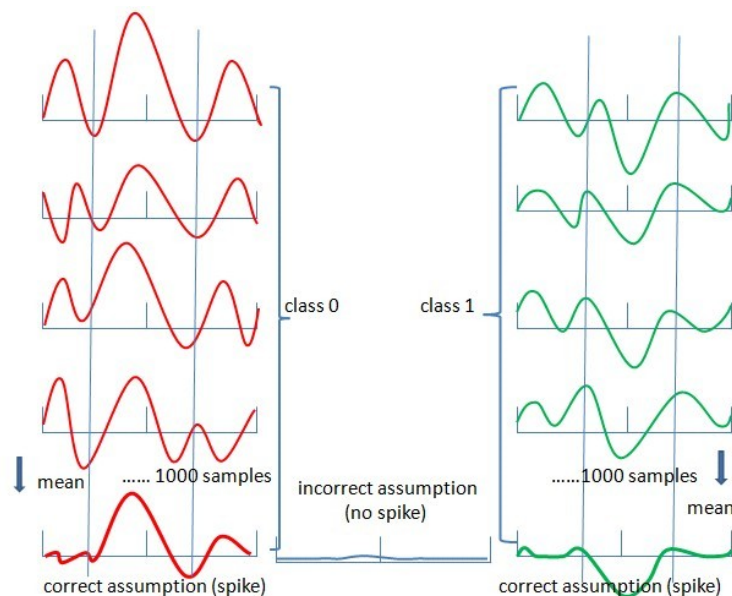


Figure 4.1.: The idea of the DPA attack

The procedure of DPA attack is divided as follows:

- Firstly assume the value of a certain bit of the secret key.

- After that choose an amount of samples based on the assumption.
- According to the value of the assumption, the samples can be divided into two classes, for instance, the class 0 states the value of assumption is "0", and the class 1 states the value "1".
- Input the samples of either class into the cryptographic system.
- Observe and classify the corresponding power traces.
- Record the power consumption traces of samples and analyze them with averaging statistical software.
- If the assumption is correct, that means the average power consumption trace is correlated to the guessed value of assumption. Then a spike can be observed; Otherwise if the average power consumption is not correlated with the guessed value, then no spike occurs. Since the value of assumption might be either "0" or "1", from the incorrect result of assumption the real value of the bit is determined.
- For those algorithms in which the bit string sequence of the private key is related to the power traces of decryption or signature function, all bits of the private key can be recovered step by step with DPA attack.

In the scalar multiplication execution of elliptic curve, the secret key is correlated with the power consumption traces of "doubling" and "addition" operations. Therefore the "doubling and addition always" algorithm is potential suitable for DPA attack. More details will be given in Section 4.2.

4.2 The Elliptic Curve and the "Doubling and Addition" Algorithm

The visualization of DPA attack is based on elliptic curve cryptographic system. For more on elliptic curve cryptography (ECC) see reference [ELL09]. We briefly present here some basic definitions and formulas about elliptic curve cryptographic system as follows:

Definition 1 *The elliptic curve discrete logarithm problem (ECDLP): For two points P and Q on an elliptic curve, which satisfy the equation $Q = kP$, k is the logarithm of Q to the base P . In elliptic curve cryptographic system (ECC) the equation $Q = kP$ is called scalar point multiplication, and the logarithm k is also named scalar multiplier. The scalar point multiplication is consider the main cryptographic operation in ECC. The security of ECC relies on the difficulty of solving the ECDLP. If P and Q are given, it is computationally intractable to find k if this is a large number. Actually the equation $Q = kP$ has also infinitely many solutions of the form $k + |P| * n$ (where $|P|$ is the order of point P , and $n \in \mathbb{Z}$).*

Definition 2 *ECC: An ECC operates over points on an EC. The elliptic curves used in ECC are normally defined in two fields:*

- *the prime field $GF(p)$: p is a prime and all coordinate values on the elliptic curve are the elements of p .*

The form of elliptic curve in prime field is as follows:

$$E: y^2 = x^3 + ax + b, (a, b \in GF(p), 4a^3 + 27b^2 \neq 0, p > 3)$$

e.g. $GF(11)$ means the prime field group $\langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \rangle$ and in $GF(11)$ the EC equation $y^2 = x^3 - x + 1$ is isomorphic to $y^2 = x^3 + 10x + 1$

- *The binary field $GF(2^m)$: here each element is a binary polynomial of degree m which can be represented as an m -bit string for each coefficient is either "0" or "1". All the coordinate values on the elliptic curve are defined in an m -bit binary string form [ELL04].*

The form of elliptic curve on binary field is as follows:

$$E: y^2 + xy = x^3 + ax^2 + b, (a, b \in GF(2^m), b \neq 0)$$

In this thesis the prime field $GF(p)$ is selected and implemented.

Definition 3 *The parameters a, b of the EC equation: the parameters of elliptic curve, the values of a and b are set in $GF(p)$, furthermore a and b satisfy the inequation: $4a^3 + 27b^2 \neq 0$*

Definition 4 *The infinity Point O of EC: group identity, which is a point at infinity distance far away from the elliptic curve.*

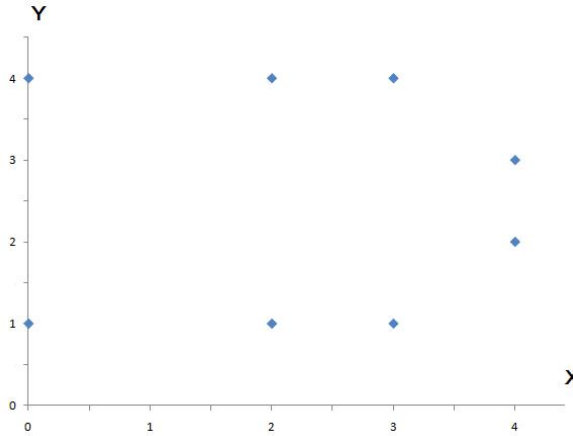


Figure 4.2.: All points on $EC : y^2 = x^3 + x + 1$ in $GF(5)$

Definition 5 The order of EC: the number of points on the elliptic curve + 1 (the infinity point O), and presented as $|EC|$.
 e.g. for the $EC : y^2 = x^3 + x + 1$ in $GF(5)$ there are altogether 8 points on it: $(0,1), (0,4), (2,1), (2,4), (3,1), (3,4), (4,3), (4,2)$ including the infinity point O , the order of this elliptic curve is $|EC| = |(0,1), (4,2), (2,1), (3,4), (3,1), (2,4), (4,3), (0,4), O| = 9$ (see Figure 4.2).

Definition 6 The order k of a Point P : the minimum number k in $GF(p)$ which satisfies $kP =$ infinity point, is called the order of Point P , and presented as $|P|$.

e.g. for the Point $P = (0,1)$ on $EC : y^2 = x^3 + x + 1$ in $GF(5)$, $2P = (4,2), 3P = (2,1), 4P = (3,4), 5P = (3,1), 6P = (2,4), 7P = (4,3), 8P = (0,4), 9P =$ infinity point O , so the order of $P(0,1)$ on this EC is $|P| = |(0,1), (4,2), (2,1), (3,4), (3,1), (2,4), (4,3), (0,4), O| = 9$.

Definition 7 The mathematical form of EC: let $GF(p)$ be a prime finite field, p is a prime number, and let $a, b \in GF(p)$ satisfy the inequation $4a^3 + 27b^2 \neq 0 \pmod{p}$. Then an elliptic curve $EC(GF(p))$ over $GF(p)$ defined by the parameters $a, b \in GF(p)$ consists of the set of solutions or points $P = (x,y)$ for $x, y \in GF(p)$ to the equation: $y^2 \equiv x^3 + ax + b \pmod{p}$ together with the infinity point O . The equation $y^2 \equiv x^3 + ax + b \pmod{p}$ is called the defining equation of the elliptic curve $(GF(p))$. For a given point $P = (x, y)$, x is called the x -coordinate of P , and y is called the y -coordinate of P .

