Faculteit
Ingenieurswetenschappen

Departement
Elektrotechniek – ESAT

KATHOLIEKE
UNIVERSITEIT
LEUVEN

Power Analysis Attacks on a Hardware Implementation of the Stream Cipher MICKEY

Submitted for partial fulfillment of requirements for the Master of Science: Electrical Engineering (ICT)

Hongxu Zhao
Shiqi Li

Promotor:
Prof. dr. Ingrid Verbauwhede
Daily supervision:
Ir. Elke De Mulder
Ir. Benedikt Gierlichs

2008—2009

© Copyright K.U.Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor(en) als de auteur(s) is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend U tot de K.U.Leuven, Departement Elektrotechniek – ESAT, Kasteelpark Arenberg 10, B-3001 Heverlee (België). Telefoon +32-16-32 11 30 & Fax. +32-16-32 19 86 of via email: `info@esat.kuleuven.be`.

Voorafgaande schriftelijke toestemming van de promotor(en) is eveneens vereist voor het aanwenden van de in dit afstudeerwerk beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

© Copyright by K.U.Leuven

Without written permission of the promotors and the authors it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilize parts of this publication should be addressed to K.U.Leuven, Departement Elektrotechniek – ESAT, Kasteelpark Arenberg 10, B-3001 Heverlee (Belgium). Tel. +32-16-32 11 30 & Fax. +32-16-32 19 86 or by email: `info@esat.kuleuven.be`.

A written permission of the promotor is also required to use the methods, products, schematics and programs described in this work for industrial or commercial use, and for submitting this publication in scientific contests.

Acknowledgement

First and foremost, we want to thank our promotor Prof. Ingrid Verbauwhede for giving us the opportunity to develop this thesis in the COSIC research group at K.U.Leuven. Without her help this thesis would not have been possible. We would like to thank our daily supervisors Benedikt Gierlichs and Elke De Mulder for their support and guidance during the development of the thesis, but mostly for being kind and helpful and for everything we have learned from them.

We also want to thank our friends in Leuven for the good times we shared together.

Last but not least, to our parents for their encouragement and support.

Abstract

Nowadays, various side-channel analysis techniques start to play an important role in cryptography and bring a serious threat on implementations of cryptographic algorithms. Information extracted from the power consumption or the electromagnetic leaks of these implementations are used to perform the attack effectively. So far, side-channel attacks on block ciphers and public key algorithms have been discussed extensively. However, the literature about side-channel attacks on stream ciphers is not as much as the other cryptographic algorithms. The few existing references mainly treat timing and template attacks, or provide a theoretical analysis of weaknesses of stream cipher constructions and some design suggestions on general countermeasures. There has been little published on the side-channel attacks such as differential power analysis, differential EM-analysis and fault injection techniques specific to stream ciphers. To assist this effort, prototype quantities of an ASIC, which is called eSCARGOt, containing all the phase-III hardware candidates has been designed and fabricated on 0.18um CMOS, for example, MICKEY, Trivium and Grain.

In this thesis, we first implement a communication interface between the ASIC and a PC which is used to interact with the ASIC and facilitates to collect the power traces for further analysis. Afterwards, side-channel attack has been used to reveal the complete secret key. The most effort has been put into applying several Differential Power Analysis techniques on the implementation of the MICKEY-128 algorithm. Additionally, the comparison among these different methods is discussed.

Keywords: Side-channel analysis, Differential Power Analysis, eSCARGOt, MICKEY-128

Contents

Acknowledgement **ii**
Abstract **iii**
Contents **iv**
List of symbols **vi**
List of figures **viii**
1 Introduction and overview **1**
1.1 Motivation . . . 1
1.2 Goals . . . 2
1.3 Structure of the thesis . . . 3
2 Design and implementation of the interface **4**
2.1 Introduction of the eSCARGOt ASIC and communication protocol . . . 4
2.2 Design of the interface on FPGA . . . 7
2.2.1 Software design on the PowerPC . . . 9
2.2.2 Hardware design for the Mickey peripheral . . . 10
2.2.3 Test result from the Logic Analyzer . . . 11
2.3 Summary . . . 14
3 Introduction of Stream Ciphers **15**
3.1 Stream Ciphers in a nutshell . . . 15
3.2 Introduction of MICKEY-128 . . . 16
3.2.1 Introduction . . . 16
3.2.2 Input and output parameters . . . 17
3.2.3 Acceptable use . . . 17
3.2.4 Components of the keystream generator . . . 17
3.2.4.1 The registers . . . 17
3.2.4.2 Clocking the register R . . . 17
3.2.4.3 Clocking the register S . . . 18
3.2.4.4 Clocking the overall generator . . . 20
3.2.5 Key loading and initialisation . . . 21
3.2.6 Generating keystream . . . 21
3.2.7 The intended strength of the algorithm . . . 21
3.2.8 Performance of the algorithm . . . 22
4 Theoretical background of Side-Channel Attacks **23**
4.1 Timing Attack . . . 24
4.2 Power Consumption Attacks . . . 24
4.2.1 Power consumption . . . 25
4.2.2 Simple Power Analysis . . . 26

4.2.3 Differential Power Analysis 26
4.2.3.1 General description 26
4.2.3.2 Statistical Analysis: Attacks based on the Correlation Coefficient 27
4.2.3.3 Statistical Analysis: Attacks based on the Difference of Means 28
4.2.3.4 Statistical Analysis: Attacks based on the Distance of Means . 28
4.2.3.5 Power Model: Attacks based on the Bit Model 29
4.2.3.6 Power Model: Attacks based on the Hamming-Weight Model . 29
4.2.3.7 Power Model: Attacks based on the Hamming-Distance Model 29
4.2.3.8 Power Model: Attacks based on the Zero-Value Model 29
4.3 Electromagnetic Analysis Attack 30
4.4 Preventing Side-Channel Attacks 31
4.5 Summary 32
5 Practical attack on the hardware implementation of the MICKEY Stream Cipher 33
5.1 First Step: Measuring the Power Consumption 33
5.2 Second step: Building the hypothetical power consumption matrix by different power models 37
5.3 Third step: Attacks based on different statistical analyses 39
5.4 Practical attacks 40
5.4.1 Comparison: power traces measured from V_{DD} vs. GND 40
5.4.2 Comparison: attacks based on S register alone vs. S and R registers . . 41
5.4.3 Correlation coefficient analysis with Hamming-Distance model 43
5.4.4 Correlation coefficient analysis with Hamming-Weight model 47
5.4.5 Correlation coefficient analysis with Zero-Value model 48
5.4.6 Difference of Means analysis with Hamming-Distance model 48
5.4.7 Difference of Means analysis with Hamming-Weight model 51
5.4.8 Distance of Means analysis with Hamming-Distance model 52
5.4.9 Distance of Means analysis with Hamming-Weight model 53
5.5 Summary 54
6 Conclusions and future work 56
6.1 Conclusions 56
6.2 Future work 56
Bibliography 58
A Algorithms for different DPA attack techniques 61
B The GEZEL implementation of the interface 63

List of symbols

We provide a list of the most important variables and functions used in this thesis. They are arranged in the order of appearance.

eSCARGOt	European Stream Ciphers Are Ready to Go
eSTREAM	A project brought by ECRYPT
V_{IO}	Voltage for the I/O of the eSCARGOt ASIC
V_{CORE}	Voltage for the internal core of the eSCARGOt ASIC
K	Secret key in MICKEY
IV	Initialization vector in MICKEY
R	Linear register in MICKEY
S	Non-linear register in MICKEY
z	Keystream bit in MICKEY
N	RSA modulus
d	Private key in RSA
$i_{DD}(\mathrm{t})$	Instantaneous current
$p_{cir}(\mathrm{t})$	Instantaneous power consumption
$P_{stat}(\mathrm{t})$	Static power consumption
$P_{dyn}(\mathrm{t})$	Dynamic power consumption
I_{leak}	Leakage current
C_L	Output capacitance
P_{chrg}	Average charging power
p_{chrg}	Instantaneous charging power
$p_{sc}(\mathrm{t})$	Instantaneous short-circuit power
$f(d, k)$	Function influenced by both plaintext and key
$\mathbf{T}$	Matrix of power consumption values
D	Number of power traces used for a differential analysis attack
T	Length of a power trace
$\mathbf{V}$	Matrix of hypothetical intermediate values in a DPA attack
$\mathbf{H}$	Matrix of hypothetical power consumption values in a DPA attack
$\mathbf{R}$	Matrix of results of a DPA attack

$r_{i,j}$	Estimator for correlation coefficient
$\mathbf{m}$	Mean vector
$\mathbf{V}_{pre}$	Matrix contains hypothetical power consumption values during previous run of the encryption
$\mathbf{V}_{current}$	Matrix contains hypothetical power consumption values during current run of the encryption
HD	Hamming-Distance model
HW	Hamming-Weight model
ZV	Zero-Value model

List of figures

1.1 Block diagram of the measurement setup. 2

2.1 Package view of the eSCARGOt ASIC. [1] 4
2.2 Pin assignment of the ASIC. [1] 6
2.3 The block diagram of the 3 components involved in an encryption process. 7
2.4 XUP Virtex-II Pro Development System Board. 8
2.5 Block diagram of the interface. 9
2.6 Finite state machine for the interface. 11
2.7 Result from the Logic Analyzer (Mickey). 12
2.8 Result from the Logic Analyzer (Trivium). 13

3.1 The general stream cipher process. 15
3.2 Linear feedback shift register. 16
3.3 Clocking the R register with CONTROL_BIT_R = 0. 18
3.4 Clocking the R register with CONTROL_BIT_R = 1. 18
3.5 Table of sequences needed by clocking S. 19
3.6 Clocking the S register. 20
3.7 The MICKEY-128 Stream Cipher Architecture. 20
3.8 Synthesis result of area consumptions for eSTREAM candidate algorithms, compared to an efficient AES implementation. [2] 22

4.1 Power analysis methodology. 25
4.2 A CMOS inverter. 25
4.3 A loop antenna is placed parallel with the FPGA under measurement. [3] 31

5.1 Test board with a scope probe and a current probe connected. 34
5.2 Measurement setup. 35
5.3 A power consumption trace measured at V_{DD}. 36
5.4 A power consumption trace measured at *GND*. 36
5.5 Matrix **T** contains the measured power traces. 37
5.6 Matrix **V** contains the intermediate values. 38
5.7 Matrix **T** contains the hypothetical power consumptions. 38
5.8 The procedure of analysis. 39
5.9 Attack using the power traces measured at *GND*. 40
5.10 Attack using the power traces measured at V_{DD}. 41
5.11 Attack based on S register only. Key hypotheses at the 21st key-bit. 42
5.12 Attack based on R and S registers. Key hypotheses at the 21st key-bit. 43

5.13 Key hypotheses at the 128th key-bit. 44
5.14 Correlation as a funtion of the number of power traces. 45
5.15 Correlation as a funtion of the number of power traces (only looking at the 101st column in **R**). 46
5.16 Key hypotheses at the 1st key-bit (only looking at the 101st column in **R**). 46
5.17 Key hypotheses at the 3rd key-bit. 47
5.18 Correlation as a funtion of the number of power traces. 48
5.19 Key hypotheses at the 1st key-bit. 49
5.20 Difference of means as a funtion of the number of power traces. 50
5.21 Around the attack target point (zoomed version of figure 5.19). 50
5.22 Plots for the correct key hypotheses in the attack on the last 16 key bits. The x-axes show the samples and the y-axes show the difference of means. 51
5.23 Key hypotheses at the 2nd key-bit. 52
5.24 Key hypotheses at the 1st key-bit. 53
5.25 Distance of means as a funtion of the number of power traces. 53
5.26 Key hypotheses at the 4th key-bit. 54

List of algorithms

1 Pseudocode for DPA with the Hamming-Distance power model and the Correlation Coefficient analysis. 61
2 Pseudocode for DPA with the Hamming-Weight power model and the Correlation Coefficient analysis. 61
3 Pseudocode for DPA with the Hamming-Distance power model and the Difference of Means analysis. 62
4 Pseudocode for DPA with the Hamming-Distance power model and the Distance of Means analysis. 62

Chapter 1

Introduction and overview

1.1 Motivation

Cryptography has a long history originated in Egypt about 4000 years ago and played a very important role in the twentieth century especially during both world wars. Generally speaking, it is concerned with protecting information. Classic cryptography was solely considered as message encryption. One notable example in ancient time is the Caesar cipher, which is a type of substitution cipher where each letter or groups of letters in the plaintext were replaced according to some rules. Cryptography evolved extensively during the last decades, with the aid of development of digital computer and electronics. It finds its various applications in the technically advanced society, from military, electronic commerce to computer security and wireless communication. The modern study of cryptography can be divided into several areas, including symmetric-key cryptography, public-key cryptography, etc. The symmetric-key ciphers mainly consist of ciphers encrypting/decrypting blocks of fixed length symbols (block ciphers [4]) and ciphers working on a continuous stream of individual characters (stream ciphers [4]).

From the traditional cryptanalysis' point of view, many ciphers are secure, and the only way to reveal the secret key is to try all possible combinations by brute force. Thus, an exhaustive search becomes unfeasible if the number of combinations is large enough. For example, the RSA [5] public-key algorithm is said to be secure currently with at least 2048 bits modulus [6]. In reality, an algorithm will always have to be implemented on a cryptographic device, which will somehow leak additional information that can be used for the attacker to mount attacks on the implementation of the algorithm. These approaches, which make use of the leaked information including power consumption, electromagnetic radiation and time delay are knowns as Side-Channel Attacks (SCAs). SCA typically includes Simple Power Analysis (SPA), Differential Power Analysis (DPA) [7], Simple and Differential EM-Analysis (SEMA & DEMA) [8] and [9] and Timing Attacks [10]. Since first proposed in 1996, they have been used in a widespread way to extract the keys of both symmetric and public key algorithms running on different platforms such as ASICs [11], FPGAs [12], DSPs [13] and CPUs [14]. Concerning these attacks, most of the researches are oriented to attacks on public key algorithms and block ciphers such as AES [15]. To date SCAs on stream ciphers did not gain so much research effort.

eSTREAM [16] is a project that aims to identify "new stream ciphers that might become suitable for widespread adoption", organized by ECRYPT (European Network of Excellence for Cryptology) [17]. The call for submissions of stream ciphers fall into two catalogs of profiles: One for software applications with high throughput requirements and one for hardware applications with restricted resources. The project was completed in May 2008. The announced eSTREAM portfolio includes 4 Profile-1 (Software) ciphers and 3 Profile-2 (Hardware) ciphers. Until now, there has not been much research published on the various side-channel attacks specific to these stream cipherse except the one [18] on TRIVIUM [19] and Grain [20]. To assist this effort, these stream ciphers have been designed and fabricated on a prototype ASIC called eSCARGOt [21] using 0.18 um CMOS technology for further hardware performance assessment and side-channel analysis.

1.2 Goals

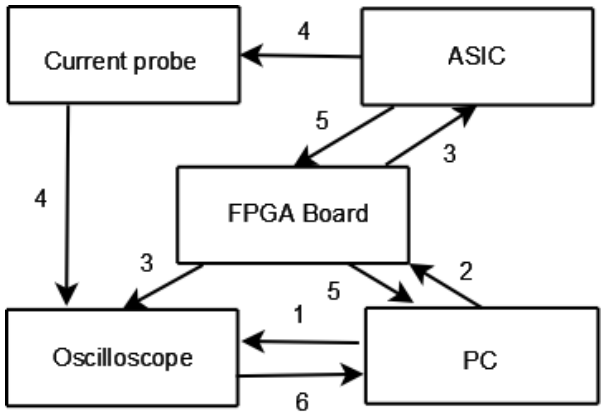

Figure 1.1: Block diagram of the measurement setup.

In our thesis we perform physical attacks on an implementation of the MICKEY algorithm, which is one of the stream ciphers integrated in the cryptographic ASIC. Differential Power Analysis (DPA) is the main approach we use and by using it we attack the hardware implementation of the algorithm instead of the algorithm itself. In other words, we focus on the information leaked from the implementation, which is the power consumed by each execution of MICKEY. The power traces are collected by an oscilloscope with the help of a current probe as shown in figure 1.1. Normally, a large amount of power traces on different data blocks (initial vectors in our case) are required to reveal the secret key effectively. A PC is used to generate different initial vectors, key[1] and plaintext for the ASIC. In order to make the communication between the PC and the ASIC possible, an FPGA is used to build an interface between them. This interface ensures the transfer of the initial vectors, key and plaintext to the ASIC to start the encryption with MICKEY. A trigger is sent to the oscilloscope at a specific moment to gather the corresponding power traces. The interface is also used to send back the ciphertext from the ASIC to the PC after each successful encryption.

[1]Normally, in a hardware implementation of stream cipher the key is kept secret inside. In the case of the eSCARGOt implementation, since the intention is to provide the development and performance assessment for side-channel analysis on these stream ciphers, the key is set as an external input as well.

At the end we collect thousands of power traces. We perform DPA attacks based on different power models or different statistical analyses in order to find the most efficient approach.

1.3 Structure of the thesis

Generally, our thesis work can be divided into two parts: interface design on an FPGA and physical attack on an implementation of the MICKEY algorithm. The structure of this book is:

Chapter 2 In this chapter we introduce the specification of the cryptographic ASIC and show the FPGA design of the interface in details.

Chapter 3 and chapter 4 These two chapters tell the story about the theoretical background of the stream cipher and physical attack.

Chapter 5 In this chapter we show the practical work of the DPA attacks on MICKEY. A lot of measurement and analysis results are given to make the comparison among them.

Chapter 6 At the end we conclude our thesis work and give some suggestions on the future work.

Chapter 2

Design and implementation of the interface

As we introduced in the first chapter, our target is to perform side-channel attacks on the eSCARGOt ASIC. However, we need to build an interface which enables us to communicate with the attacked device beforehand. Therefore, in this chapter, we introduce the specific ASIC first and then continue with the detailed process that shows how such an interface is designed on an FPGA.