Definition 8 The point multiplication: a point P on the elliptic curve is multiplied with a scalar multiplier k and then obtain another point Q on the same elliptic curve, such operation $Q = kP$ is called the point multiplication on EC. The point multiplication is achieved with two basic operations of elliptic curve:

- Point doubling: adding two points P_1 and P_2 to obtain another point Q on the curve, $Q = P_1 + P_2$
- Point addition: adding a point P_1 to itself to obtain another point Q , $Q = 2P_1$

e.g. assume that P is a point on an elliptic curve and k is a scalar multiplier that is multiplied with the point P to obtain another point Q on the curve, i.e. $Q = kP$.

For instance: if $k = 53$ then $kP = 53P = (110101)_2 P = 2(2(2(2(2P + P)) + P)) + P$. By applying point doubling and point addition repeatedly, it is possible to get the result. Such a method to execute the point multiplication of EC is called "doubling and addition" algorithm.

The geometrical representation¹ of point addition and point doubling are as follows in Figures 4.3 and 4.4 [ELL04].

For the two points A and B on the $EC: y^2 = x^3 - 4x + 4$ shown in Figure 4.3, the point addition operation is:

¹ For presentation purpose the figures of elliptic curve in real field is chosen to be illustrated, the geometrical representation of prime field is similar.

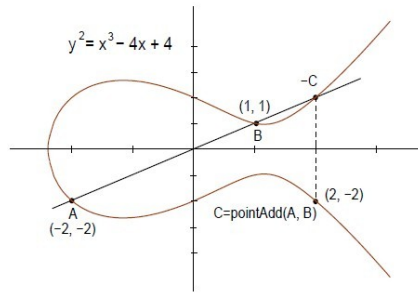


Figure 4.3.: A geometrical example for point addition²

1. If $A \neq -B$, then a line drawn through the points A and B intersects the EC at exactly one point $-C$. The reflection of point $-C$ with respect to x -axis is the point C , which is the result of the point addition of A and B .
2. If $A = -B$, then the line through both of the points intersects the EC at infinity distance, regarded intersects at the infinity point O . So in this case, the result of the addition operation of A and B is the infinity point O .

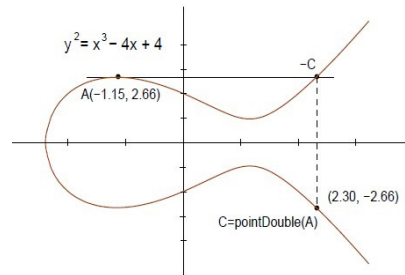


Figure 4.4.: A geometrical example for point doubling

For the point A on the EC: $y^2 = x^3 - 4x + 4$ shown in Figure 4.4, the point doubling operation is:

1. If the y coordinate of the point A is not "0", then the tangent line at A intersects the EC at exactly one point $-C$. The reflection of the point $-C$ with respect to x -axis give the point C , which is the result of the point doubling operation of the point A .
2. If the y coordinate of the point A is "0", then the tangent at point A intersects the EC also at the infinity point O . Therefore $2A = O$, when $y_A = 0$.

The point doubling and point addition formulas on elliptic curve $EC(GF(p))$ are given below:

Known: $P_1 = (x_1, y_1), P_2 = (x_2, y_2), P_3 = (x_3, y_3) \in EC(GF(p))$

Then:

1. The point doubling formula for $P_3 = P_1 + P_1 = 2P_1$, $\frac{3x_1^2 + a}{2y_1}$ in the formula is the slope of the tangent line of the point P_1 (see Figure 4.3):

$$P_3 = \begin{cases} x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \\ y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 \end{cases}$$

² Both of Figures 4.3 and 4.4 are picked from [ELL04].

2. The point addition formula for $P_3 = P_1 + P_2$, ($P_1 \neq P_2$), $\frac{y_2 - y_1}{x_2 - x_1}$ in the formula is the slope of the line of points P_1 and P_2 (see Figure 4.3):

$$P_3 = \begin{cases} x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \\ y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 \end{cases}$$

Definition 9 The division in prime field $GF(p)$: the division a/b in prime field p is defined as $a/b \pmod{p} = a * b^{-1} \pmod{p}$. here b^{-1} is the multiplicative inverse of b on prime field p . The division operation is used in the execution of doubling and addition operations.

4.3 The Vulnerability of the "Doubling and Addition always" Algorithm

The vulnerability of "doubling and addition" algorithm is similar to the weakness of "squaring and multiplication" algorithm. The condition statement (if-statement) can be regarded as the defect of the algorithm, which may leak the information of the secret key. With the help of a dummy "addition" operation, the original "doubling and addition always" algorithm is robust enough to SPA attack. However, the countermeasure by adding a dummy operation is still vulnerable to DPA attack. With statistic methods an adversary enlarges the tiny differences, filters the noise in the measurement and corrects errors by using the averaging analysis method. We present a concrete instance which explains step by step how a secret key is recovered with DPA attack.

Principle: power consumption traces of the computation for "doubling and addition always" are correlated to a specific bit of Q . If no correlation for a certain kP is observed, that states the point $Q = kP$ has not been computed.

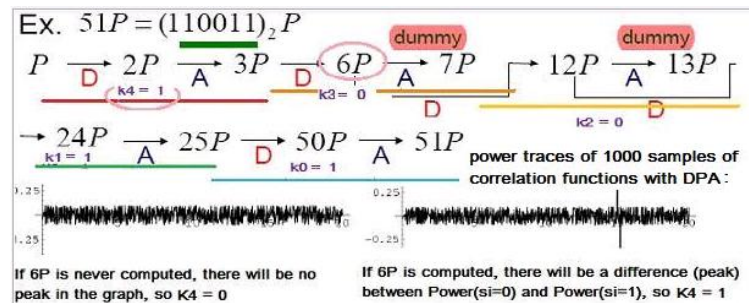


Figure 4.5.: An example for the DPA attack against "doubling and addition always"

As the instance in Figure 4.5 above described:

$Q = kP = 51P = (110011)_2 P$. The second most significant bit k_{n-2} of k can be recovered by computing and analyzing the correlation between power consumption and any specific bit of the binary representation of $6P$. We gather 1000 power consumption traces' samples of computing $6P$, and let S_i be any specific bit of $6P$. We use the correlation function: $g(t) = \text{Power}(S_i = 0) - \text{Power}(S_i = 1)$ to examine the relation between the certain bit of $6P$ and the power traces of Q . If $6P$ is relate to simulated correlation function $g(t)$, a peak (referring to the right power consumption traces in Figure 4.5, which presents the power consumption traces from 1000 samples of correlation functions for DPA) is observed corresponding to the computation of $6P$, otherwise if there is no peak, the second significant bit is 0. The reason for that is clear, if $k_4 = 1$, $6P$ is computed as $Q[0] = 2 * 3P = 6P$, otherwise by $k_4 = 0$, $2 * 2P$ will be carried out during the doubling operation instead of $6P$. Therefore if the second significant bit is 0, the power consumption traces of Q are not correlated to the binary representation form of any specific bit of $6P$. If $6P$ is not computed, no peak will be observed in the correlated power consumption traces. In the same way all following bits k_3, k_2, \dots, k_0 can be recursively recovered step by step.

4.4 The Countermeasures against the DPA Attack

In order to protect the EC cryptographic system from DPA attack, there are three methods which are commonly applied as the countermeasures against the DPA attack:

- Randomization of the scalar multiplier k :

If the scalar multiplier k is randomized in each turn of execution, the attacker cannot recover the private key per comparison and analysis of the differences between the power traces of "doubling and addition always" operations.

The process of randomizing k is described as follows:

At first we choose a random factor in group $[1, p-1]$ (p is the prime field of the curve), then a new randomized scalar multiplier $k' = k + r * |P|$ will be calculated, since $|P| * P$ can be regarded as the infinity point O of the chosen elliptic curve, we get:

$$Q' = (k + r * |P|) * P = k * P + r * (|P| * P) = kP + r * 0 = Q$$

With such a method even for the same private key k , the power traces are different. The relation between certain input parameter k and corresponding power trace during operation process is broken.