2.1 Introduction of the eSCARGOt ASIC and communication protocol

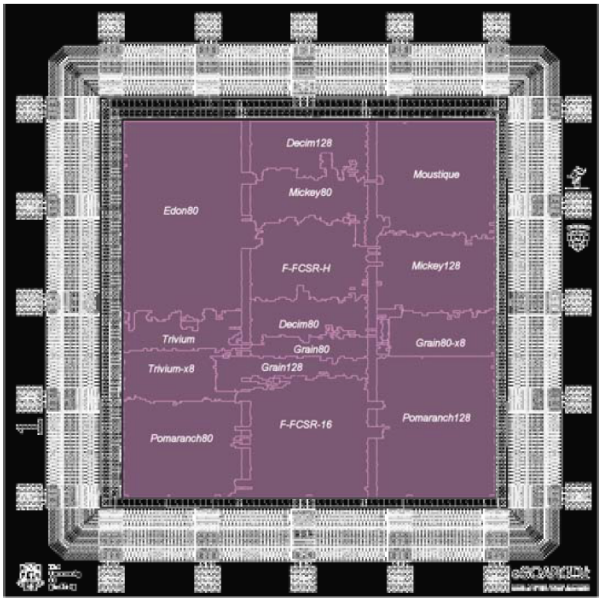

Figure 2.1: Package view of the eSCARGOt ASIC. [1]

All the candidates from the eSTREAM phase-III hardware profile have been designed and fabricated on a prototype ASIC called eSCARGOt [1] using 0.18 um CMOS technology. It uses a common synchronous serial interface with handshaking and includes a total of 15 designs listed in table 2.1. All of the designs are integrated in the ASIC as shown in figure 2.1. The final submissions for eSTREAM are listed in table 2.2, such as the chosen stream ciphers Trivium [19], MICKEY and Grain [20] from the hardware profile.

Table 2.1: All the phase-III hardware candidates in eSTREAM which are implemented on the eSCARGOt ASIC.

1	Moustique
2	Edon80
3	Trivium
4	Decim80
5	Decim128
6	F-fcsr-h
7	F-fcsr-16
8	Grain80
9	Grain128
10	Mickey80
11	Mickey128
12	Pomaranch80
13	Pomaranch128
14	Grain80 (x8 internally)
15	Trivium (x8 internally)

Table 2.2: The final eSTREAM portfolio.

Profile 1 (SW)	Profile 2 (HW)
HC-128	Grain v1
Rabbit	MICKEY v2
Salsa20/12	Trivium
SOSEMANUK	

The nominal voltage of the I/O is compatible with 3.3V and the internal core of the chip operates around 1.8V. The maximum acceptable frequency is 50 MHz. The overall chip size is $1521 \times 1521\,\mu$m with totally 20 pins attached as shown in figure 2.2. The functionality of each pin is explained in table 2.3.

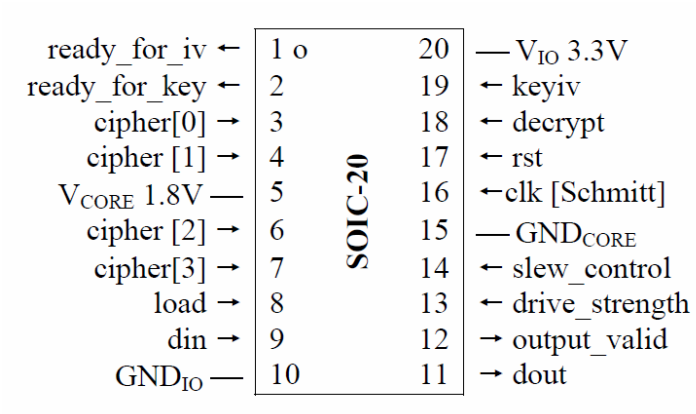

Figure 2.2: Pin assignment of the ASIC. [1]

Table 2.3: The description for the pins functionality.

clk	operating clock, maximum 50 MHz.
rst	reset the current selected cipher.
keyiv	synchronous serial key/IV input.
din	synchronous serial plain/cipher text input.
dout	synchronous serial cipher/plain text output.
load	handshake input to load I/O.
ready_for_key	handshake output to acknowledge key.
ready_for_iv	handshake output to acknowledge IV.
output_valid	handshake output to acknowledge plain/cipher text input.
decrypt	selects decrypt (Moustique only) also selects alternate design variation for Decim and Pomaranch.
cipher [0...3]	select "current cipher", all the other ciphers remain unaffected.
slew_control	output driver cells (1 = fast slew), normally low.
drive_strength	output driver cells (1 = max strength), normally low.
$\mathbf{V_{IO}}$, $\mathbf{V_{CORE}}$	3.3 V nominal for IO, 1.8 V nominal for core.
$\mathbf{GND_{IO}}$, $\mathbf{GND_{CORE}}$	Ground pins for core and I/O respectively.

The communication protocol designed for this chip is described in the following steps:

1. Power up the chip and internal core by 3.3 V and 1.8 V respectively.
2. Supply a monotonic 3.3 V LVTTL (Low Voltage Transistor-Transistor Logic) standard clock signal generated from the FPGA to the "clk" pin.
3. Choose the target cipher by 4-bit cipher select pins.
4. Reset the whole chip by setting the reset signal high for at least one clock cycle.
5. Wait for the "Ready_for_Key" or "Ready_for_IV" signals to be high and react to these signals with key and initial vector respectively. Meanwhile, "load" signal has to be driven high to indicate, that the output bit of key or initial vector is valid.

6. After the key and initial vector are received by the ASIC correctly, the signals of "Ready_for_Key" and "Ready_for_IV" go low which indicates, that the selected cipher is running.
7. Wait until the "output_valid" signal is driven high, send out the plaintext bit by bit with setting the "load" signal high. After one clock cycle the ciphertext for one bit plaintext is available on the "dout" pin. The ciphertext equals the result of keystream XOR plaintext.

This protocol should be strictly followed in the design of the interface on an FPGA, which is introduced in the next section.

2.2 Design of the interface on FPGA

The design on FPGA works as an interface between the PC and the ASIC providing us the opportunity to perform the power analysis attack on the eSCARGOt ASIC. The communication between the PC and the FPGA is done by an RS-232 interface. In order to perform an encryption, the PC first sends the cipher select command followed by the plaintext to the FPGA. The FPGA forwards it to the ASIC to run the specific cipher chosen by the cipher select commmand. Subsequently, the ASIC performs an encryption and returns the corresponding ciphertext to the FPGA. At last, the FPGA sends the ciphertext back to the PC. In figure 2.3 here is a general overview of the whole process.

Figure 2.3: The block diagram of the 3 components involved in an encryption process.

The communication interface is implemented on the XUP Virtex-II Pro Development System [22] shown in figure 2.4 with a high performance Virtex-II Pro XC2VP30 FPGA and a lot of peripheral components, including RS-232 serial port, expansion connector, USB and compact flash configuration ports, which are the main blocks we are using in the design.

Serial Port The XUP Virtex-II Pro Development System provides a single RS-232 serial port configured with hardware handshake using a standard DB-9 serial connector. It is used to transfer data between the PC and the interface.

Expansion Connectors A total of 80 low-speed pins are provided to accept both Digilent peripheral devices and ribbon cables for user-defined use. We use some of them to transfer data between the interface and the ASIC using a standard Parallel ATA cable.

Virtex-II Pro FPGA with Embedded IBM PowerPC 405 processor cores.

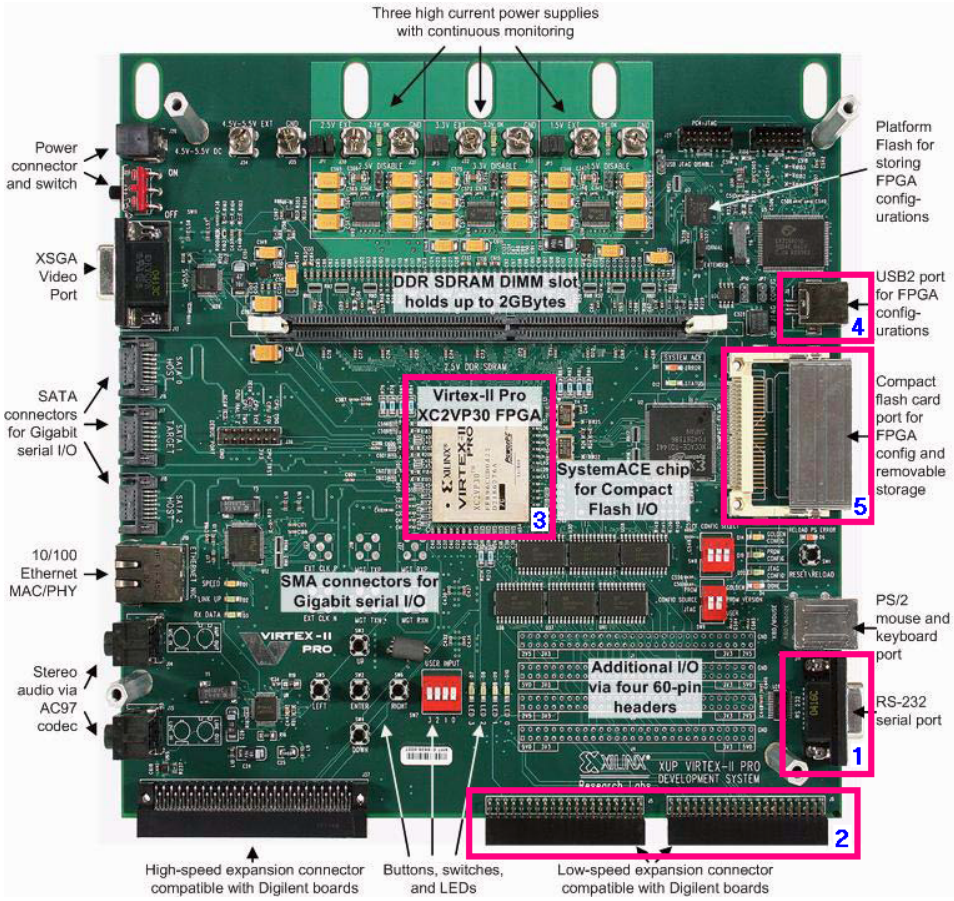

Figure 2.4: XUP Virtex-II Pro Development System Board.

USB port is used for programming or configuring the Virtex-II Pro FPGA in Boundary-Scan mode.

Compact Flash port is provided by the System ACE Compact Flash Controller to either configure the FPGA or be used as a general-purpose non-volatile storage. In case that the USB configuration may fail, the compact flash port can be used to download the program to configure the FPGA.

As we indicated in the beginning of this chapter, the target design is an interface between a PC and a cryptographic ASIC. For connection with the ASIC we simply use the low-speed expansion ports on the FPGA while for connection with the PC we choose the RS-232 serial ports. Fortunately, the RS-232 serial ports can work as a peripheral of PowerPC, which is a hard-core microprocessor embedded in the FPGA. In other words all the handshake and communication protocol for RS-232 are handled by the PowerPC automatically. Therefore, for the ease of the design, the PowerPC is used as a higher level software component to receive the command from the PC and meanwhile control the hardware design to generate the response.

The Embedded Development Kit (EDK) is a suite of tools that enables you to design a complete embedded processor system for implementation in a Xilinx FPGA device. We use it to design

our embedded system which involves both software and hardware components. The block diagram of the whole interface system is shown in figure 2.5.

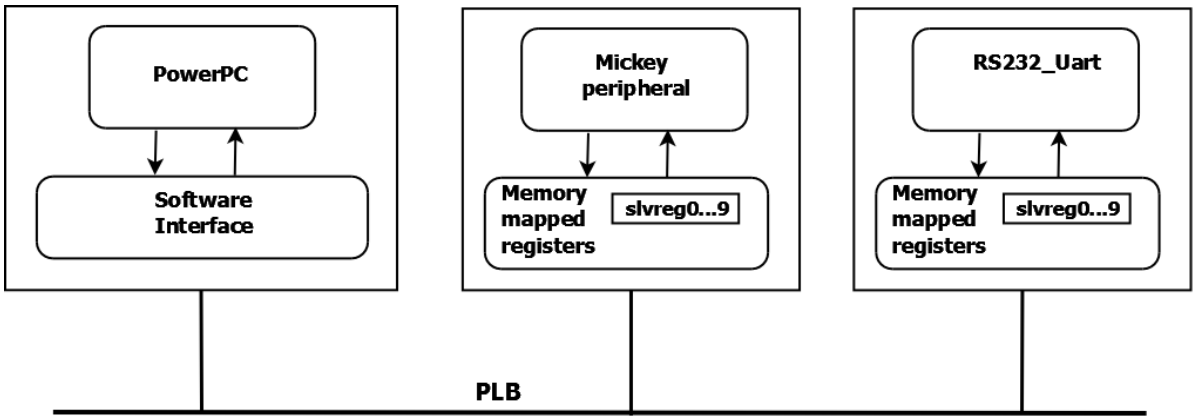

Figure 2.5: Block diagram of the interface.

This block diagram shows the architecture and the used components of the system. Our design includes the PowerPC and two peripherals which are the RS-232 and the Mickey peripheral. We use C language to program the PowerPC and VHDL for the Mickey peripheral. Between them we use a Processor Local Bus [23] (PLB) for exchanging data. The processor local bus is an on-chip bus specifically designed for on-chip processor memory interconnect. It has several features designed for high performance:

- There are separated address, read data and write data buses for the instruction and data transfer.
- The PLB supports pipelined transfers, allowing a new transfer to be initiated before the old one has completed.
- The PLB is a fully-synchronous bus with a single clock source shared by all masters and slaves attached to it.

The interface can be regarded as a hardware/software co-design embedded system, which is treated in two parts respectively.

2.2.1 Software design on the PowerPC

The functions of the PowerPC in our system are:

1. receives data from the PC via the RS-232 interface and further transfers it to the hardware component. The received data includes the cipher select command, initial vector, secret key and plaintext.

2. afterwards, the ciphertext is transferred back to the hardware component from the ASIC, the PowerPC forwards it to the PC.

Inside the transferring of the data on both sides of the interface, we have to do a simple realignment. It is required due to the reason that the RS-232 works on 8 bits while the

PowerPC is a 32 bits microprocessor, and the memory mapped registers, namely the slave registers "slvreg" provided to the hardware component is also 32 bits. The detailed process works as follows: for example, in the case of data transferring from the PC to the ASIC, we store the first received byte into a 32 bits register at bit position [7...0]. The next received byte is also stored into this register but at bit position [15...8] and the two subsequent bytes are shifted into the register as well which means one 32 bits register can hold up 4 received bytes. For the upcoming bytes, new registers are needed and thus we create a register array with each element containing one 32 bits register. Until all the input data are collected into the register array, they are forwarded to the hardware component via the PLB bus. The working principle for transferring data from the hardware component to the PC is roughly the same except that the received data from the Mickey peripheral is first decomposed into bytes and then sent back to the PC.

2.2.2 Hardware design for the Mickey peripheral

When we design the hardware we do not use VHDL directly, instead, we use GEZEL which is a cycle-based hardware description language based on the Finite-State-Machine + Datapath (FSMD) model. The GEZEL tools offer stand-alone simulation, co-simulation, and code-generation into synthesizable code. At the end it is translated to VHDL code for synthesis on FPGA. For the further detailed information of GEZEL please refer to [24].

Actually, we have implemented the interfaces for three stream ciphers, which are Trivium, MICKEY 128 and Grain 80. The entire Finite State Machine of the design is shown in figure 2.6.

The system starts with staying in the initial state as long as the "Flag" signal is zero. This "Flag" signal is a kind of control signal coming from the software component, at some moment this signal is set high which means all the received input data from the PC has been stored in the registers which are visible for the hardware component. In the meanwhile the hardware component may start communicating with the ASIC. Afterwards, the hardware component first read and store the input data into cipher select registers, initial vector registers, secret key registers and plaintext registers separately, and then output the "Cipher Select" signal to the ASIC. Next, as required by the specification the "Reset" signal has to be set high for at least one complete clock cycle, and we keep the "Reset" high for five clocks for safety. After this step, the hardware component stays in the update state which updates all the input signal every clock cycle. From now on the hardware component starts to listen to the ASIC and works passively. If the ASIC requires the secret key, it sets the "Ready for Key" signal high, the hardware component reacts to this signal by sending out the key until the entire key is correctly received by the ASIC. For the case of the initial vector the ASIC sets the "Read for IV" signal high until it receives the whole initial vector. Different implementations of designs require the key and IV in two different sequences, but our design can handle both the cases well because our system is in passive mode and it only follows the orders from the ASIC after the reset state. Once the key and IV are sent, the cipher inside the ASIC starts running. The hardware component waits again in the update state. Whenever the "Output Valid" signal is high, it means the keystream is valid and the hardware component is allowed to output the plaintext to the ASIC. One clock cycle later the ciphertext is generated and sent back to the hardware component.

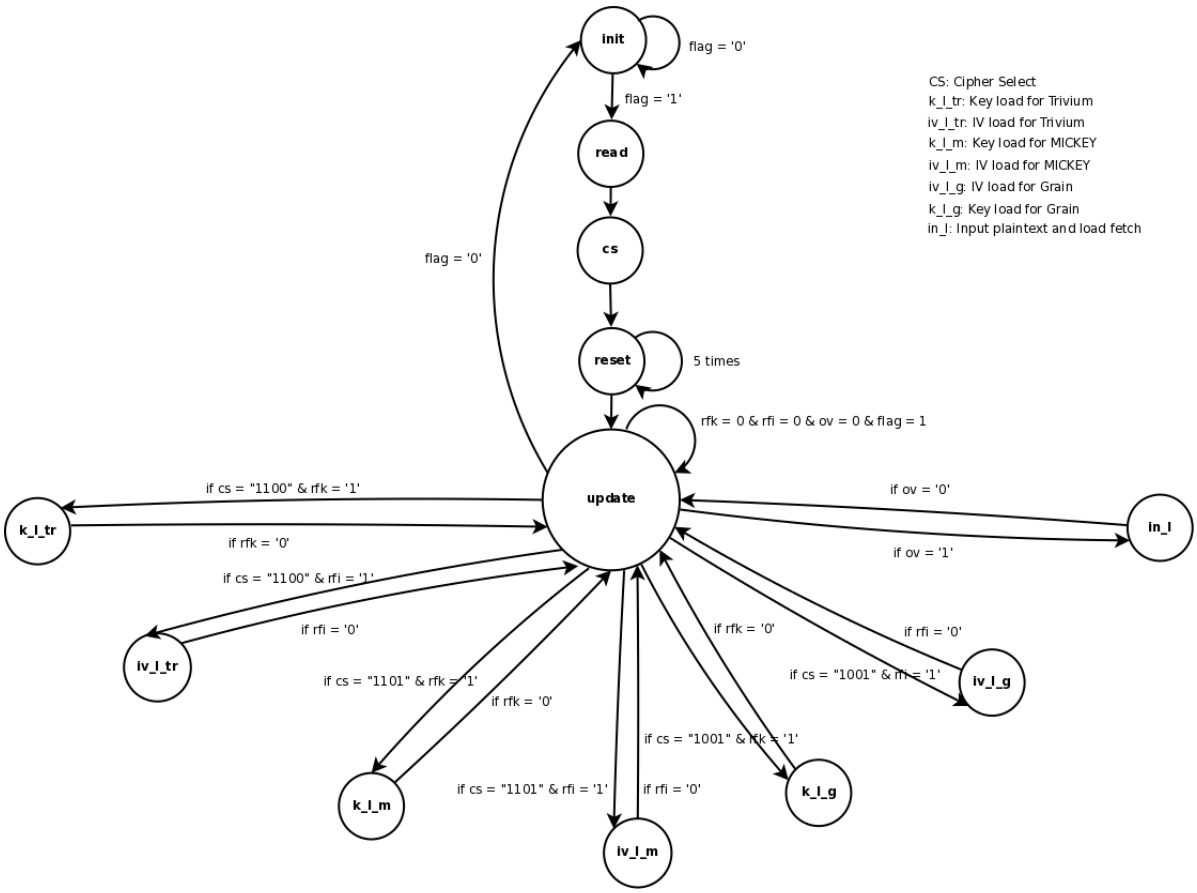

Figure 2.6: Finite state machine for the interface.

2.2.3 Test result from the Logic Analyzer

Before we connect the interface with the ASIC and the PC, we need to test the interface first. It is done by using a logic analyzer to emulate the behavior of the ASIC and a PC to provide the test data to the interface. The output from the ASIC is transferred to the interface via a set of data probes of the logic analyzer while several test probes are also connected to inspect the correct handshaking and data transfer as expected in the specification of the ASIC. After this is verified, we connect the ASIC with the interface and use the logic analyzer to trace all the input and output ports of the ASIC. By doing this we are also able to check whether the communication sequence is correct and simply find the point at where the ASIC stops when it goes wrong. Here we choose the MICKEY encryption and the Trivium encryption as two examples to show the detailed communication procedure between the FPGA and the ASIC. The timing diagrams are given in figure 2.7 and 2.8.

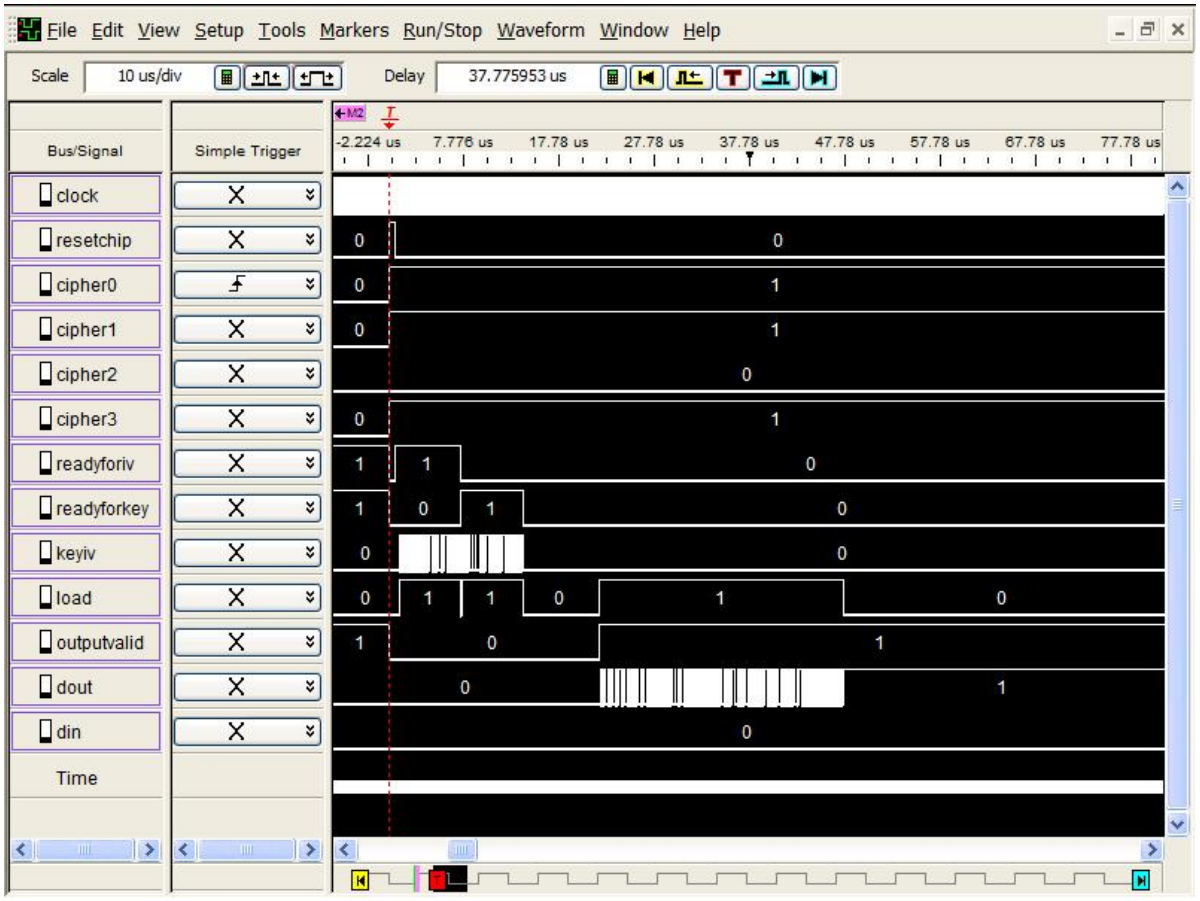

Figure 2.7: Result from the Logic Analyzer (Mickey).

Figure 2.7 shows the communication procedure of MICKEY encryption. First, a monotonic clock is applied to the ASIC where the frequency is 1MHz (maximum allowed operation frequency for the ASIC is 50 MHz). Afterwards, the target cipher has to be selected by the 4-bit cipher select pins. For example, in our case "1101" stands for choosing the MICKEY encryption. Next, the chip has to be reset and the FPGA drives the reset signal high for 5 clock cycles. When the ASIC is ready to receive the key or iv, the FPGA sends out the key or iv respectively with keeping the load signal high. In the case of MICKEY, both the key and iv are 128 bits long. After receiving the entire key and iv correctly, the ASIC starts to run the cipher. On the other hand, the FPGA waits for the output_valid signal. Once it is set high, the FPGA is allowed to input the plaintext and read the corresponding ciphertext one clock cycle later. And once again, the load signal has to be set high while the FPGA sends the plaintext. In figure 2.7 the plaintext contains 512 consecutive zeros.

Figure 2.8 in the next page shows the communication procedure of the Trivium encryption, which is almost the same as MICKEY but with three exceptions. The first difference is the loading sequence of the key and iv. For the case of MICKEY, iv is first loaded. But for Trivium key is first required. The second difference is the length of the key and iv which are set as 80 bits consecutive zeros in our test. The third difference is that the bit ordering for the key, IV and keystream is quadbyte swapped in contrast to MICKEY.

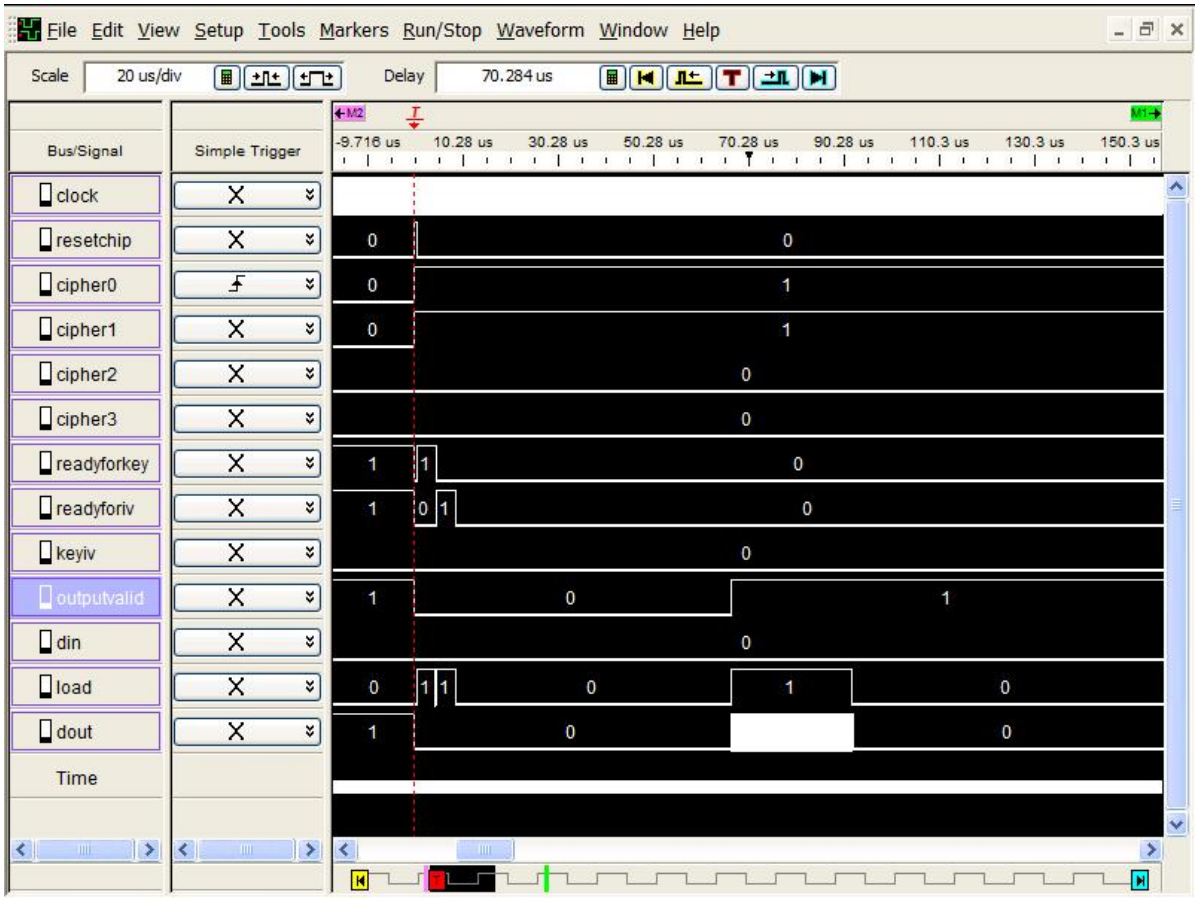

Figure 2.8: Result from the Logic Analyzer (Trivium).

At the end, we give a general overview shown in table 2.4 of the design summary report generated from the Xilinx ISE design tool. The design of the interface for MICKEY consumes about 200 slices and 1 Digital Clock Manager (DCM) to divide the default 100 MHz system clock into 1 MHz.

Table 2.4: Design summary of Mickey interface in ISE.

Target Device		xc2vp30 - 7ff896	
Product Version		ISE 10.1.03 - WebPACK	
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	198	13696	1%
Number of Slice Flip Flops	286	27392	1%
Number of Slice Flip Flops	286	27392	1%
Number of bonded IOBs	18	556	3%
Number of GCLKS	2	16	12%
Number of DCMS	1	8	12%

2.3 Summary

In this chapter, we have introduced the targeted cryptographic device under attack. The design and implementation of the interface between the ASIC and the PC is discussed in detail as well. Next, before we perform the attack, some background theory about stream ciphers and physical attacks will be introduced.

Chapter 3

Introduction of Stream Ciphers

3.1 Stream Ciphers in a nutshell

In the world of cryptosystem, there exist secret key and public key systems, which correspond to the symmetric and asymmetric systems respectively. Symmetric indicates the encryption and decryption use the same key while for the asymmetric case different keys are used for the encryption and decryption. If we further divide the symmetric system into two subgroups, we get block cipher and stream cipher. They mainly differ in the amount of the message that is processed during each encryption or decryption. Block ciphers work on blocks of data usually, for instance, 64, 128 or 256 bit blocks. On the other hand, stream ciphers work on bit or byte at a time. A stream cipher always runs faster due to the reason that a stream cipher performs less complex operations compared to a block cipher. In reality, stream ciphers are used in a wide range of applications. For example, GSM mobile phones use A5/1 encryption or some secret hot lines use stream ciphers. The general cipher encryption/decryption process is shown in figure 3.1.

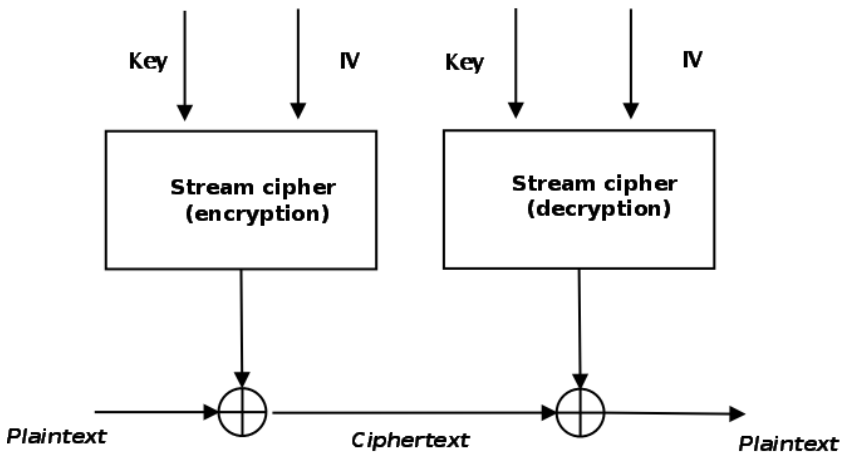

Figure 3.1: The general stream cipher process.

We could informally divide the stream ciphers into two types based on how the internal state of

the cipher is updated: if the next state is defined independently of the plaintext or ciphertext, the cipher is classified as a synchronous stream cipher. In contrast, a self-synchronizing stream cipher uses parts of the previously generated ciphertext to compute the keystream.

The most common component in a stream cipher is a linear feedback shift register (LFSR). In general, an LFSR consists of several flip-flops and a feedback network. Figure 3.2 shows the working principle of an LFSR.

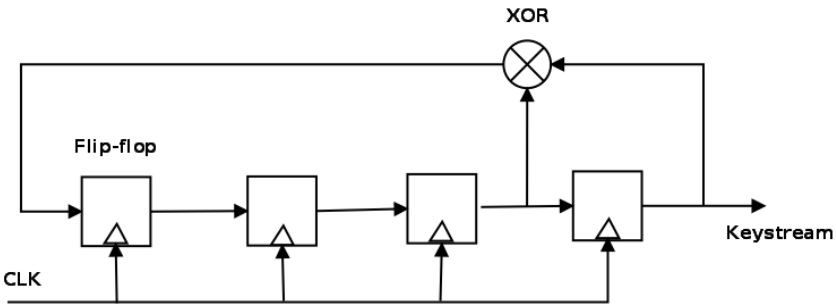

Figure 3.2: Linear feedback shift register.

However, the standalone usage of LFSRs is extremely insecure since an LFSR is inherently linear. Thus, a combination of several LFSRs can provide some level of security. One approach to introduce non-linearity is to have the clocking of one LFSR controlled by the output of a second LFSR. For example, the A5/1 algorithm in GSM is a combination of three linear feedback shift registers with irregular clocking.

3.2 Introduction of MICKEY-128

MICKEY-128 is a stream cipher developed by Steve Babbage and Matthew Dodd. The cipher is a Phase 2 Focus candidate for the eSTREAM project. The cipher is targeted for fast hardware implementations and efficient implementations for embedded systems. Several versions of MICKEY-128 with different key lengths exists. The version selected for Phase 2 has a key length of 128 bits. MICKEY-128 (which stands for Mutual Irregular Clocking KEYstream generator with a 128-bit key) is aimed at resource-constrained hardware platforms, but where a key size of 128 bits is required.

Please note that this section, 3.2 is mainly quoted from the algorithm description of MICKEY-128 2.0 [25].

3.2.1 Introduction

MICKEY-128 2.0 is intended to have low complexity in hardware, while providing a high level of security. It uses irregular clocking of shift registers, with some novel techniques to balance the need for guarantees on period and pseudorandomness against the need to avoid certain cryptanalytic attacks.

3.2.2 Input and output parameters

MICKEY-128 2.0 takes two input parameters:

- a 128-bit secret key K, whose bits are labelled $k_0...k_{127}$;
- an initialisation variable IV, anywhere between 0 and 128 bits in length, whose bits are labelled $iv_0...iv_{IVLENGTH-1}$.

The keystream bits output by MICKEY-128 2.0 are labelled z_0,z_1,.... Ciphertext is produced from plaintext by bitwise XOR with keystream bits, as in most stream ciphers.

3.2.3 Acceptable use

The maximum length of keystream sequence that may be generated with a single (K,IV) pair is 2^{64} bits. It is acceptable to generate 2^{64} such sequences, all from the same K but with different values of IV. It is not acceptable to use two initialisation variables of different lengths with the same K. And it is not, of course, acceptable to reuse the same value of IV with the same K.