- Randomization of the initial point P :

The second way to protect "doubling and addition always" against DPA attack is the randomization of the initial point P . If the initial point P is randomized in each turn of execution, it is also difficult to recover the private key through analysis of the correlation between a certain kP and its power trace.

The process of randomizing the initial point P is as follows:

First we choose randomly a point R on the same elliptic curve which is different from the initial point P . Then we calculate $R + P = P'$ as the new initial point for the cryptographic system. After "doubling and addition always" algorithm we get $Q' = kP' = k(P + R) = kP + kR = Q + kR$ as the result. And any point on the elliptic curve satisfies the rule: if $S = (x, y)$ then $-S = (x, -y)$, that means $Q = Q' - kR = Q' + (-kR)$, in this way we can calculate the result Q but concealing the real initial point P , therefore is the correlation between power traces and the initial input P also concealed.

- Randomization of the isomorphic curve E :

The third countermeasure is named randomization of the isomorphic curve. The basic idea of this method is as follows: we create randomly an isomorphic curve EC' from the original curve EC . Then we process the "doubling and addition always" algorithm on EC' , we get the result Q' based on EC' . After that we try to recover the original result Q from Q' . In this way the adversary cannot find the correlation from the DPA attack, since the elliptic curve for each turn is always distinct.

The process is as follows:

First we choose randomly a number R in prime field $[1, p-1]$, and then we calculate $a' = R^4 a, b' = R^6 b P' = (R^2 X_p, R^3 Y_p)$. With new a', b', P' we get a new isomorphic curve: $EC' : y^2 = x^3 + a'x + b'$. After that we compute $Q' = kP'$ on the new isomorphic curve. $Q' = (X'_q, Y'_q)$. Finally we recover Q with the formula $Q = (R^{-2} X'_q, R^{-3} Y'_q)$. In this way the countermeasure breaks the correlation between EC points and their power traces.

Some examples are given to explain the countermeasures more precisely:

- First a random EC for the original "doubling and addition always" algorithm is selected.

The example of appointing the parameter is given in Figure 4.6:

As in the figure displayed, the selected parameters p , a , and b are 307, 2, and 3 respectively. The determined Elliptic Curve is:

$$E : y^2 = x^3 + 2x + 3, GF(307)$$

The order of this curve is $|E| = 310$. Among all 309 points on the elliptic curve (except the infinity point O), point $(31,154)$ is chosen as initial point P and its order is 10. Except the situation that the scalar multiplier of P is 0 or 1, among the other 8 numbers (2 to 9), k is randomly selected as 6. After the appointing of all parameters, the "execute" button is enabled and the user can animate the process of "doubling and addition always" method.

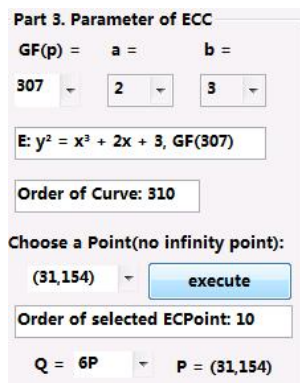


Figure 4.6.: An example for "ECC algorithm"

Below "Parameter of ECC" is the area in which the process and principle of "DPA against ECC" and all parameter of three kinds of countermeasures are introduced and listed. In the case of explanation of the principle and process of "DPA against ECC", the content of the vulnerable "squaring and addition always" method is displayed in Figure 4.7:

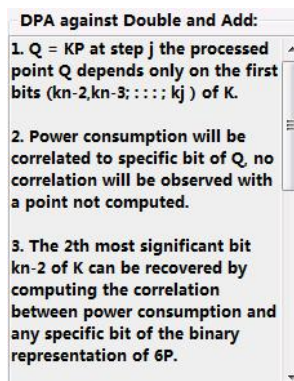


Figure 4.7.: The group "DPA against ECC"

The process of execution will be listed in the right located table, referring to the figure 4.8:

By insecure "doubling and addition always":

Here we describe the insecure "doubling and addition always" algorithm. As described in Figure 4.8, the binary form of the scalar multiplier k is "110", according to the description of the method "doubling and addition always" in Figure 4.7, the loop will be carried out twice:

1. For the first loop, since the second bit is "1", the point after addition $(299, 150)$ will be accepted;
2. For the second loop, since the third bit is "0", the point after doubling $(239, 73)$ is taken as the end result.

So the result for this instance $Q = 6P = (239, 73)$.

| Part 4. Double and Add (from left highest to right lowest valuable bit) | | |
|---|--------------------------------|------------------------------------|
| Round Counter(left to right) | Result after Double | Result after Add |
| Input: | $Q[0] = 2Q[0]$ | $Q[1] = Q[0] + P$ |
| k in binary form: 110 | | |
| Process: | | |
| 2. highest bit = 1 | $Q[0] = 2Q[0] = 2P = (86,197)$ | $Q[1] = Q[0] + P = 3P = (299,150)$ |
| 3. highest bit = 0 | $Q[0] = 2Q[0] = 6P = (239,73)$ | $Q[1] = Q[0] + P = 7P = (299,157)$ |
| Q = (239,73) | | |

Figure 4.8.: An example for "doubling and addition always" in the table

- Now an example for the first countermeasure "Randomizing Scalar Multiplier" is given.

The procedure for the presentation of a countermeasure is divided into three steps:

1. Firstly the switch from "doubling and addition always" to its countermeasures is realized by clicking the button in Figure 4.9. After clicking, the user can select which kind of countermeasures to be processed and visualized.

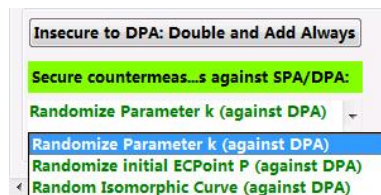


Figure 4.9.: The selection menu of the DPA countermeasures

The user is also able to transfer from the countermeasure to the original vulnerable "doubling and addition always" method by clicking the button below (Figure 4.10):

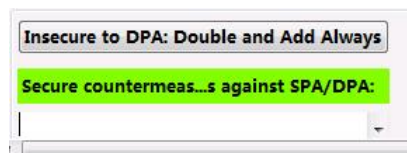


Figure 4.10.: The selection menu of the "doubling and addition always" algorithm

2. Secondly, after choosing the countermeasure as "randomizing scalar multiplier" in the selection menu, the user can enable the process of the countermeasure by clicking the button "execute" in the group "Parameter of ECC", then a random scalar multiplier will be created as Figure 4.11 describes below.

The randomly selected factor r is "1", then with the formula $k' = (k + |P| * r) \bmod p = (6 + 10 * 1) \bmod 307 = 16$, and k' in binary form is "10000".

3. Then $Q = k'P$ is calculated and the temporary results are listed in the table as seen in Figure 4.12.

Since k' in binary is "10000", it will be processed four times and for each loop the temporary results after the doubling operation will be accepted and the final result stays the same as in the original method: (239,73). Because for each execution, the scalar multiplier is always randomly selected, the DPA attack by observation of the averaging power consumption traces is defended.

- The example for the second countermeasure "randomizing the initial point" is given as follows:

1. Similar to the countermeasure "randomizing the scalar multiplier", the user must firstly select the second countermeasure and click the "execute" button to enable the process.

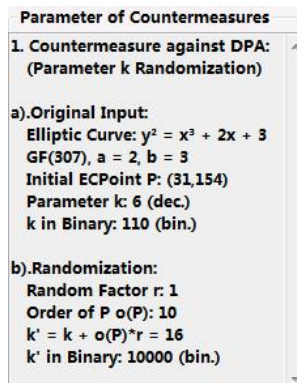


Figure 4.11.: The description for the method "randomizing scalar multiplier"

| Round Counter(left to right) | Result after Double | Result after Add |
|------------------------------|---------------------------------|-------------------------------------|
| Input: | $Q[0] = 2Q[0]$ | $Q[1] = Q[0] + P$ |
| $k': 10000$ (bin.) | | |
| k in binary form: 110 | | |
| Process: | | |
| 2. highest bit = 0 | $Q[0] = 2Q[0] = 2P = (86,197)$ | $Q[1] = Q[0] + P = 3P = (299,150)$ |
| 3. highest bit = 0 | $Q[0] = 2Q[0] = 4P = (239,234)$ | $Q[1] = Q[0] + P = 5P = (306,0)$ |
| 4. highest bit = 0 | $Q[0] = 2Q[0] = 8P = (86,110)$ | $Q[1] = Q[0] + P = 9P = (31,153)$ |
| 5. highest bit = 0 | $Q[0] = 2Q[0] = 16P = (239,73)$ | $Q[1] = Q[0] + P = 17P = (299,157)$ |
| $Q = (239,73)$ | | |

Figure 4.12.: An example for "randomizing scalar multiplier" in the table

2. Then as in Figure 4.13 described, a random point $R: (175,212)$ on the same elliptic curve will be chosen and the point $P + R: (191,225)$ will be used as the new initial point P' . According to the point multiplication operation, it is not difficult to execute the results:

$$S = kR = 6R = (245, 107) \text{ and } Q' = k(P + R) = 6(P + R) = (128, 149)$$

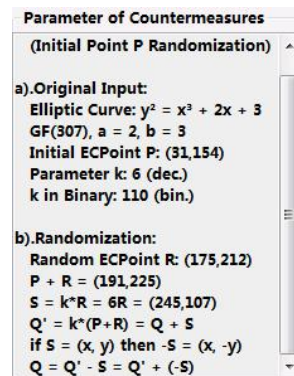


Figure 4.13.: The description for the method "randomizing initial point"

3. The process of execution of Q' is displayed in Figure 4.14:

For $Q' = k(P + R) = kP + kR = Q + S$, results $Q = Q' - S$, and for elliptic curve the subtraction operation $Q' - S$ is equal to $Q + (-S)$ and for $S = (x, y)$ satisfies $-S = (x, -y)$, then we get $Q = Q' + (-S) = (128, 149) + (245, -107)$. After the point addition operation we get $Q: (239,73)$. Such a method is called "randomizing initial point" which can also be used to break the averaging sampling by the DPA attack.

- The last example for the countermeasure "randomizing isomorphic curve" is described below:

| Part 4. Double and Add (from left highest to right lowest valuable bit) | |
|---|---------------------------------------|
| Round Counter(left to right) | Result after Double |
| Input: | Q[0]' = 2Q[0]' |
| Random ECPoint R: (175,212) | |
| P + R = (191,225) | |
| S = k*R = 6R = (245,107) | |
| k in binary form: 110 | |
| Process: | |
| 2. highest bit = 1 | Q[0]' = 2Q[0]' = 2(P + R) = (106,203) |
| 3. highest bit = 0 | Q[0]' = 2Q[0]' = 6(P + R) = (128,249) |
| Output: | |
| Q' = k*(P+R) = Q + S = (128,249) | |
| S = (x, y) then -S = (x, -y) = (245,-107) | |
| Q = Q' - S = Q' + (-S) | |
| Q = (128,249) + (245,-107) = (239,73) | |

Figure 4.14.: An example for "randomizing initial point" in the table

1. Firstly select the countermeasure "randomizing isomorphic curve" and click the "execute" button.
2. Secondly in order to create an isomorphic elliptic curve EC', a factor "13" is randomly selected. As in Figure 4.15 displayed, the random factor r is used to produce the parameters a', b', and new initial point P' of EC':

| Parameter of Countermeasures |
|---|
| 3. Countermeasure against DPA: (Isomorphic Curve Random.) |
| a).Original Input: Elliptic Curve: $y^2 = x^3 + 2x + 3$ GF(307), a = 2, b = 3 Initial ECPoint P: (31,154) Parameter k: 6 (dec.) k in Binary: 110 (bin.) |
| b).Randomization: Random Factor r = 13 $a' = a * r^4 = 20$, $b' = b * r^6 = 158$ $P' = (r^2 * X_p, r^3 * Y_p) = (20, 24)$ EC': $y^2 = x^3 + a'x + b'$ $= x^3 + 20x + 158$ |

Figure 4.15.: The description for the method "randomizing isomorphic curve"

$$a' = a * r^4 = 20, b' = b * r^6 = 158, P' = (r^2 * X_p, r^3 * Y_p) = (20, 24)$$

Then we get an isomorphic curve EC' as:

$$y^2 = x^3 + a'x + b' = x^3 + 20x + 158$$

3. After that the result of $Q' = kP' = 6P'$ on the new elliptic curve will be executed, the temporary results are presented in Figure 4.16:

After two loops, we get the result $Q' = (174,127)$. According to the description of "randomizing isomorphic curve", with the formula $Q = (r^{-2}X_{q'}, r^{-3}Y_{q'})$, the Q can be recovered from $Q' = (X_{q'}, Y_{q'})$.

As shown in the table of Figure 4.16, the final result of Q remains the same (239,73).

In this section the idea of three kinds of countermeasures against the DPA attack is introduced and for better understanding of each countermeasure concrete instances are also offered.

4.5 The Design of the DPA Plug-in

The class structure of the DPA plug-in is described in Figure 4.17. The DPA plug-in is made up of four packages:

- org.jcryptool.algorithm
- org.jcryptool.visual.sidechannelattack

| Round Counter(left to right) | Result after Double | Result after Add |
|---------------------------------------|------------------------------------|--|
| Input: | $Q[0]' = 2Q[0]'$ | $Q[1]' = Q[0]' + P$ |
| k in binary form: 110 | | |
| Process: | | |
| 2. highest bit = 1 | $Q[0]' = 2Q[0]' = 2P' = (105,246)$ | $Q[1]' = Q[0]' + P' = 3P' = (183,139)$ |
| 3. highest bit = 0 | $Q[0]' = 2Q[0]' = 6P' = (174,127)$ | $Q[1]' = Q[0]' + P' = 7P' = (183,168)$ |
| Output: | | |
| $Q' = (Xq^*r^2, Yq^*r^3) = (174,127)$ | | |
| $Q = (174^*13^2, 127^*13^3)$ | | |
| Xq = 239, Yq = 73 | | |
| Q = (239,73) | | |

Figure 4.16.: An example for "randomizing isomorphic curve" in the table

- org.jcryptool.visual.sidechannelattack.dpa.views
- com.swtDesigner

We describe the classes of these packages as follows:

- Packages "org.jcryptool.visual.sidechannelattack" and "com.swtDesigner" are created automatically by the SWT and RCP environment. Classes located in these packages like "SWTResourceManager.java" and "DPAPugin.java" have the same functions like the similar classes in the SPA plug-in.
- Packages "org.jcryptool.algorithm" and "org.jcryptool.visual.sidechannelattack.dpa.views" are the core components.

Package "org.jcryptool.algorithm": The classes "ECCAdd.java", "ECCDouble.java", "ECCMultiply.java", "ECCOrderAndPoints.java" and "RandomFactorCreator.java" are designed to implement the function of "doubling and addition always" algorithm and the corresponding countermeasures. They realize the basic function of elliptic curve cryptographic system, including point addition, point doubling, point multiplication, the execution of the order of the initial point, the selected elliptic curve, etc..

Package "org.jcryptool.visual.sidechannelattack.dpa.views": the last two in this package located classes "DPAView.java" and "Constants.java" are the main classes of the whole plug-in. The original algorithm and its countermeasures are visualized in this package.

Some code segments for the implementation of the classes in these packages are given in Section 4.7.



Figure 4.17.: The structure of the DPA plug-in

The View of the DPA plug-in is displayed schematically in Figure 4.18. There are altogether 24 SWT objects.

All individual elements of the user interface are listed in Table 4.1. The functionality for each column is similar to the table of the SPA plug-in (see Section 3.5).