3.2.4 Components of the keystream generator

3.2.4.1 The registers

The generator is built from two registers R and S. Each register is 160 stages long, each stage containing one bit. We label the bits in the registers $r_0...r_{159}$ and $s_0...s_{159}$ respectively. Broadly speaking, we think of R as the "linear register" and S as the "non-linear register".

3.2.4.2 Clocking the register R

Define a set of feedback tap positions for R:($RTAPS$=)

{0,4,5,8,10,11,14,16,20,25,30,32,35,36,38,42,43,46,50,51,53,54,55,56,57,60,61,62,
63,65,66,69,73,74,76,79,80,81,82,85,86,90,91,92,95,97,100,101,105,106,107,108,
109,111,112,113,115,116,117,127,128,129,130,131,133,135,136,137,140,142,145,148,
150,152,153,154,156,157}

We define an operation CLOCK_R (R, $INPUT_BIT_R$, $CONTROL_BIT_R$) as follows:

- Let $r_0 ... r_{159}$ be the state of the register R before clocking, and let $r'_0 ... r'_{159}$ be the state of the register R after clocking.

- $FEEDBACK_BIT = r_{159} \oplus INPUT_BIT_R$

- For $1 \leq i \leq 159$, $r'_i = r_{i-1}$; $r'_0 = 0$

- For $0 \leq i \leq 159$, if $i \in RATPS$, $r'_i = r'_i \oplus FEEDBACK_BIT$

- If $CONTROL_BIT_R = 1$:
  - For $0 \leq i \leq 159$, $r'_i = r'_i \oplus r_i$

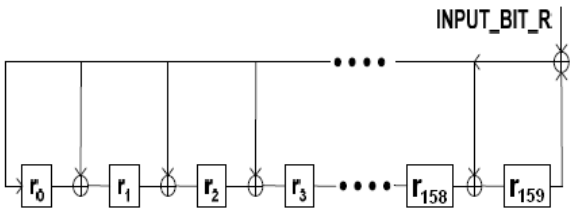

Figure 3.3: Clocking the R register with CONTROL_BIT_R = 0.

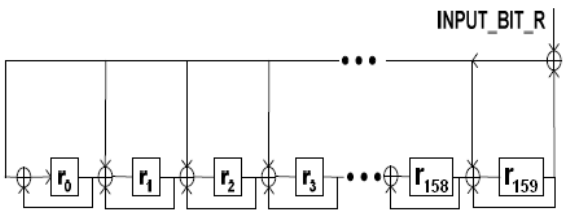

Figure 3.4: Clocking the R register with CONTROL_BIT_R = 1.

3.2.4.3 Clocking the register S

Define four sequences $COMP0_1 ... COMP0_{158}, COMP1_1 ... COMP1_{158}, FB0_0 ... FB0_{159}, FB1_0 ... FB1_{159}$ as follows:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
$COMP0_i$		1	1	1	1	0	1	0	0	1	0	0	1	1	1	1	0	1	1	0	1	0	1	1	1	0	1
$COMP1_i$		0	0	0	1	1	0	0	1	1	1	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1
$FB0_i$	1	1	1	1	0	1	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	1
$FB1_i$	1	1	0	1	0	1	0	1	1	1	1	0	1	1	1	0	0	0	1	0	1	1	1	1	1	1	0

i	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
$COMP0_i$	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0	0	0	0	1	1	0	0	1
$COMP1_i$	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	1	1	1	1	0	0	0	1	1	0	1	1
$FB0_i$	0	0	0	1	1	0	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	1	1	1	1	0	1
$FB1_i$	1	1	0	0	1	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	1	1	0	0	1	1	1

i	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
$COMP0_i$	0	0	1	0	0	1	1	1	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0
$COMP1_i$	0	1	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	0
$FB0_i$	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	1	0	0	1	0	1
$FB1_i$	1	0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	0	0	0

i	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107
$COMP0_i$	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	1	0	0	0	1	1	0	0	1	0	0	1
$COMP1_i$	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	0	0	0	1	0	1	1	0	0
$FB0_i$	0	1	0	0	1	1	1	0	1	1	0	0	1	1	0	1	0	0	0	1	0	0	1	1	1	0	1
$FB1_i$	1	0	1	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1	0	0	1	1	1	0	1	1	1

i	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134
$COMP0_i$	1	0	1	1	1	1	1	1	0	1	0	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0
$COMP1_i$	0	1	1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	1	0
$FB0_i$	0	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1	0	0	0	0	0	1
$FB1_i$	1	1	1	0	1	1	0	1	0	0	1	0	0	0	1	1	0	1	1	0	1	1	1	1	0	1	1

i	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159		
$COMP0_i$	1	0	1	1	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	0	0	0			
$COMP1_i$	1	1	1	0	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1			
$FB0_i$	1	1	1	0	1	0	0	0	0	1	1	0	0	0	1	1	0	1	1	0	0	0	0	0	1		
$FB1_i$	1	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	1	1	0	0	0	1	0	0	0		

Figure 3.5: Table of sequences needed by clocking S.

We define an operation CLOCK_S (S, $INPUT_BIT_S$, $CONTROL_BIT_S$) as follows:

- Let $s_0 ... s_{159}$ be the state of the register S before clocking, and let $s'_0 ... s'_{159}$ be the state of the register after clocking. We will also use $\hat{s}_0 ... \hat{s}_{159}$ as intermediate variables to simplify the specification.

- $FEEDBACK_BIT = s_{159} \oplus INPUT_BIT_S$
- For $1 \leq i \leq 158$, $\hat{s}_i = s_{i-1} \oplus ((s_i \oplus COMP0_i) \cdot (s_{i+1} \oplus COMP1_i))$; $\hat{s}_0 = 0$; $\hat{s}_{159} = s_{158}$.
- If $CONTROL_BIT_S = 0$:
  - For $0 \leq i \leq 159$, $s'_i = \hat{s}_i \oplus (FB0_i \cdot FEEDBACK_BIT)$
- If $CONTROL_BIT_S = 1$:
  - For $0 \leq i \leq 159$, $s'_i = \hat{s}_i \oplus (FB1_i \cdot FEEDBACK_BIT)$

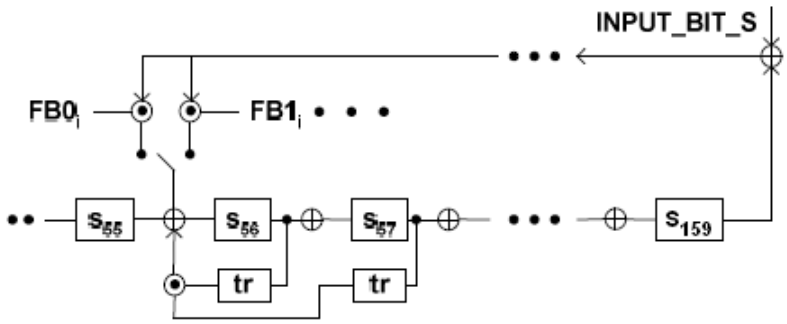

Figure 3.6: Clocking the S register.

3.2.4.4 Clocking the overall generator

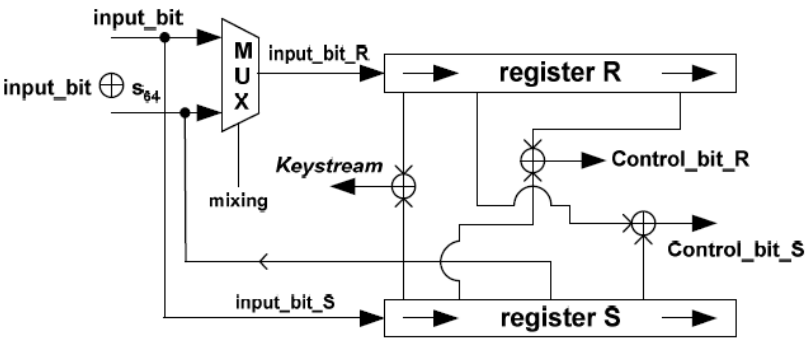

Figure 3.7: The MICKEY-128 Stream Cipher Architecture.

We define an operation CLOCK_KG (R, S, $MIXING$, $INPUT_BIT$) as follows:

- $CONTROL_BIT_R = s_{54} \oplus r_{106}$
- $CONTROL_BIT_S = s_{106} \oplus r_{53}$

- If $MIXING = TRUE$,
  - CLOCK_R (R, $INPUT_BIT_R = INPUT_BIT \oplus s_{80}$, $CONTROL_BIT_R = CONTROL_BIT$)
  - CLOCK_S (S, $INPUT_BIT_S = INPUT_BIT$, $CONTROL_BIT_S = CONTROL_BIT$)
- If $MIXING = FALSE$,
  - CLOCK_R (R, $INPUT_BIT_R = INPUT_BIT$, $CONTROL_BIT_R = CONTROL_BIT$)
  - CLOCK_S (S, $INPUT_BIT_S = INPUT_BIT$, $CONTROL_BIT_S = CONTROL_BIT$)

3.2.5 Key loading and initialisation

The registers are initialised from the input variables as follows:

- Initialise the registers R and S with all zeros.
- (Load in IV.) For $0 \leq i \leq IVLENGTH - 1$:
  - CLOCK_KG (R, S, $MIXING = TRUE$, $INPUT_BIT = iv_i$)
- (Load in K.) For $0 \leq i \leq 127$:
  - CLOCK_KG (R, S, $MIXING = TRUE$, $INPUT_BIT = k_i$)
- (Preclock.) For $0 \leq i \leq 159$:
  - CLOCK_KG (R, S, $MIXING = TRUE$, $INPUT_BIT = 0$)

3.2.6 Generating keystream

Having loaded and initialised the registers, we generate keystream bits $z_0 \ldots z_{L-1}$ as follows:

- For $0 \leq i \leq L - 1$:
  - $z_i = r_0 \oplus s_0$
  - CLOCK_KG (R, S, $MIXING = FALSE$, $INPUT_BIT = 0$)

3.2.7 The intended strength of the algorithm

MICKEY-128 2.0 is intended to resist any attack faster than exhaustive key search. The designers have not deliberately inserted any hidden weaknesses in the algorithm.

3.2.8 Performance of the algorithm

MICKEY-128 2.0 is not designed for notably high speeds in software, although it is straightforward to implement it reasonably efficiently.

There may be scope for more efficient software implementations that produce several bits of keystream at a time, making use of look-up tables to implement the register clocking and keystream derivation.

Since MICKEY-128 is intended to be built on resource-constrained hardware platforms while having low complexity, its area consumption is rather small compared to the other candidates in the eSTREAM project. The chart for the comparison is shown in figure 3.8.

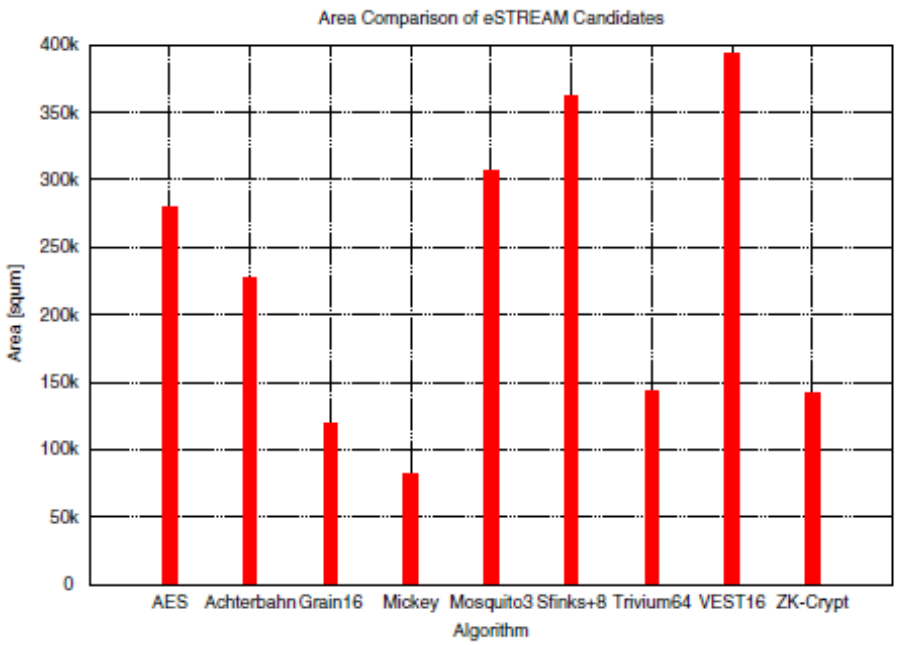

Figure 3.8: Synthesis result of area consumptions for eSTREAM candidate algorithms, compared to an efficient AES implementation. [2]

Chapter 4

Theoretical background of Side-Channel Attacks

In general, cipher attacks can be performed based on brute force or theoretical weaknesses in the algorithm. However there is another option named side-channel attack which is defined as any attack with using the information extracted from the physical implementation of a cryptosystem. In other words, it takes advantage of implementation characteristics to reveal the secret key of the attacked device. The typical sources of side-channel information produced by the encryption/decryption processes of the cryptographic devices are: timing information, power consumption and electromagnetic emanation.

The most commonly used approaches of categorizing physical attacks are introduced as:

Active vs. passive: active attacks try to make the attacked device behave irregularly and reveal the secret key according to the irregularly behavior. Differential fault analysis is an active attack which influences the operation by inducing errors. On the other hand, the passive attacks simply observe the device's normal behavior during its process, within the specification. Side-channel attacks belong to the latter group.

Invasive vs. non-invasive: invasive attacks require physically taking the attacked device apart to make a direct access to the internal component. For the case of non-invasive attacks, such as side-channel attacks, they only use external information like power consumption or time consumption and so on. In [26], Skorobogatov and Anderson add a new distinction with what they call semi-invasive attacks. In this kind of attacks, the device is also depackaged. However, no direct electrical contact to a chip surface is made - the passivation layer stays intact.

This thesis will relate only to the most common types of side-channel attacks, which are: Timing attacks, electromagnetic analysis attacks and power consumption attacks. We will introduce them respectively in the following sections and put more emphasis on power consumption attacks.

4.1 Timing Attack

In cryptography, a timing attack is a simple kind of side-channel attacks. It is first introduced by Kocher [10] and it takes the advantage of the time taken to execute different cryptographic algorithms. Most cryptosystems have a common characteristic that different amounts of time are needed to process different inputs. For example, the difference may come from the performance optimization, which means that bypassing the unnecessary operations can cause a different time consumption in total. Such different time consumption with corresponding input messages can be used to reveal the secret information with the aid of specific statistical models such as correlation.

For example, there is a simple algorithm involves a long time operation and a short time operation. These two operations are executed depend on the value of the key bit and they never run simultaneously. If the key bit equals 1 the long time operation is executed and if the key bit equals 0 the other one is executed. Hence, we can reveal this key bit by recording the time consumption of the algorithm.

Instead of the algorithm itself, the timing attack exploits useful information from the physical implementation of the algorithm. Against timing attack, effort should be put into the implementation of the algorithm. For example, design the implementation such that every call to a subroutine always returns in constant time. In such an implementation, timing information is not useful anymore for attack.

4.2 Power Consumption Attacks

Power analysis attack is another form of side-channel attacks based on analyzing the power consumption of a cryptographic device (such as a smart card, tamper-resistant "black box", or integrated circuit) while it performs the encryption operation. The attacker can non-invasively extract secret keys and other information from the device by combining with other cryptanalysis techniques. Power analysis attacks generally consist of two catalogues, namely Simple Power Analysis and Differential Power Analysis. They were first introduced by Kocher et al. in [7].

Simple power analysis (SPA) involves visually inspecting power traces while an encryption operation is being performed over time. Differential power analysis (DPA) is carried out in a more advanced form which can allow an attacker to extract the intermediate values within a cryptographic operation by applying a suitable statistical model on data collected from multiple cryptographic operations.

A general power analysis attack workflow is described as follows and also shown in figure 4.1: it compares observations of the power consumption with estimations of the power consumption. The hypothetical power consumption comes from a power model of the cryptographic device requiring guesses on the secret key. The correct key is revealed by finding the best match between the measurements and the hypothetical power consumption from different key guesses.

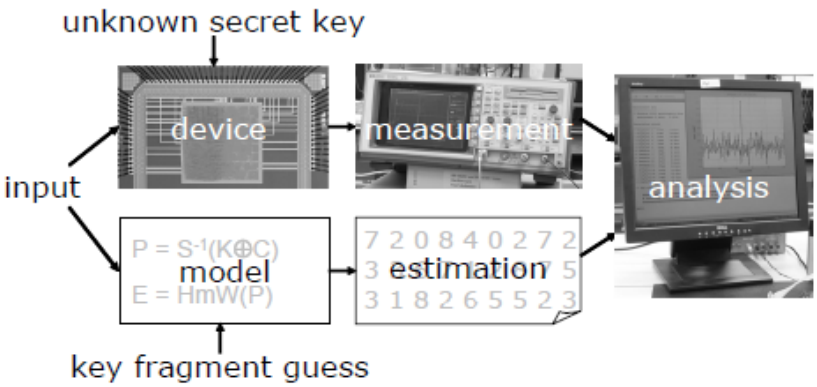

Figure 4.1: Power analysis methodology.

4.2.1 Power consumption

In this section, we discuss the power consumption of cryptographic devices in details. The total power consumption of a CMOS circuit is composed of the power consumptions of the logic cells which make up the circuit. Hence, the total power consumption is closely related to the number of logic cells in a circuit, the connections between them and the fact how the cells are built. We take an example of a CMOS inverter to explore when the CMOS cells dissipate power. A typical inverter is shown in figure 4.2, which consist of two transistors $P1$ and $N1$.