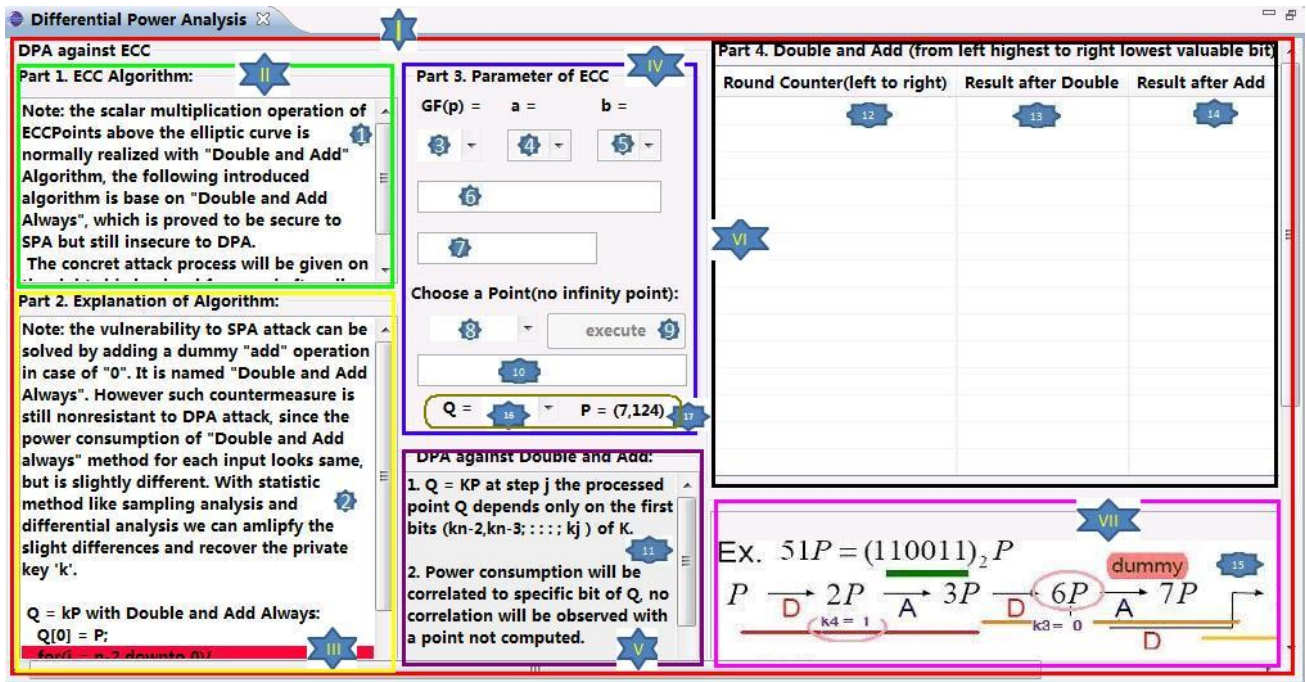


Figure 4.18.: The user interface of the DPA plug-in

For better understanding of the structure, a UML activity diagram (Figure 4.19) is also offered for the DPA plug-in.

4.6 The Functionality of the DPA Plug-in

According to the different functionalities, the DPA plug-in is divided into several components as follows:

- similar to the structure of SPA plug-in, the first part (Figure 4.20) of the DPA view, called "ECC Algorithm", is defined in the "eccAlgorithmGroup". The vulnerability of "Elliptic Curve Cryptography" is introduced in this part. The method "doubling and addition always" is secure against the SPA attack, however, with statistic methods, it is still insecure to the DPA attack, since the tiny difference of the power traces between different operations can be enlarged and recognized with the DPA method.
- The second part is called "Explanation of Algorithm", in which the algorithm "doubling and addition always", and three countermeasures against the DPA attack, including "Randomizing the Scalar Multiplier", "Randomizing the Initial Point" and "Randomizing the Isomorphic Curve" are presented and explained in this part. Default content of part 2 is the explanation of the vulnerable algorithm "doubling and addition always" (Figure 4.21), in which the red background color marked lines are the weak point of ECC to DPA attack.

The user can change the content of part 2 by appointing the sort of countermeasure in the drop-down button (Figure 4.9). And the content of "Explanation of Algorithm" can be reset to the default algorithm "squaring and addition always" by clicking the button "Insecure to DPA: Double and Add Always" (Figure 4.10).

According to the selection in part 2, the content of part 2 will also automatically change to the description of the corresponding method.

The contents, which are displayed in Figure 4.21 of part 2 for the original method and its countermeasures are listed for example as:

- The content for the description of the algorithmic idea and principle of the original insecure algorithm "doubling and addition always":

Note that the vulnerability to SPA attack can be solved by adding a dummy "addition" operation in case the bit is "0". It is named as "doubling and addition always". However such countermeasure is still nonresistant to the DPA attack, since the power consumption of the "doubling and addition always" method for each input

Table 4.1.: The behavior description of DPA plug-in

| No. | type | initial condition | text | description |
|------|-------------------|-------------------|----------------|---|
| 1. | text field | enabled | see Figure 4.1 | description of ECC algorithm |
| 2. | text field | enabled | see Figure 4.1 | explanation of algorithm |
| 3. | drop-down button | enabled | empty | selection of prime field |
| 4. | drop-down button | disabled | empty | selection of parameter a |
| 5. | drop-down button | disabled | empty | selection of parameter b |
| 6. | output text field | disabled | empty | output of defined elliptic curve |
| 7. | output text field | disabled | empty | output of defined EC's order |
| 8. | drop-down button | disabled | empty | selection of an EC point on the curve |
| 9. | button | disabled | execute | activation of selected algorithm |
| 10. | output text field | disabled | empty | output of selected EC point's order |
| 11. | text field | enabled | see Figure 4.1 | explanation of parameter algorithm |
| 12. | table column | disabled | empty | round counter |
| 13. | table column | disabled | empty | result after doubling |
| 14. | table column | disabled | empty | result after addition |
| 15. | text field | enabled | see Figure 4.1 | explanation of DPA against "EC" |
| 16. | drop-down button | invisible | empty | selection of scalar multiplier |
| 17. | text field | invisible | empty | display the selected initial point |
| I. | group | enabled | see Figure 4.1 | red marked main group |
| II. | group | enabled | see Figure 4.1 | green marked ECC algorithm group |
| III. | group | enabled | see Figure 4.1 | yellow marked explanation of algorithm group |
| IV. | group | enabled | see Figure 4.1 | blue marked parameter of ECC group |
| V. | group | enabled | see Figure 4.1 | purple marked parameter of algorithm group |
| VI. | group | disabled | see Figure 4.1 | black marked table of intermediates group |
| VII. | group | enabled | see Figure 4.1 | pink marked explanation of DPA against EC group |

looks the same, but is slightly different. With the statistic methods like sampling analysis and differential analysis we can amplify the slight differences and recover the private key 'k' for $Q = kP$.

The "doubling and addition always" method in pseudo code:

```

1
2 Q[0] = P;
3 for(i = n-2 downto 0)
4 { Q[0] = 2Q[0];
5 Q[1]=Q[0] + P;
6 Q[0] = Q[di]; }

```

– The contents of the description of the algorithmic idea and principle of the countermeasure "randomizing the scalar multiplier":

- * Parameter k Randomization: if the scalar multiplier k is randomized in each turn of execution, the attacker is not able to recover the private key by comparing the differences between the power traces analysis of "doubling and addition always" operations.

- * The idea and principle are: first choose a random factor in group $[1, p-1]$, then a new randomized scalar multiplier $k' = k + r \cdot |P|$ will be calculated, since $|P| \cdot P$ is the infinity point (O) of the chosen elliptic curve, we get:

$G' = (k + r \cdot |P|) \cdot P = k \cdot P + r \cdot (|P| \cdot P) = kP + r \cdot O = G$ With such a method even for the same private key k, the power traces are always different. The relation between a certain input parameter k and its corresponding power trace during the operation process will be broken.

The contents for algorithmic explanation of the other two kinds of countermeasures "randomizing the initial point P" and "randomizing the isomorphic curve" are also displayed in the part 2 of the DPA plug-in (the group "Explanation of Algorithm") after switching to the corresponding countermeasure by selecting the dropdown button in Figure 4.9.

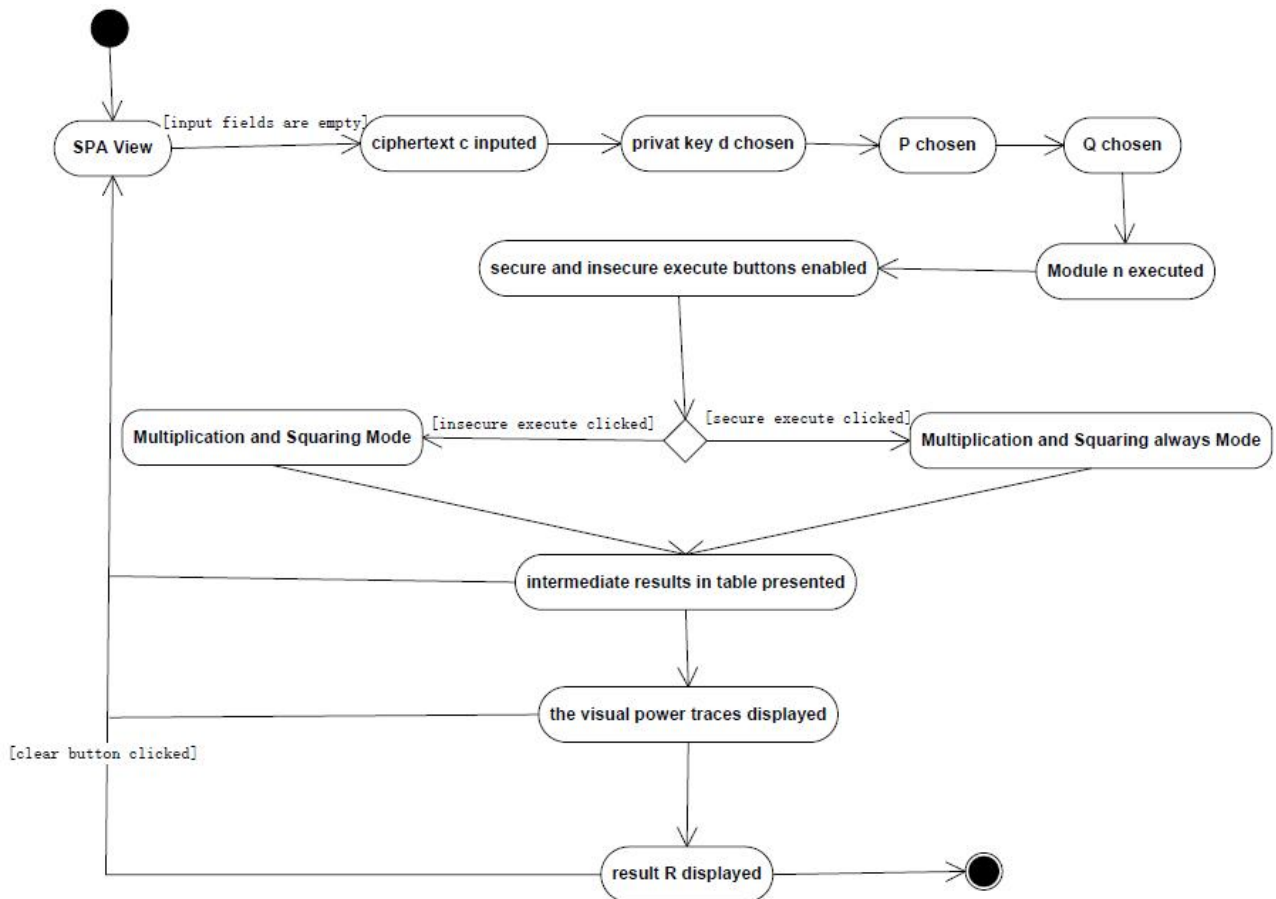


Figure 4.19.: The UML activity diagram of the DPA Plug-in

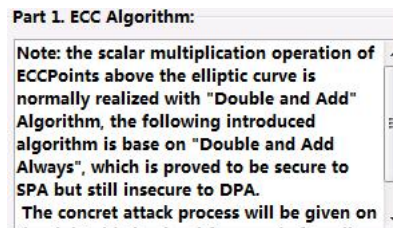


Figure 4.20.: The group "ECC Algorithm"

- The next part of the DPA plug-in, which is located in the upper middle of the main view, is called "Parameter of ECC" (Figure 4.20), the function of this group is already described in Section 4.4. In this group the user can define all parameters of an Elliptic Curve. After the prime field and parameter a and b are selected, the corresponding EC and its order will be displayed in the text field below. Once the drop-down button "Choose a Point (no infinity point O)" is enabled, the user can choose any point except the infinity point on the EC as the initial point P. After the selection of the initial point, its order is also displayed in the menu. Then the user can choose any number between 2 and $(|P|-1)$ as the scalar multiplier of the chosen EC. When the scalar multiplier is appointed, the button "execute" will be enabled. After clicking the "execute" button, three countermeasures or the "doubling and addition always" algorithm are processed and their initial parameters and temporary results of each step will be displayed in the right table and the text field below.
- The next middle below located group "DPA against Double and Add" and the right upper located table are used to create, display, and explain the original algorithm and its corresponding countermeasures. All of these components are used to realize the visual function of the plug-in. They create examples randomly for both vulnerable "squaring

```

Part 2. Explanation of Algorithm:
Always". However such countermeasure is
still nonresistant to DPA attack, since the
power consumption of "Double and Add
always" method for each input looks same,
but is slightly different. With statistic
method like sampling analysis and
differential analysis we can amplipy the
slight differences and recover the private
key 'k'.

Q = kP with Double and Add Always:
Q[0] = P;
for(i = n-2 downto 0){
  Q[0] = 2Q[0]; Q[1] = Q[0] + P;
  Q[0] = Q[d];
}

```

Figure 4.21.: The "doubling and addition always" algorithm

```

Part 2. Explanation of Algorithm:
Parameter k Randomization:
if the scalar multiplier k is randomized in
each turn of execution, the attacker could
not recover the private key per comparison
the differences between the power traces
analysis of "doubling and addition always"
operations.

The idea and principle are like so:
First Choose a random factor in group
[1,p-1], then a new randomized scalar
multiplier k'=k+r*order[P] will be
calculated,
since order[P]*P is the infinity point(O) of
the chosen elliptic curve, we get:
G'=(k+r*order[P])*P=k*P+r*(order[P]*P)
=kP+r*O=G

```

Figure 4.22.: By the selection of the "randomizing the scalar multiplier"

and addition always" algorithm and its countermeasures to explain how they actually work. More details about this group and the visual table can be found in Section 4.4.

- The last component is located in the lower right side of the main view, which is used to give a visual example (Figure 4.5) of part 4. The details of the example are explained in Section 4.3. With this concrete instance the user can understand more easily the described algorithm in part 4.

4.7 The Implementation of the DPA Plug-in

The basic execution of the "doubling and addition" algorithm is realized in the package "org.jcryptool.algorithm". There are altogether five classes defined in this package, including "ECCAdd.java", "ECCDouble.java", "ECCMultiply.java", "ECCOrderAndPoints.java", and "RandomFactorCreator.java". In this section some program segments are selected to explain.