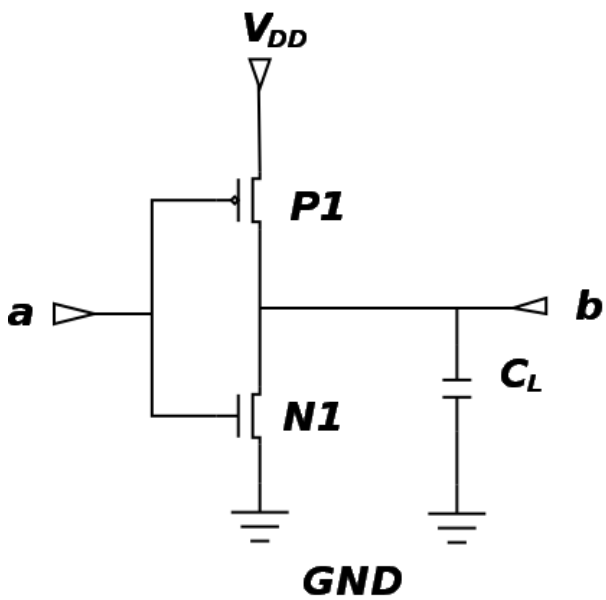

Figure 4.2: A CMOS inverter.

The power consumption of a CMOS inverter can generally be divided into two parts. One is static power consumption P_{stat}. This power is consumed in the condition that there is no switching activity in a cell, it is mainly caused by the leakage current. The other one is

dynamic power consumption P_{dyn}. The dynamic power consumption is the result of charging and discharging the load capacitance and it is the most important part in the total power consumption of a CMOS device. It is data dependent and can be regarded as a function of the switching activity in a circuit. Hence, the total power consumption of a CMOS cell is the sum of P_{stat} and P_{dyn}.

4.2.2 Simple Power Analysis

Simple power analysis (SPA) is a side-channel attack which involves visual inspections of the graphs of current used by a device over time. Variations in power consumption happen as the device performs different operations. For example, different instructions executed by a microprocessor will lead to different power consumption profiles.

Simple power analysis has the ability to reveal the sequence of instructions executed. Therefore, when the attacked cryptographic device involves some operations which depend on the processing data, SPA is applicable. Such operations can be comparison or multiplication. Hence, most of the cryptographic units were vulnerable to SPA attacks.

4.2.3 Differential Power Analysis

Differential power analysis attacks are the most popular type of power analysis attack nowadays. During our thesis work DPA attacks are the main method which we used for revealing the secret key, so we are going to tell the story about DPA in more details.

In contrast to SPA attacks, DPA attacks do not require detailed knowledge about the cryptographic device but on the other hand, they do require a large amount of power traces. Therefore, physically possessing a cryptographic device for a long time turns out to be necessary for the attack in our case. In this section we provide a comprehensive introduction to typical DPA attacks.

4.2.3.1 General description

Differential power analysis is a side-channel attack which consists of both a visual method and a statistic analysis method. The goal of DPA attacks is to reveal the secret keys based on a large amount of power traces. The most important difference between SPA and DPA is the way they analyze the power traces. In SPA attacks, the attacker is most interested in analyzing the shape of power consumption along the time axis. But in the case of DPA attacks, the relation between the power consumption at a fixed moment and the processed data is the most interesting part. Therefore DPA attacks emphasize on the data dependency of the power traces.

Now we discuss in detail how DPA attacks reveal the secret key of a cryptographic device. Here is a general strategy for the operation:

Step 1: Measuring the power consumption

In a DPA attack we start with measuring the power consumption of encryptions or decryptions. The DPA attack involves statistical analysis at the end, it sometimes requires large amount of different power traces, so the encryption or decryption has to be executed a lot of times for different data blocks.

Step 2: Finding the hypothetical power consumption and the correlation with the key

In this step, we first look for a proper intermediate value which is strongly related to the power consumption. Afterwards, we transform this intermediate value to the hypothetical power consumption by using different power models. Power models influence the attack result depends on how well they describe the power consumption. Normally, power models include bit model, Hamming-Weight model, Hamming-Distance model and Zero-Value model.

Step 3: Implementing statistical analysis on the power consumption and hypothetical power consumption values

In the last step, we use different statistical analyses to find out the relationship between the power consumption and hypothetical power consumption. The most related hypothetical power consumption corresponds to the correct key hypothesis. By this way the entire secret key can be revealed. Different statistical analyses have different effects on the attack result, for instance the most widely known analyses are, Correlation Coefficient [27], Difference of Means [7], Distance of Means [28] and Maximum-Likelihood testing [29].

As described in step 2 and 3, there are different power models and statistical analyses. Hence, the DPA attacks can be categorized based on different combinations. In the following, each of them is treated separately.

4.2.3.2 Statistical Analysis: Attacks based on the Correlation Coefficient

The Pearson correlation coefficient is the most widely used way to determine the linear relationships between data. Hence, it is an excellent choice for statistical analysis tool when it comes to perform DPA attacks. Assume there are two sets of data T (real power consumption measurements) and H (hypothetical power consumption values), the correlation can be calculated according to equation 4.1.

$$C(T,H) = \frac{E(T \cdot H) - E(T) \cdot E(H)}{\sqrt{Var(T) \cdot Var(H)}} \tag{4.1}$$

Where $E(T)$ denotes the expected value (mean) of the measurement data of the set of measurements T and $Var(T)$ the variance. If this correlation is high, it is assumed that the prediction of the hypothetical power consumption is correct.

4.2.3.3 Statistical Analysis: Attacks based on the Difference of Means

Besides the correlation coefficient method there is another method that can be used as statistical analysis to determine the relationship between data, which is called difference of means method. The basic operation of this method is that the attacker splits the hypothetical power consumption into two sets based on the statistical view of the intermediate values. Afterwards the real power consumption is also split into two sets according to the corresponding hypothetical power consumption. Further, one adds all the real power consumption values together in each set and then calculates the two corresponding mean values. At the end compute the difference between these two mean values, if there is a significant difference then the corresponding hypothetical power consumption is correct. By repeating this process the entire secret key can be revealed.

Theoretically, the DPA attacks based on the difference of means method should require more power traces than the case of correlation coefficient. The reason is that the correlation coefficient method considers the differences of values as well as variances, but difference of mean method only focuses on the differences.

4.2.3.4 Statistical Analysis: Attacks based on the Distance of Means

Distance of means method is a variation of the difference of means method and it has an improvement compare to the latter. In this attack the power consumption is split into two sets for each key hypothesis in exactly the same way as the previous method. The difference only occurs during the comparison between the means of the two sets. For difference of means one just subtracts the means, for the distance of means one needs to further divide the difference of means by the standard deviation of the difference distribution of the two sets. The standard deviation is estimated by the square root of the pooled variance ${s_p}^2$ in equations 4.2 and 4.3 as described in [28].

$$ {s_p}^2 = \frac{(n-1) \cdot {s_X}^2 + (m-1) \cdot {s_Y}^2}{m+n-2} \tag{4.2} $$

$$ s_{\bar{X}-\bar{Y}} = s_p \cdot \sqrt{\frac{m+n}{m \cdot n}} \tag{4.3} $$

Where X and Y denote two sets of traces, and $s_{\bar{X}-\bar{Y}}$ denotes the estimation of the standard deviation of the difference distribution of the two sets X and Y, with the number of power traces are n and m respectively.

The advantage of distance of means compared to difference of means is, it considers both variances and differences as the correlation coefficient method. Hence, theoretically, the attack requires about the same number of power traces as the attack based on the correlation coefficient.

From now on we assume correlation coefficient is always chosen as the statistical analysis model, different power models can be applied to perform the attack.

4.2.3.5 Power Model: Attacks based on the Bit Model

The power model in the second step of the DPA attack is used to find the hypothetical power consumption values. The general rule is the better the used power model describes the power consumption of the attacked device, the less power traces are needed for revealing the secret key. Hence, the power model has a significant influence on the DPA attack. We start with the simplest one, which is called bit model that only considers one specific bit of the intermediate value. The observation is that this power model is sufficient for software implementation of the attacked algorithms but not enough for hardware implementation.

4.2.3.6 Power Model: Attacks based on the Hamming-Weight Model

In the case of hamming-weight model, the attacker assumes that the power consumption is proportional to the number of bits that are set in the processed data value, but the data values that are processed before and after this value are ignored. Therefore hamming-weight is another option of the power model. Instead of bit model, hamming-weight model is assumed to be better to describe the power consumption. However, they have a similar problem that they are effective with the attack on software implementation of algorithm but not on hardware implementation. The reason is that the bit model and the hamming-weight model typically describe the power consumption of CMOS circuits very badly. In other words, these two models consider the state of the bits. But the power consumption of a CMOS circuit depends on whether there occurs transition or not, not on the processed value.

4.2.3.7 Power Model: Attacks based on the Hamming-Distance Model

The problem met with the previous two power models has been solved by the hamming-distance model, which describes the power consumption of CMOS circuits much better than the hamming-weight and bit model. However, in order to use this model the attacker is required to have some additional knowledge about the attacked device. Essentially, the attacker needs to know the state of a cell in the circuit before or after it processes the attacked intermediate result.

As rule of thumb, attacks based on the hamming-distance model require less power traces than the previous two models for the case of hardware implementation, because of the better description of the power consumption it has.

4.2.3.8 Power Model: Attacks based on the Zero-Value Model

At the end we introduce the last power model which is the zero-value model. This model is suitable for the attack when we can exploit some useful information from some special computation. Let's take an example of multiplication. If the input is zero which means something multiplies by zero, this leads to less power consumption. Otherwise the multiplication requires more power.

After seeing four kinds of power models, it can be concluded that when the attacks are based on the correlation coefficient, the bit model and HW model only lead to small correlation

coefficient while the HD and ZV model lead to much higher value, the HD and ZV have better performance than the HW and bit model.

4.3 Electromagnetic Analysis Attack

The flow of electrical power can be seen as the propagation of electromagnetic waves. Therefore, the leaked electromagnetic radiation can directly provide plaintexts and other information. Electromagnetic attacks, first introduced by Quisquater and Samyde [30] and [9], and further developed in [31], compromise information being transfered by means of EM emanation.

Timing attack, power analysis and electromagnetic analysis can be regarded perspectively as increasingly-dimensional side channels. Timing attack provides a single dimentional scalar (the running time) for each measurement. While power analysis provides a vector showing, at each time location, the corresponding power consumption. Electromagnetic analysis allows to build a 3 - dimensional map of the magnetic field's evolution along time, thus providing 4 - dimensional information. This allows for example to separate the contributions of various components of the chip, and therefore to study them separately. This is actually also a type of hardware "divide-and-conquer" approach.

Since the EM emanation is related to the power consumption, techniques like SPA and DPA are applicable as well. The information measured can be analyzed in the same way as power consumption which are simple and differential electromagnetic analysis. In this case, they are called SEMA and DEMA respectively. Compared to power consumption analysis, there are several aspects worth noticing which make EM analysis more valuable at some points. First of all, for carrying out electromagnetic analysis, no direct connection to the device under attack as in the case of power analysis is needed. This property makes this type of attack very attractive to attackers in the situation where getting physical contact with the cryptographic device is difficult. Secondly, it is shown in [31] that these emanations consist of a multiplicity of signals, each leaking somewhat different information about the underlying computation. In the literature they also sort the EM emanations in two main categories: direct emanations, i.e., emanations that result from intentional current flow, and unintentional emanations, caused by coupling effects between components in close proximity. According to them, exploiting unintentional emanations can be much more effective that trying to work with direct emanations. It is also shown that the EM side-channel analysis is very useful even in cases where the power consumption analysis is infeasible, i.e., the EM side-channel analysis can be used to break power analysis resistant implementations.

Essentially, EM analysis is a non-invasive attack, as it is measuring the near field EM emanations. However, this attack can be made much more efficient by depackaging the chip first, to allow closer measurements and to avoid disturbance due to the passivation layer.

Countermeasures used to defeat power analysis attacks can also be applied against EM analysis attacks. In addition, shielding can be a valuable approach to decrease EM side-channel leakage. This is actually the main barrier for EM analysis.

EM capturing equipment typically includes a high performance oscilloscope capable of 2GSa/s, amplifier, antennas (Far-field) and near-field probes. An example of a loop atenna measuring the EM radiation from the FPGA is shown in figure 4.3.

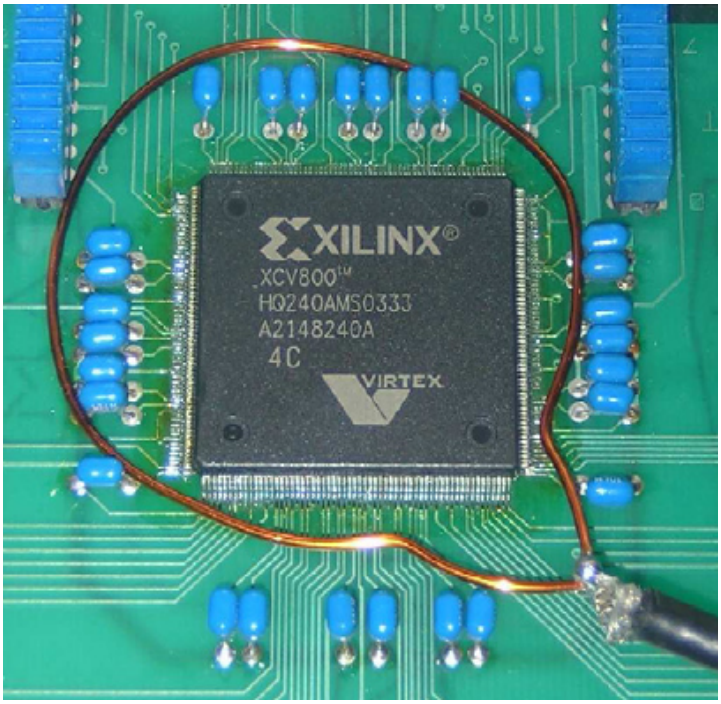

Figure 4.3: A loop antenna is placed parallel with the FPGA under measurement. [3]

4.4 Preventing Side-Channel Attacks

In this part of the chapter, we present some known techniques that have been discussed in the published literature. We first introduce some general countermeasures which are applicable to all attacks. Furthermore, we also discuss some specific countermeasures against the attacks we described before.

Generally, all operations that are performed by the module shall provide data-independency in their time consumption. The general feature of making the time needed for fixed execution for every block of data prevents all timing attacks. This can be achieved by adding delays to balance the time difference between two operations.

In [7], it is shown that avoiding procedures that use secret intermediate values or keys for conditional branching operations will mask many SPA characteristics.

In a software implementation, the branching statements or conditional statements should not appear in the critical part of the codes. This added feature can make it very difficult for the attacker to guess the key using measurements of time or power consumption. The reason behind it is, when the critical portion of the codes is always running independent of the input and key, the time and power taken to operate these commands does not depend on the data and thus do not leak any information of its properties. It is shown that in the research during recent years, the most effective approach against DPA attacks involves modification of the algorithm itself. This may solve the problem, but it does require design changes both in the algorithms

and protocols themselves if we refer to hardware implementation. Sometimes it is likely to be a problem, since the resulting product is not compatible with standards and specifications anymore.

Introducing noise into power consumption measurements increases the number of samples needed for a particular attack, possibly to an infeasible large number. One suggestion in [32] is to prevent DPA attacks adding noise caused by random calculations. It increases the noise level high enough to make the DPA bias spikes undetectable. The results presented in [32] also give some indication of how much noise should be added enough to stop an attack but yet not to bring so much overhead.

In practice, compact physical shielding can make both power and EM analysis attacks unfeasible.

4.5 Summary

In this chapter, we have briefly discussed several different types of side-channel attacks, especially focused on the differential power analysis which is the main method we use in our thesis work. In the following chapter we are going to represent what we have achieved by applying DPA attacks on the hardware implementation of the stream cipher MICKEY.

Chapter 5

Practical attack on the hardware implementation of the MICKEY Stream Cipher

In the previous chapter, the basic operation flow and different theoretical approaches of DPA attacks have been introduced carefully, now we are going to see the practical work for our thesis.

The second task of our thesis is to attack the hardware implementation of the MICKEY stream cipher by using DPA attacks. We have successfully revealed the secret key of MICKEY hardware implementation with different types of DPA attacks. In the following sections we will explain each applied attack method with their result separately and at the end a short comparison among them is included. In order to be clear enough, the explanation will follow the operation flow of a DPA attack which is described in section 4.2.3.1, step by step.

5.1 First Step: Measuring the Power Consumption

Measurement setups for power analysis attacks usually consist of several components that interact with each other. We list our measurement setup modules below.

Cryptographic Device: The device under attack is what we have already introduced before, namely the eSCARGOt ASIC. In order to mount power analysis attacks on this device, a dedicated printed circuit board was made prior to our thesis. For detailed information please refer to chapter 2.

Interface on FPGA: The interface between the ASIC and the PC is implemented on an FPGA which is capable of receiving commands and data from the PC, triggering the oscilloscope, sending data to the ASIC and sending the result back to the PC.

Power Supply: The ASIC and the interface do need an external power supply. The ASIC operates normally under 3.3V nominal supply and the internal core operates from a 1.8V nominal supply.

Probe: A small resistor (typically resistance value is 50 Ω) is inserted into the *GND* or V_{DD} line of the ASIC. The voltage drop across this resistor is proportional to the current that is flowing into the device. Assuming the voltage level of the power supply is constant, this voltage drop is also proportional to the power consumption of the device, which can be measured by a probe. Notice that the insertion of a resistor into a power supply line of the cryptographic device is the simplest way of building a power measurement circuit. There also exists other proposals for more complex power measurement circuits. An alternative way to measure the power consumption is to use a current probe. In our measurement setup, we use a probe to measure the power consumption at *GND* by inserting a resistor while a current probe is connected in series with V_{DD} to measure the current consumption which can be sampled by the oscilloscope into voltage variations. In figure 5.1, a current probe and a scope probe are connected with the PCB test board.