- The addition operation of ECC is implemented as follows:

First the validity of both points on the elliptic curve will be checked with an if-statement. In the prime field p , the calculation of an integer A divided by integer B can be processed by multiplication of A and the inverse of B . So the inverse of $(x_2 - x_1)$ is executed and then the original division operation is replaced by $(y_2 - y_1)(x_2 - x_1)^{-1}$:

```

1
2 // this class is used to carry out ECAddition operation with the
3 // given parameter Point a, b and Prime field ecf public
4 ECPoint ecAddition(ECPoint a, ECPoint b, ECFieldFp ecf){
5     if(a.getAffineX().equals(null)||a.getAffineY().equals(null)||
6     b.getAffineX().equals(null)||b.getAffineY().equals(null)||
7     ecf.getP().compareTo(new BigInteger("3"))<=0) {
8         System.out.println("Invalid ECPoint!");

```

```

9         System.exit(-1);
10     }
11
12     // primeField, numerator and denominator are used to save the prime field ecf,
13     // numerator and denominator BigInteger primeField = ecf.getP();
14     BigInteger numerator =
15     (b.getAffineY().subtract(a.getAffineY())).mod(primeField);
16     BigInteger denominator =
17     (b.getAffineX().subtract(a.getAffineX())).mod(primeField);
18     if (denominator.equals(BigInteger.ZERO)){ return
19         ECPPoint.POINT_INFINITY; }
20
21     BigInteger inverseOfDenominator =
22     denominator.modInverse(primeField);
23     BigInteger tempValue =
24     (numerator.multiply(inverseOfDenominator)).mod(primeField);
25     BigInteger x3 =
26     (tempValue.pow(2).subtract(a.getAffineX()).subtract(b.getAffineX())).
27     mod(primeField);
28     BigInteger y3 =
29     ((tempValue.multiply(a.getAffineX().subtract(x3))).subtract(
30     a.getAffineY())).mod(primeField); ECPPoint addSumECPPoint
31     = new ECPPoint(x3,y3);
32     return addSumECPPoint; }

```

- There are altogether three kinds of countermeasures of DPA. The variable "counterFlag" is defined to identify which countermeasure is chosen. A "selectionListener" is added on the drop-down list "countermeasureselectionCombo", with this drop-down list the user can select which kind of countermeasure to be shown in the table. Besides the drop-down list for countermeasures there is also a button "insecureAlgoButton" available for shifting from the selected countermeasure to the original insecure "doubling and addition always". The implementation of this button is realized in a similar way as the drop-down list.
 - The default value of "counterFlag" is "1", that states the original vulnerable algorithm "doubling and addition always".
 - If its value is set to "2", that means the countermeasure of "randomizing scalar multiplier" is selected.
 - If the value is "3", that indicates the countermeasure of "randomizing initial point" is selected.
 - The last value of "counterFlag" is "4", that presents the countermeasure of "randomizing isomorphic curve".

With different values of "counterFlag" the processing of algorithms and temporary values which are listed in the table in part 4 are also distinct:

```

1     countermeasureselectionCombo.addSelectionListener(new SelectionAdapter() {
2         public void widgetSelected(final SelectionEvent e) {
3             parameterOfCountermeasuresGroup.setText(
4             PARAM_OF_COUNTERMEASURES_GROUP_TITEL);
5             counterFlag = countermeasureselectionCombo.getSelectionIndex()+2;
6             if(counterFlag == 2) {
7                 insecureText.setText("");
8                 insecureText.setText(RANDOMIZED_SCALAR_MULTIPLIER_TEXT);
9             }
10            else if(counterFlag == 3) {
11                insecureText.setText("");
12                insecureText.setText(RANDOMIZED_INITIAL_POINT_TEXT);
13            }
14            else if(counterFlag == 4) {
15                insecureText.setText("");
16                insecureText.setText(RANDOMIZED_ISOMORPHIC_CURVE_TEXT);
17            }

```

```

18     }
19   });
20   insecureAlgoButton.setForeground(SWTResourceManager.getColor(
21   255, 128, 128));
22   insecureAlgoButton.addMouseListener(new MouseAdapter() {
23     public void mouseUp(final MouseEvent e) {
24       parameterOfCountermeasuresText.setText(
25       TEXT_OF_PARAMETEROF_COUNTERMEASURESTEXT);
26       parameterOfCountermeasuresGroup.setText(
27       TITLE_OF_PARAMETEROF_COUNTERMEASURESGROUP);
28       counterFlag = 1;
29       unsecureText.setText("");
30       unsecureText.setText(UNSECURE_DOUBLE_ADD_ALWAYS_TEXT);
31       unsecureText.setLineBackground(4, 3,
32       new Color(parent.getDisplay(), cc));
33     }
34   });

```

- The following thread is used to click the execution button automatically so that the secure countermeasure can restart while it is possible that by randomizing the randomly chosen point on the elliptic curve may lead to invalid temporary result of execution of doubling or addition operation. If the temporary result is useless, this thread will be fired to choose a random point again. For example, in the doubling operation if the value of the y-axis of a randomly chosen point equals 0 or in the addition operation the value of the x-axis of both P_1 and P_2 is identity, the execution cannot be carried out since the denominator is 0 (the value of the whole equation is infinite). Thus it is necessary to create a thread which is used to examine the random selected points and avoid invalid values.

```

1     new Thread(){
2       Event event;
3       public void run(){
4         try {
5           Thread.sleep(30);
6         } catch (InterruptedException e) {}
7         event = new Event();
8         event.type = SWT.MouseMove;
9         event.x = pt.x-55;
10        event.y = pt.y-180;
11        executeButton.getDisplay().post(event);
12        try {
13          Thread.sleep(30);
14        } catch (InterruptedException e) {}
15        event.type = SWT.MouseDown;
16        event.button = 1;
17        executeButton.getDisplay().post(event);
18        try {
19          Thread.sleep(30);
20        } catch (InterruptedException e) {}
21        event.type = SWT.MouseUp;
22        executeButton.getDisplay().post(event);
23      }
24    }.start();

```

5 The Summary and Outlook

The motivation of this thesis is to visualize the SPA and DPA attack in the form of plug-ins for the JCrypTool platform. JCrypTool is an excellent e-learning platform with outstanding extensibility. All the teaching materials about cryptographic theories can be developed and realized in concrete visual form as a plug-in. With the help of such new form of e-learning, the users not only understand cryptography easier, but also extend the JCrypTool platform by developing their own cryptographic plug-ins. In some sense the JCrypTool platform is not only a good teaching platform but also a good practical learning platform for the software development.

5.1 Summary

Two plug-ins are implemented in this thesis: the SPA plug-in against RSA and the DPA plug-in against elliptic curve cryptographic system. And For both power analysis attacks the corresponding countermeasures are also discussed and explained. The plug-ins are realized in visual form and content help documents for the plug-ins is offered in English and German, with which the user can refer to the description of relevant ideas and principles about implementation. Furthermore, some examples are also included to support the understanding.

5.2 Outlook

The method "squaring and multiplication always" is introduced as the countermeasure of SPA attack against the RSA algorithm. Besides this method there are still two common kinds of countermeasures with which the SPA attacks can also be prevented. One is by using SPA resistant NAF_w method, the other is the Okeya and Takagi Scheme. Those methods are proposed in [QIN07] and [OKE03] respectively which can be aimed as the improvement of visualization of countermeasures against SPA attack.

For the visualization of DPA attack three kinds of countermeasures against DPA are stated in the visualization of DPA attack plug-ins. However, these methods are not robust enough to prevent the latest improved DPA attacks, such as Goubin's refined power analysis (RPA), and the zero-value point (ZVP) attack. For those improved PA attack the countermeasures should also be improved. A countermeasure named Smart's Isogeny Defense is secure against the RPA attack. However it is still vulnerable to the ZVP attack. So the visualization for those methods can be the topic of a further thesis on PA attack.