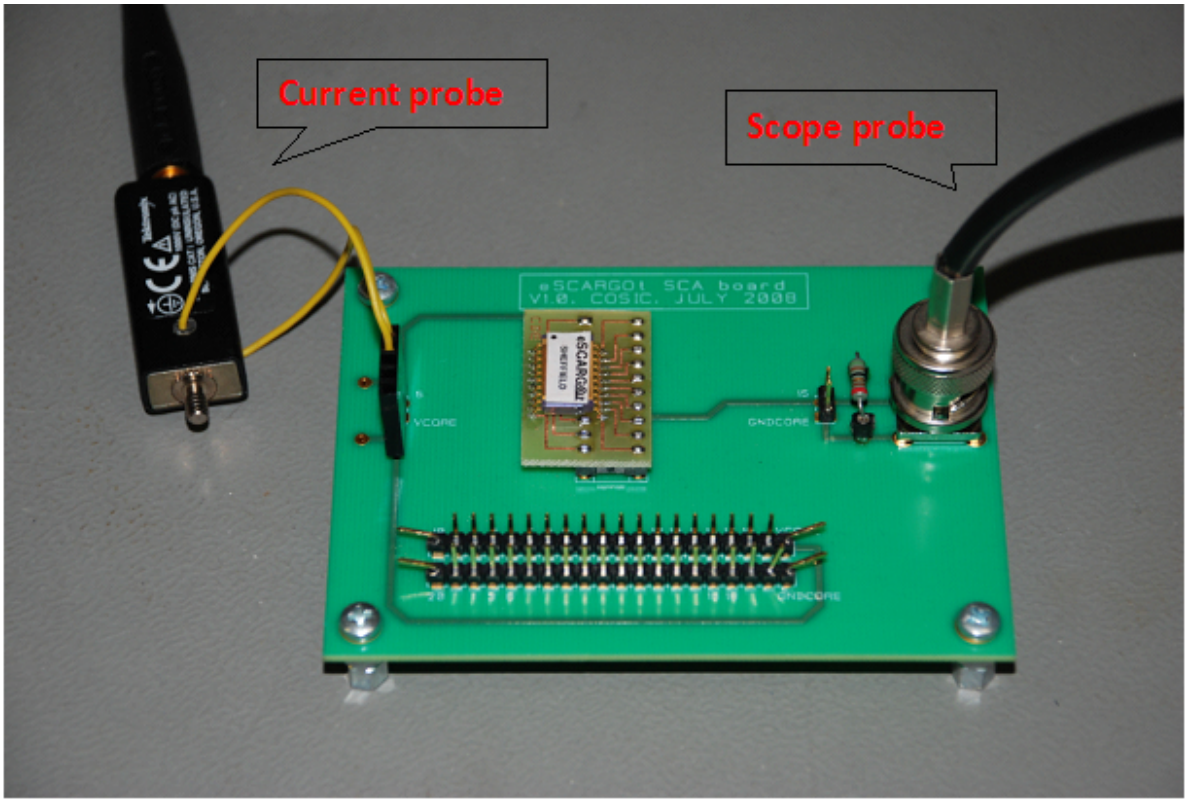

Figure 5.1: Test board with a scope probe and a current probe connected.

Digital Oscilloscope: The power consumption signal that is provided by the measurement circuit can be recorded by a digital sampling oscilloscope. It takes the analog signal as input,

converts it into a digital signal and stores it in its memory.

Personal Computer: The PC controls the whole measurement setup by running a matlab program and stores the measured power traces. It communicates both with the interface and the oscilloscope.

In order to successfully measure the power consumption of the ASIC while it executes a cryptographic algorithm, the listed components interact as follows: First the PC configures and arms the oscilloscope, then the PC sends commands and data to the FPGA via the RS232 serial interface. The FPGA forwards the data to the ASIC that start the encryption of MICKEY. At the same time, the FPGA also triggers the oscilloscope. During the execution, the oscilloscope records the power consumption of the ASIC by a probe. After the PC receives the output of MICKEY from the ASIC through the FPGA, the recorded power trace from the oscilloscope is stored in the PC via an Ethernet interface. These steps are repeated as often as necessary for the particular power analysis attack. The view of the whole measurement setup is shown in figure 5.2.

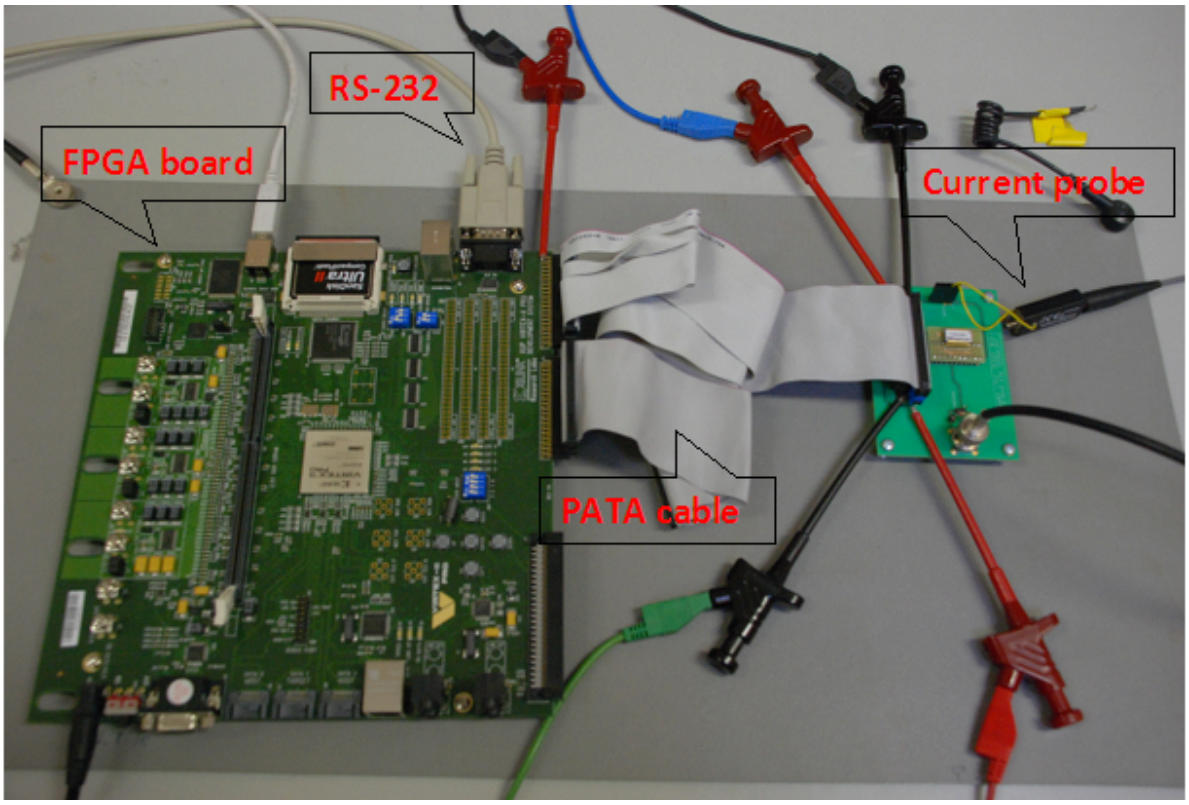

Figure 5.2: Measurement setup.

In order to make sure enough power traces are collected for different DPA attacks, we run 7000 encryptions with 7000 different IVs on the hardware implementation of MICKEY. Meanwhile, another software implementation of MICKEY is running with the same IV as the hardware version. While doing so, we can compare the ciphertexts between the hardware and software versions. If they generate the same ciphertext, the corresponding power trace is stored as a valid

one. If not, the corresponding power trace is dropped. At the end from the 7000 encryptions we collect 6000 valid power traces, with the length of 60000 points each. Figure 5.3 and figure 5.4 show two examples of power traces measured from V_{DD} and *GND* respectively. Note that here "samples" correspond to the points in time. In one power trace, the former 30000 samples are measured during the key loading phase while the rest are measured when the cipher is running.

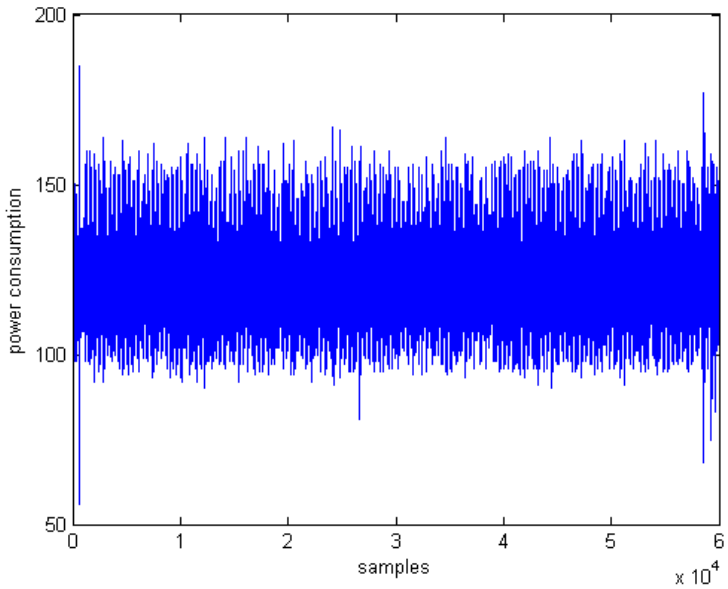

Figure 5.3: A power consumption trace measured at V_{DD}.

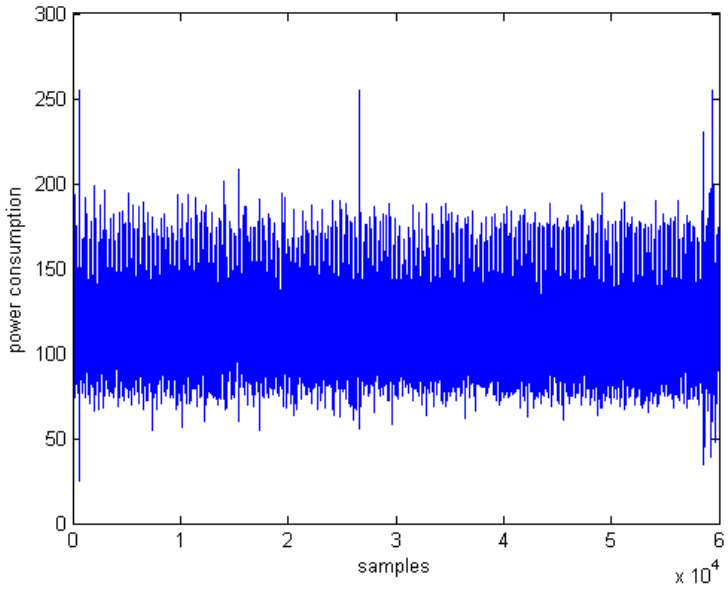

Figure 5.4: A power consumption trace measured at *GND*.

Finally we can build the power trace matrix **T** 5.5 of size $D \times 60000$, where D is smaller or equal to 6000.

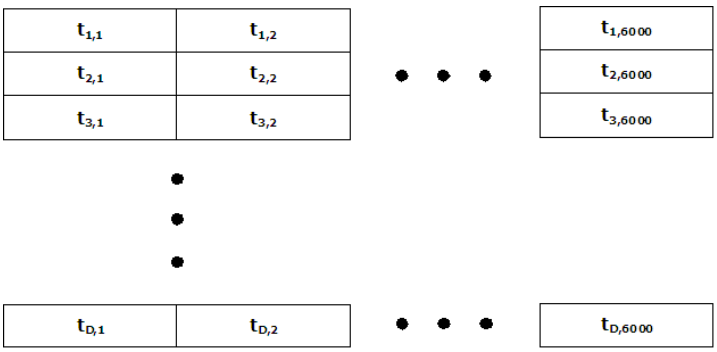

Figure 5.5: Matrix **T** contains the measured power traces.

5.2 Second step: Building the hypothetical power consumption matrix by different power models

The basic idea of this step is that, first look for intermediate values which can reflect the power consumption effectively, and then choose a proper power model to describe the power by building a hypothetical power consumption matrix.

First, let's take a look at the intermediate values. In our case the target cipher MICKEY is a simple algorithm, it only involves two registers, plus several multiplication and XOR functions. During the encryption the multiply and XOR are conditional operations, which means they are executed under some special conditions. So they can not reflect the power all the time, this is the reason why we choose the registers as our intermediate values. In the case of MICKEY, both the R and S registers are loaded by the initial vector first and then continue with the key. This also proves that these two registers can be chosen as intermediate values because their values are influenced by both the initial vector and part of the key. We are especially interested in the duration of key loading. After each bit of key or each byte of key is loaded, we consider the state of the registers with the power traces collected in step 1 to find the relationship between them. In other words, we check which key guess influences the power consumption correctly based on the view of the power traces.

Figure 5.6 shows an example of the intermediate values matrix **V** which is used to generate the hypothetical power consumption matrix in the future.

In this case, we choose both R and S registers as the intermediate values and also the states of R and S registers are captured after one bit of key is loaded. As we can see in the figure, matrix **V** only contains 2 columns which stand for two guessed key bits. The reason is we attack the key bit by bit, each bit can be guessed as 1 or 0. Each element of **V** stores the states of both R and S registers.

$v_{1,1}$	$v_{1,2}$	
$v_{2,1}$	$v_{2,2}$	
$v_{3,1}$	R	S
⋮		
$v_{D,1}$	$v_{D,2}$	

Figure 5.6: Matrix **V** contains the intermediate values.

In practice, we also choose register S alone as the intermediate values to perform the attack.

When the intermediate values matrix is available, we use a proper power model to generate the hypothetical power consumption matrix. Actually we use several different power models, for example:

- Hamming-Weight model
- Hamming-Distance model
- Zero-value model

Here in figure 5.7 is an example of the hypothetical power consumption matrix **H**.

$h_{1,1}$	$h_{1,2}$
$h_{2,1}$	$h_{2,2}$
$h_{3,1}$	$h_{3,2}$
⋮	
$h_{D,1}$	$h_{D,2}$

Figure 5.7: Matrix **T** contains the hypothetical power consumptions.

This matrix has the same size with **V**. Two columns again indicate two guessed key bit. For instance, this matrix **H** is generated by using the Hamming-Distance model, and then each

element of **H** is the number of how many bits flipped in the R and S registers after one key bit is loaded.

The difference among different power models is the way they describe the power consumption, which influences the final result of the entire attack significantly. In other words, a better power model leads to a better description of the power consumption. These three power models are used in the practical attack and will be treated later.

5.3 Third step: Attacks based on different statistical analyses

In the third step we apply different statistical analysis methods to calculate the correlation between the hypothetical power consumption matrix and the power trace matrix, where the correlation can help us to find out the correct key guess. The general procedure is shown in figure 5.8.

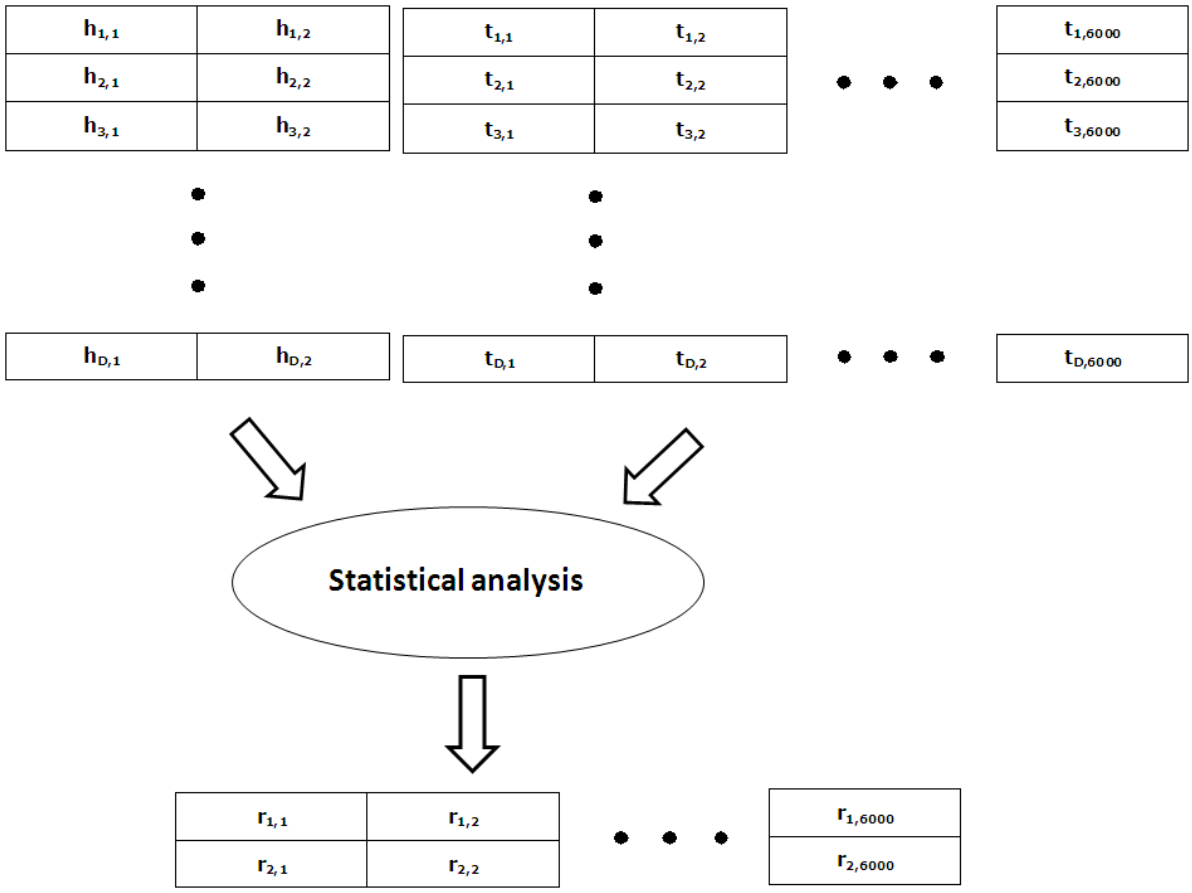

Figure 5.8: The procedure of analysis.

Three different statistical analyses have been applied:

- Correlation Coefficient analysis
- Difference of Means analysis

- Distance of Means analysis

All of them will be treated in detail in the following section.

5.4 Practical attacks

In the beginning we consider the Hamming-Distance model as the default power model and the Difference of Means analysis as the default statistical analysis. We present two examples of the attacks here to show how the power traces are measured and how we decide the intermediate values.

5.4.1 Comparison: power traces measured from V_{DD} vs. *GND*

As described in the first step we measure two different sets of power traces at V_{DD} and *GND*, here we perform the attacks based on these two kinds of power traces separately and compare their results. For both cases we use the Hamming-Distance model and the Difference of Means analysis.

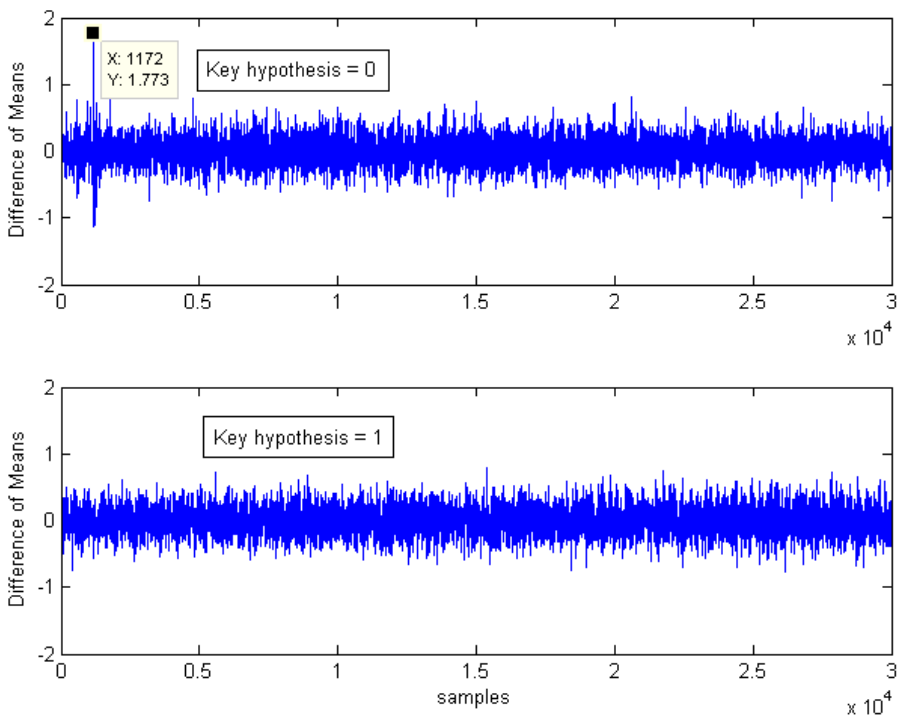

Figure 5.9: Attack using the power traces measured at *GND*.

Figure 5.9 shows the attack result based on 4000 power traces which are measured at *GND*. The upper subfigure corresponds to the case of guessing the first bit of key as 0 and the lower

subfigure corresponds to the case of guessing 1. The rule is to look for the peak value from both figures, if it occurs in the case of guessing 0 then the first bit of secret key is 0, if it occurs in the other case then the secret key bit is 1. In our case the peak value happens in the case of guessing 0 so we know the first bit of the secret key is 0.

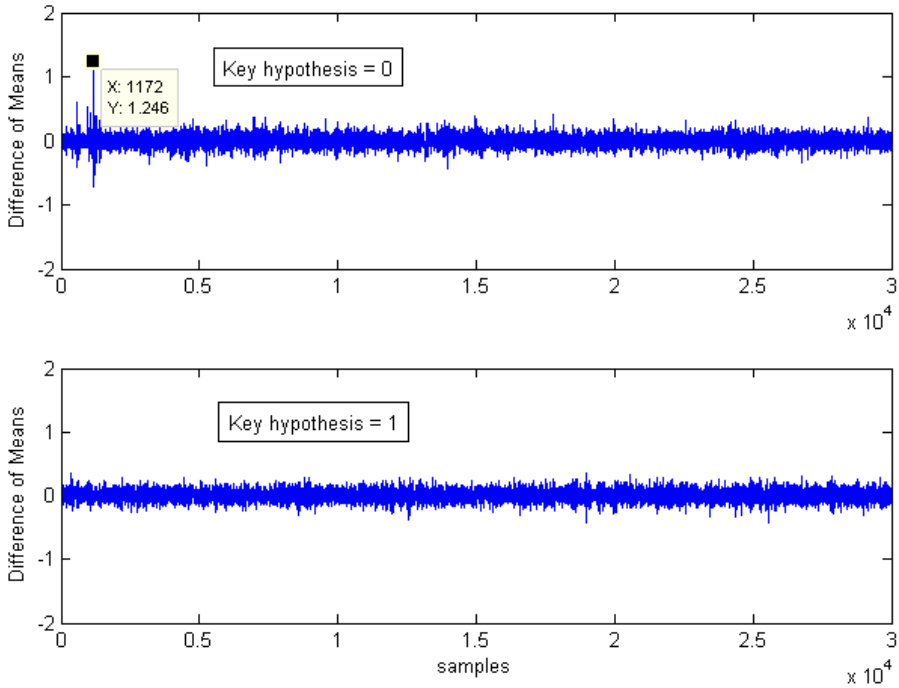

Figure 5.10: Attack using the power traces measured at V_{DD}.

In the other way around figure 5.10 shows the attack result based on the 4000 power traces measured at V_{DD}, and all the other conditions are remained. Observing the attack result we can get the same conclusion that the first bit of the key is 0.

Comparing this two figures, both of them can reveal the entire secret key with 4000 power traces by repeating the procedure for each single bit. The difference is that there is more noise in the case of using the power traces measured at GND than V_{DD}, but the peak value is higher. So the compensation between these two aspects make these two kinds of power traces having the same effect on the attack result, which means under the same condition, these two sets of power traces will not bring any significant difference into the attack. For our case we choose the power traces which are measured at V_{DD} for further attack.

5.4.2 Comparison: attacks based on S register alone vs. S and R registers

As indicated in step 2 we can choose R and S registers together as the intermediate values or only S register. In order to figure out which one has better performance we perform two attacks for

both cases based on 2000 power traces, where the results are shown in figure 5.11 and figure 5.12.

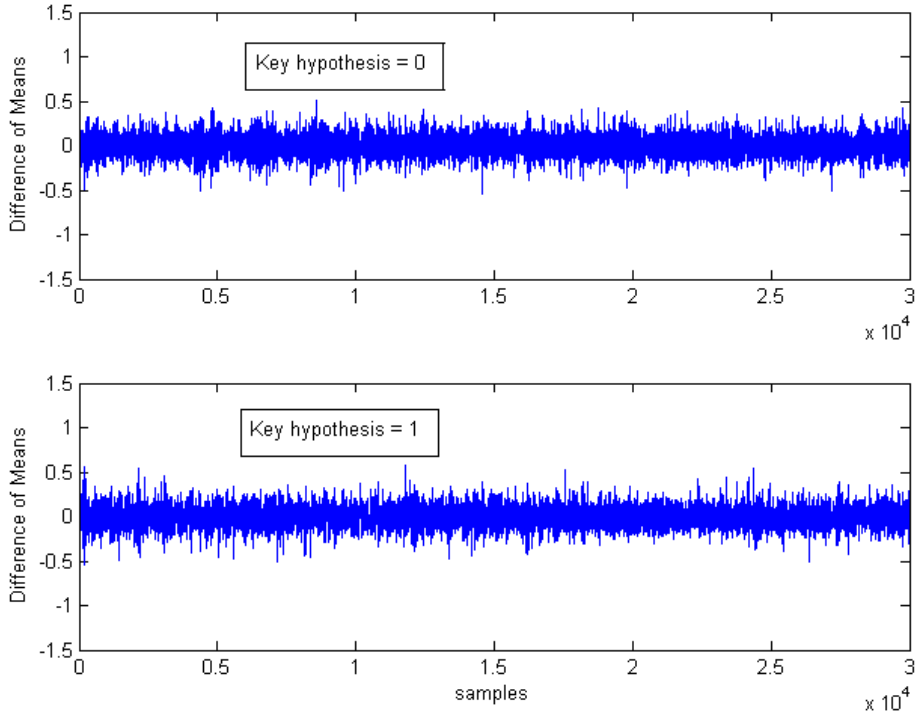

Figure 5.11: Attack based on S register only. Key hypotheses at the 21st key-bit.

Figure 5.11 shows the result of the attack based on S register only. We can observe that it is hard to recognize the highest peak in this case, which means the secret key can not be revealed by this approach based on 2000 power traces.

Figure 5.12 shows the result of the attack based on R and S registers. This time we use the same amount of power traces, but the result shows a clear peak when the key hypothesis is 0. We consider both of them; if we only use S register as the intermediate values then we need more than 2000 power traces to reveal the secret key. On the other hand, if we use both R and S registers then 2000 power traces are already enough for the attack. In other words, the combination of the R and S registers is more related to the power consumption than using S register alone. Of course, this conclusion is valid as long as the same condition is remained for both cases.

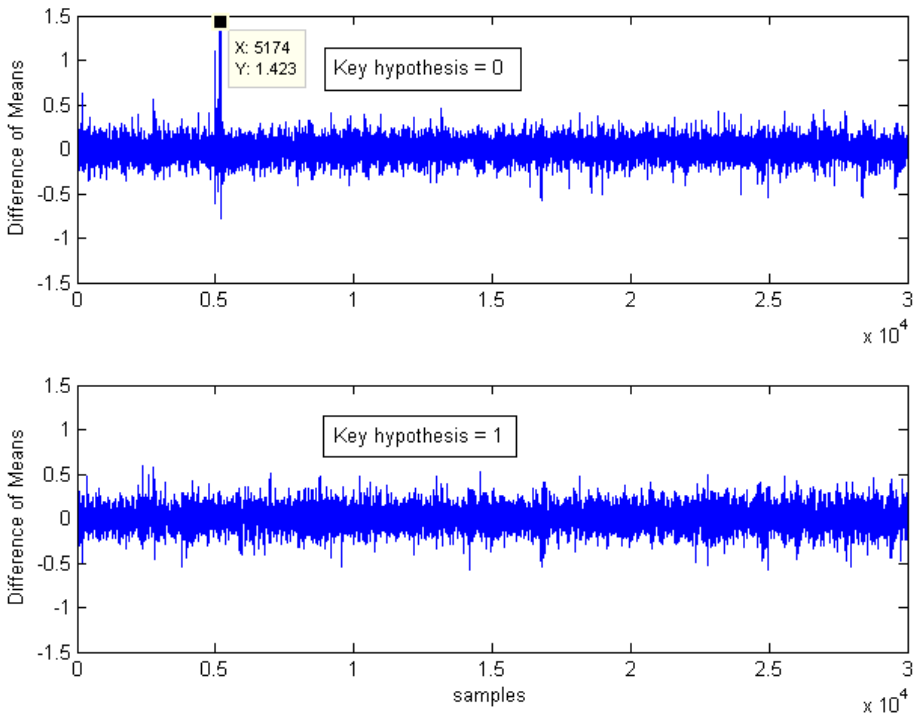

Figure 5.12: Attack based on R and S registers. Key hypotheses at the 21st key-bit.

So far we have chosen the proper intermediate values and power traces, the following attacks are performed based on the R and S registers and the power traces which are measured at V_{DD}. The goal of the following attacks is to find out which the best power model is, which the best statistical analysis method is or which the best combination of power model and statistical analysis is in our perspective.

5.4.3 Correlation coefficient analysis with Hamming-Distance model

First, the Hamming-Distance model is used as the power model to transform the hypothetical intermediate matrix $\mathbf{V}$ to the hypothetical power consumption matrix $\mathbf{H}$. The HD model describes the power consumption by counting the transitions that occur at a specific time during the processing. In practice, we build two matrices $\mathbf{V}_{pre0}$ and $\mathbf{V}_{current0}$ for the key hypothesis 0 and $\mathbf{V}_{pre1}$ and $\mathbf{V}_{current1}$ for the key hypothesis 1 as well in the way as mentioned before, one is for the previous run and the other is for the current run. Later, we compute the XOR between the two matrices elementwise to count how many transitions happen, and then store the result in the corresponding position of matrix $\mathbf{H}$.

Afterwards, correlation coefficient analysis is applied on matrices $\mathbf{H}$ and $\mathbf{T}$ to get the result matrix $\mathbf{R}$. In other words, we calculate the correlation between all the columns of $\mathbf{H}$ and all the columns of the recorded power consumption values in $\mathbf{T}$, the result is then stored in matrix $\mathbf{R}$.

At the end we verify the key guess by searching the highest correlation coefficient in matrix **R**, this leads to a successful key hypothesis. And the entire process is interpreted by Algorithm 1 in the appendix.

Figure 5.13 shows the plots for the key hypotheses 0 and 1 of our attack based on 1000 measurements. According to the sampling rate of the digital oscilloscope and the operating frequency of the ASIC during the measurements which is 1 MHz, we can conclude that the interval between two successive key loading operations is around 200 samples. Hence, we can reduce the size of the applied matrix **T** which contains the measured power traces, from $D \times 60000$ to $D \times 200$ when doing the attack on one bit at a time. This implies, during an attack on one specific key bit, only 200 sample points in a power trace are needed, and when attacking the next key bit, the entire matrix **T** is shifted to the right by 200 points in time. Consequently, the size of matrix **R** is also reduced to $D \times 200$ for attack at each single key bit. By doing this, we can dramatically reduce the time needed to perform the correlation coefficient analysis.

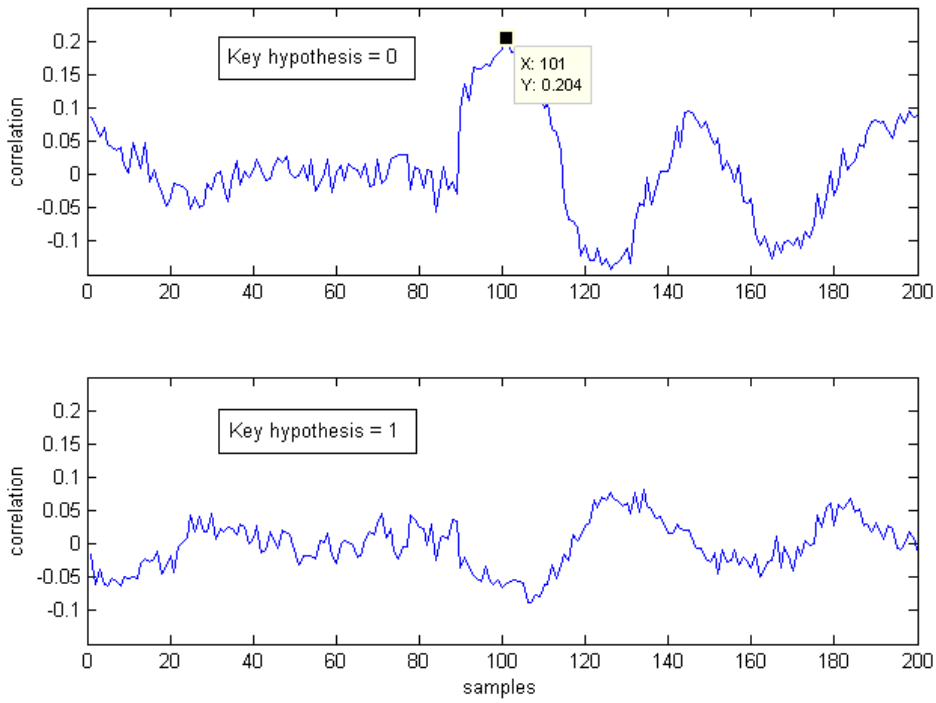

Figure 5.13: Key hypotheses at the 128th key-bit.

Observing figure 5.13, there shows a clear peak for the key guess 0 which means the actual value of this key bit is 0. As shown in figure 5.14, after approximately 1000 measurements the right key hypothesis can be distinguished from a wrong guess. Hence, we may say that about 1000 power traces are required for a successful attack.

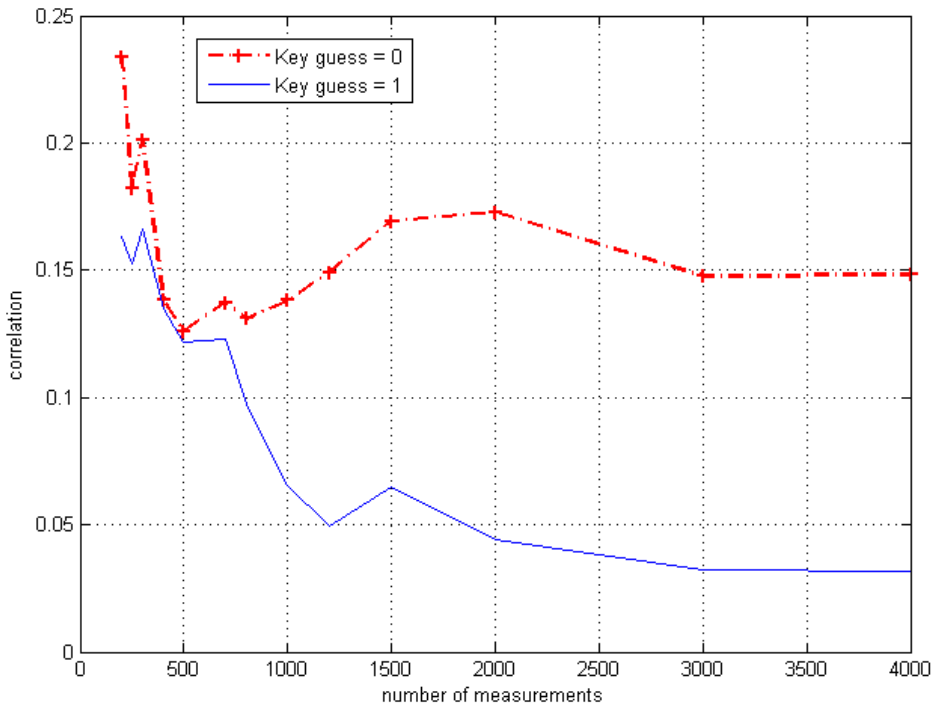

Figure 5.14: Correlation as a funtion of the number of power traces.