List of Figures

| | |
|---|----|
| 1.1. The relation between Eclipse and RCP | 2 |
| 1.2. The individual components of JCrypTool core project, taken from [GET09] | 3 |
| 1.3. The individual components of JCrypTool plug-in project, taken from [GET09] | 4 |
| 2.1. The side channels of a traditional cryptographic system | 7 |
| 2.2. The environment for PA attack | 8 |
| 2.3. The procedure of PA attack | 9 |
| 3.1. The idea of SPA attack | 10 |
| 3.2. The vulnerability of the "squaring and multiplication" algorithm | 12 |
| 3.3. The third part of the SPA plug-in | 13 |
| 3.4. The listed data for the "squaring and multiplication" | 14 |
| 3.5. An example of the visualization of the "squaring and multiplication" | 14 |
| 3.6. The listed data for the "squaring and multiplication always" | 15 |
| 3.7. An example of visualization of the "squaring and multiplication always" | 15 |
| 3.8. The structure of the SPA plug-in | 16 |
| 3.9. The user interface of the SPA plug-in | 16 |
| 3.10. The UML activity diagram of the SPA plug-in | 18 |
| 3.11. The first part of the SPA plug-in | 18 |
| 3.12. The second part of the SPA plug-in | 19 |
| 3.13. The fourth part of the SPA plug-in | 19 |
| 3.14. The specification of the "squaring and multiplication" | 20 |
| 3.15. The specification of the "squaring and multiplication always" | 20 |
| 4.1. The idea of the DPA attack | 24 |
| 4.2. All points on $EC : y^2 = x^3 + x + 1$ in $GF(5)$ | 26 |
| 4.3. A geometrical example for point addition | 27 |
| 4.4. A geometrical example for point doubling | 27 |
| 4.5. An example for the DPA attack against "doubling and addition always" | 28 |
| 4.6. An example for "ECC algorithm" | 30 |
| 4.7. The group "DPA against ECC" | 30 |
| 4.8. An example for "doubling and addition always" in the table | 31 |
| 4.9. The selection menu of the DPA countermeasures | 31 |
| 4.10. The selection menu of the "doubling and addition always" algorithm | 31 |
| 4.11. The description for the method "randomizing scalar multiplier" | 32 |
| 4.12. An example for "randomizing scalar multiplier" in the table | 32 |
| 4.13. The description for the method "randomizing initial point" | 32 |
| 4.14. An example for "randomizing initial point" in the table | 33 |
| 4.15. The description for the method "randomizing isomorphic curve" | 33 |
| 4.16. An example for "randomizing isomorphic curve" in the table | 34 |
| 4.17. The structure of the DPA plug-in | 34 |
| 4.18. The user interface of the DPA plug-in | 35 |
| 4.19. The UML activity diagram of the DPA Plug-in | 37 |
| 4.20. The group "ECC Algorithm" | 37 |
| 4.21. The "doubling and addition always" algorithm | 38 |
| 4.22. By the selection of the "randomizing the scalar multiplier" | 38 |

List of Tables

| | |
|--|----|
| 2.1. The devices for the PA attack | 8 |
| 3.1. The behavior description of SPA plug-in | 17 |
| 4.1. The behavior description of DPA plug-in | 36 |

A Bibliography

- [BUC04] Johannes Buchmann. Einfuehrung in die Kryptographie. Book, Spring Verlag GmbH, 2004.
- [CER00] Certicom Reasearch. SEC 1: Elliptic Curve Cryptography. paper, Certicom, Version 1.0, 2000.
- [CHA02] Hwasun Chuang. Differential Power Analysis Attack on Smart Card. Technical presentation, ICE615 Network Security, 2002.
- [CHA04] S. Chang, H. Eberle, V. Gupta, and N. Gura. Elliptic Curve Cryptography - How it Works. Technical tutorial, Sun Microsystems Laboratories, 2004.
available at: <http://research.sun.com/projects/crypto/HowECCWorks-USLetter.pdf>
- [CLA09] Eric Clayberg. Eclipse Plug-ins. Book, Addison-Wesley Verlag, 2009.
- [COR99] JeanSebastien Coron. Resistance against Differntial Power Analysis for Elliptic Curve Cryptosystems. Technical report, Gemplus Card International, Paris, 1999.
available at: <http://www.gemplus.com/smart/rd/publications/pdf/Cor99dpa.pdf>
- [DIS09] Wikipedia. Discrete Logarithm wikipedia, the free encyclopedia, 2010.[Oline; Januar 2010].
available at: http://en.wikipedia.org/wiki/Power_analysis
- [ELL09] Wikipedia. Elliptic Curve wikipedia, the free encyclopedia, 2009.[Oline; December 2009].
available at: http://en.wikipedia.org/wiki/Elliptic_curve
- [EXP08] Wikipedia. Exponentiation by Squaring wikipedia, the free encyclopedia, 2009.[Online; October 2008].
available at: http://en.wikipedia.org/wiki/Exponentiation_by_squaring
- [GET09] JCrypTool Group Getting started with JCrypTool. Technical tutorial, JCrypTool Group, 2009.
available at: http://jcryptool.sourceforge.net/JCrypTool/Documentation_files/getting_started_with_jcryptool.pdf
- [GOL01] S .Goldwasser and M .Bellare. Lecture Notes on Cryptography. Technical notes, MIT Laboratory of Computer Science, USA, 2001.
available at: <http://cseweb.ucsd.edu/~mihir/papers/gb.pdf>
- [HEL00] Kopka Helmut. LaTeX Band 1: Einfuehrung. Book, Addison-Wesley, 2000.
- [HEU04] Matthias Heuft. Side Channel Analysis of RSA-Implementations in Smartcards. Technical report, Berlin, 2004.
- [Joe04] Knappen Joerg. Schnell ans Ziel mit LaTeX. Book, Oldenbourg Wissenschaftsverlag, 2004.
- [JUR97] A .Jurisic and A .Menezes. Elliptic Curve and Cryptography. Technical report, CRC Press, Alabama, 1997.
available at: <http://www.scribd.com/doc/209755/EllipticCurvesAndCryptography>
- [KOC98] P Kocher, J. Jaffe and B. Jun. Introduction to Differential Power Analysis and Related Attacks. Technical report, Cryptography Research, San Francisco, 1998.
available at: <http://www.cryptography.com/resources/whitepapers/DPATechInfo.pdf>
- [LOV06] Tim Love. Advanced LaTeX. Technical document, 2006.
available at: http://wwwh.eng.cam.ac.uk/help/documentation/docsource/latex_advanced.pdf
- [OET00] Tobias Oetiker. LaTeX in 90 Minutes. Book, Version 3.16, 2000.
available at: <http://www.math.canterbury.ac.nz/php/resources/compdocs/lshort2e.pdf>
- [OKE03] K. Okeya and T. Takagi. The Width-w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks. Technial paper, Technische Universitaet Darmstadt, Darmstadt, 2003.
available at: <http://saluc.engr.uconn.edu/refs/sidechannel/okeya03widthw.pdf>
- [POW09] Wikipedia. Power Analysis. wikipedia, the free encyclopedia, 2009.[Oline; March 2009].
available at: http://en.wikipedia.org/wiki/Power_analysis

-
- [QIN07] B. Qin, M. Li, F. Kong and D. Li. Security Analysis of NAF_w and SPA Resistant Scalar Multiplication. paper, Shandong University, Qindao, 2007.
- [QUI02] J. Quisquater and M. RiZK. Side Channel Attacks. Technical report, State of the art, 2002.
available at: http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf
- [RIT07] Terry Ritter. Ritter's Crypto Glossary and Dictionary of Technial Cryptography. Glossary and dictionary, Ritter's Crypto Bookshop, 2007.
available at: <http://www.ciphersbyritter.com/GLOSSARY.HTM>
- [RSA09] Wikipedia. RSA. wikipedia, the free encyclopedia, 2010.[Online; January 2010]
available at: <http://en.wikipedia.org/wiki/RSA>
- [RUP07] C. Rupp and B. Stefan. Q.Stefan Queins: UML 2 Glasklar. Book, Carl Hanser Verlag, 2007.
- [SID09] Wikipedia. Side Channel Attack. wikipedia, the free encyclopedia, 2009.[Online; March 2009].
available at: http://en.wikipedia.org/wiki/Side_channel_attack
- [TAK04] Tsuyoshi Takagi. Some Side Channel Attacks on Elliptic Curve Cryptosystem. Course script, Technische Universitaet Darmstadt, 2004.
- [TAO08] Z. Tao and F. Yu. Side Channel Leakage Model and Security Analysis. Technial report, University of Electronic Science&Technology of China, Chengdu, 2008.
- [ZHO05] Y. Zhou and D. Feng. Side Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. Technical report, State Key Laboratory of Information Security, 2005.
available at: <http://csrc.nist.gov/groups/STM/cmvp/documents/fips1403/physec/papers/physecpaper19.pdf>