Improvement of correlation coefficient analysis

With the HD power model and correlation coefficient analysis, we are able to reveal the entire key by using 1000 power traces. However, as already discussed previously, based on the sampling rate of the oscilloscope and the frequency at which the cipher is running, we can calculate the interval in time between two successive loading of key bit. Subsequently, if we are able to identify the position along the time axis of the first bit that is revealed, for the rest of the key bits we can further reduce the range of searching for the highest correlation coefficient from 200 to some value around 10. One extreme would be looking at exactly the assumed key loading point in time, which is the 101st column of matrix **R**. This is because that, when we create matrix **T**, we locate the assumed key loading point in the middle of the index of the matrix. This approach can reduce the number of power traces required to reveal the key. Only 700 measurements are needed for a successful attack as shown in figure 5.15. And figure 5.16 shows one example of attacks using 700 power traces.

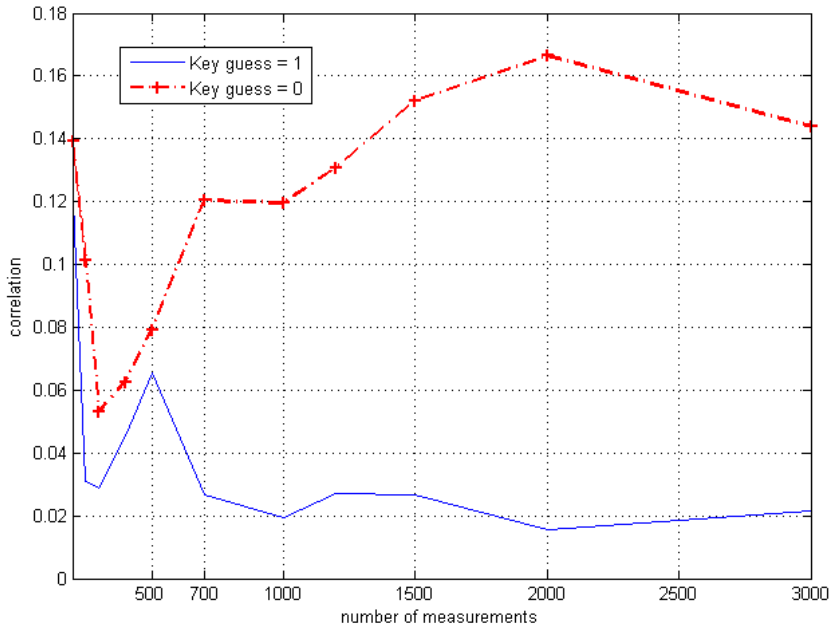

Figure 5.15: Correlation as a funtion of the number of power traces (only looking at the 101st column in **R**).

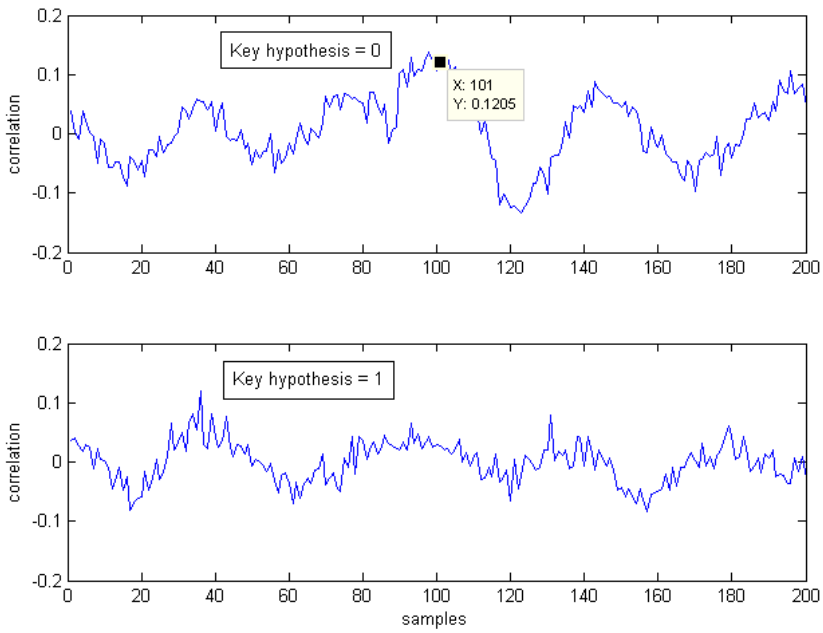

Figure 5.16: Key hypotheses at the 1st key-bit (only looking at the 101st column in **R**).

5.4.4 Correlation coefficient analysis with Hamming-Weight model

In this attack we use the Hamming-Weight model instead of the Hamming-Distance model. The difference is the number of bits that are set in the processed data value is required to simulate the power consumption. So only the matrix $\mathbf{V}_{current}$ for the current run is needed. Each element in $\mathbf{H}$ is the result of counting the number of bits that are 1s in both R and S registers. And then the correlation coefficient analysis is again applied to the calculation between matrices $\mathbf{H}$ and $\mathbf{T}$. Algorithm 2 of this method is shown in the appendix.

Figure 5.17 shows the plots for the key hypotheses 0 and 1 of the attack based on 4000 measurements.

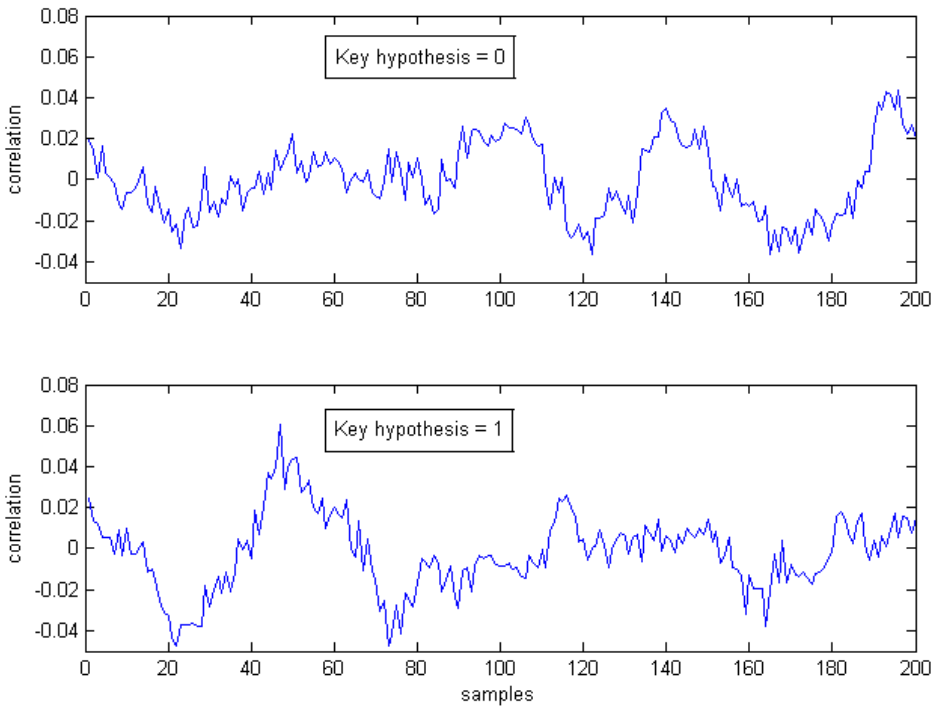

Figure 5.17: Key hypotheses at the 3rd key-bit.

Observing figure 5.17, there is no clear peak for both key guesses 0 and 1 which means 4000 measurements are not sufficient to reveal the key, while 1000 measurements are already enough for the case of HD model. The reason is the way these two power models describe the power consumption is different. In other words, the HW model dose not describe the power consumption as good as the HD model. As shown in figure 5.18, althrough after approximately 3000 measurements the right key hypothesis can be distinguished from a wrong guess, however, this is not the case for the other key bits. In our analysis, we are not able to reveal the entire key bits within 4000 power traces, only about 70% of the whole key bits are extracted correctly. Hence, we may say that more power traces are required for a successful attack.

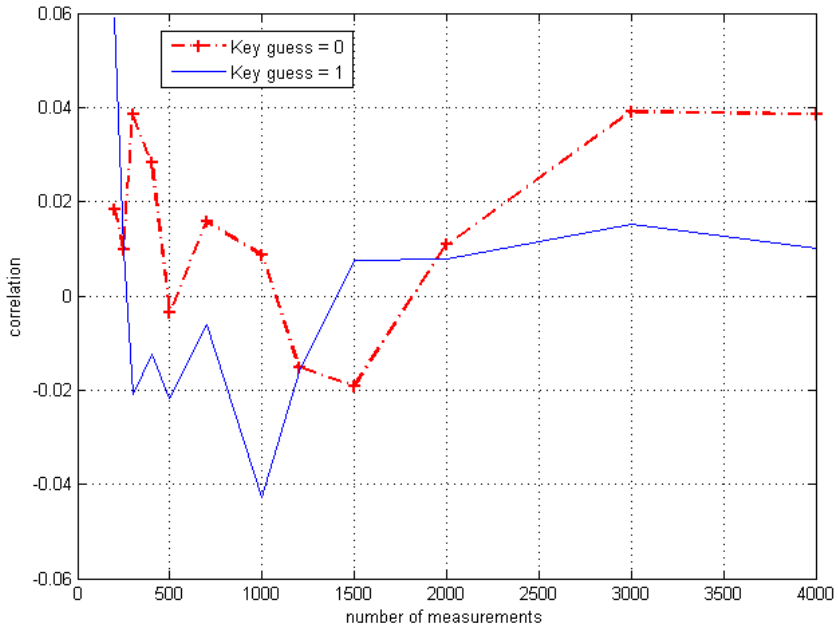

Figure 5.18: Correlation as a funtion of the number of power traces.

5.4.5 Correlation coefficient analysis with Zero-Value model

The Zero-Value model assumes that the power consumption for the data value 0 is lower than the power consumption for all other values. For example, if there is a multiply operation occurs in the algorithm, the power consumption for 0×123 must be significantly lower than the case of 123×123. So it is a good power consumption description in some cases, but not in our case. In the MICKEY algorithm, there are only EXOR and multiply operations, for the data values they are always one or zero. This implies the power consumption of the EXOR or multiply operations for data value 1 and 0 will be more or less the same. Therefore, Zero-Value model is not suitable for our attack.

5.4.6 Difference of Means analysis with Hamming-Distance model

In this attack, the HD model is used again but the statistical analysis model is changed to the Difference of Means analysis. Difference of Means analysis is another method to determine the relationship between the columns of **H** and **T**. The principle is first to split all the power traces into two sets according to matrix **H**. The mean value M_1 and M_0 are calculated for each set afterwards. Again, at the end we search for the peak from the difference between M_1 and M_0 to verify the key guess. In practice, by using the HD model, each element in **H** stores the number of transitions occurring. A maximum of 320 transitions may happen which means all the states of both R and S registers are flipped, so we choose 160 as the edge according to statistics. If the element in **H** is greater than 160, then the corresponding trace will be put into set one. On the other hand, if it is smaller than 160 the corresponding traces will be put into set zero.

The following work is the same as described before, first calculate the difference of mean value and then search for the peak value. The whole process is interpreted by Algorithm 3 in the appendix.

Figure 5.19 shows the plots for the key hypotheses 0 and 1 of the attack based on 2000 measurements.

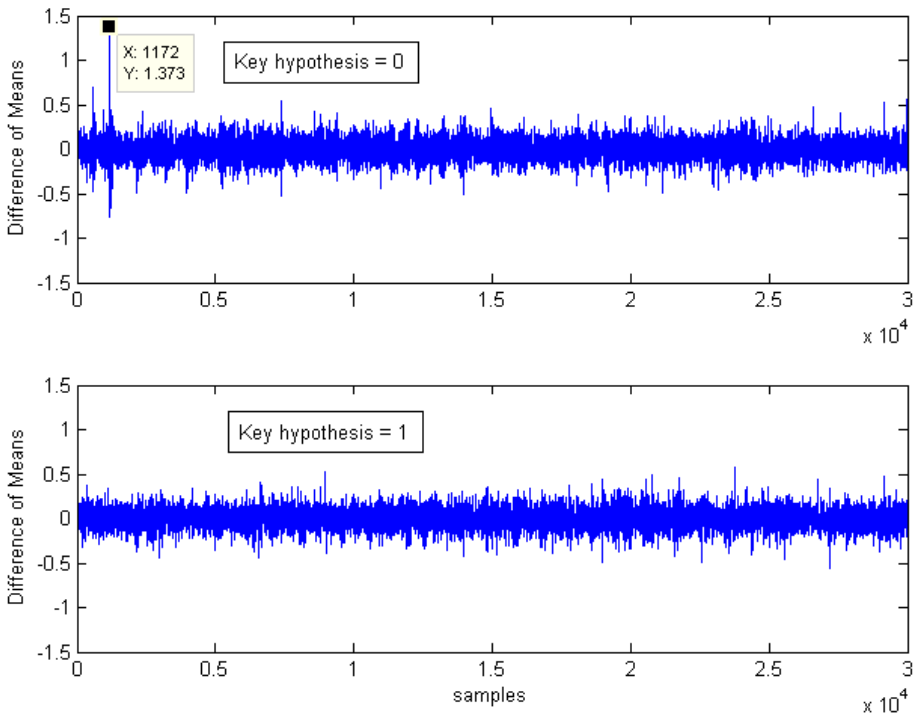

Figure 5.19: Key hypotheses at the 1st key-bit.

Observing figure 5.19, there shows a clear peak in the upper subplot which tells the real value of the first key bit is 0. As known the correct value of the first bit is 0, we perform several attacks based on different numbers of power traces to find out at least how many power traces are needed to reveal the secret key. In figure 5.20, the dash-dot line indicates the values at the first bit position for guessing 0 when different amount of power traces are used. The solid line indicates the values at the first bit position for guessing 1. For example, we use 1500 power traces, when we assume the key bit equals 0 the corresponding difference of means value is 1.3, when we assume the key bit is 1 the difference of means value is 0.7. Obviously there is a difference between these two differnece of means values, which enable us to reveal this key bit. But 1500 power traces are not sufficient to reveal the entire secret key, because figure 5.20 only tells us that the attack can discover the 1st key bit, which does not mean it will be always successful for the rest of the key bits. After large amount of attacks, we notice 2000 power traces are the least requirement for revealing the entire secret key.

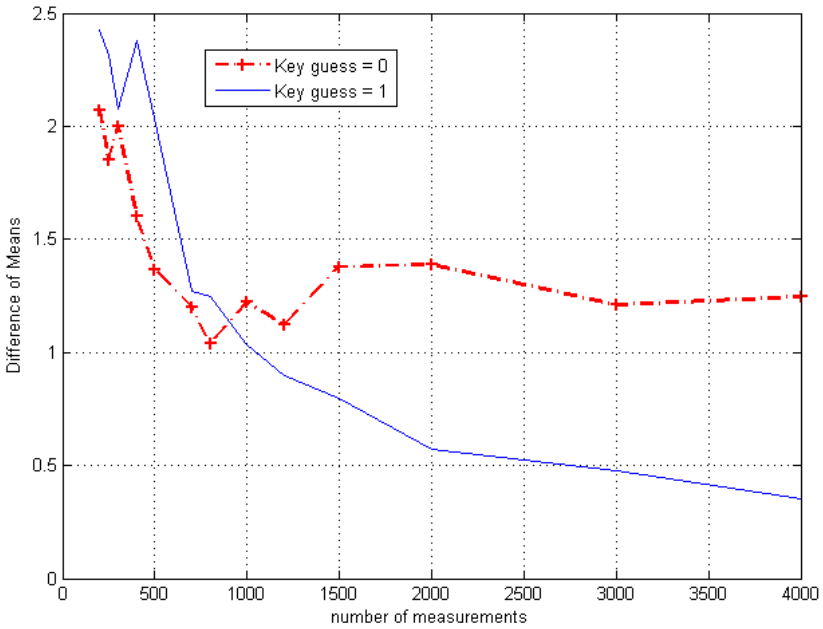

Figure 5.20: Difference of means as a funtion of the number of power traces.

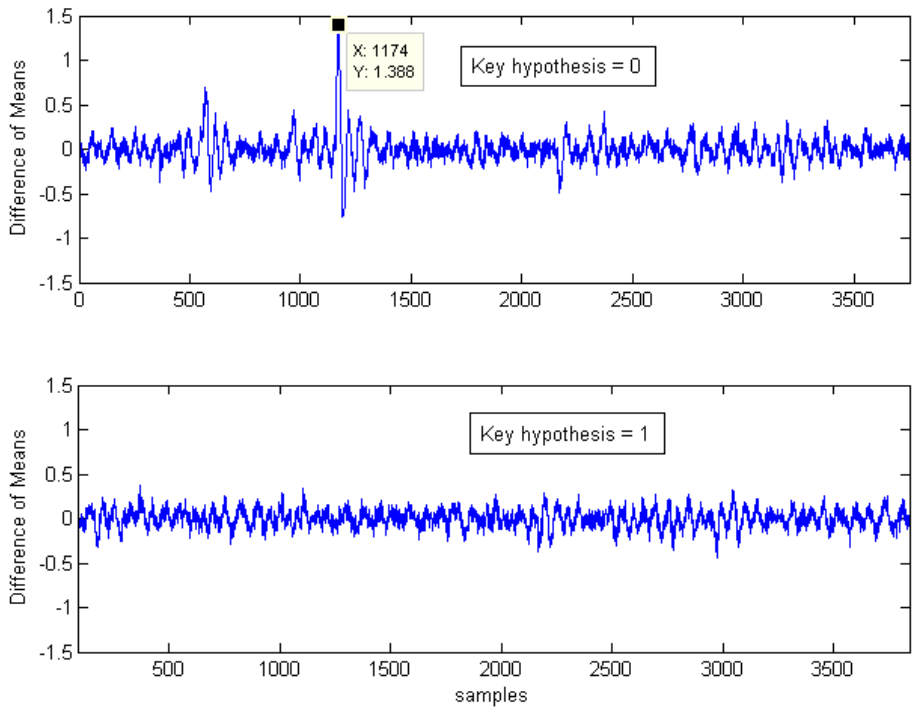

Figure 5.21: Around the attack target point (zoomed version of figure 5.19).

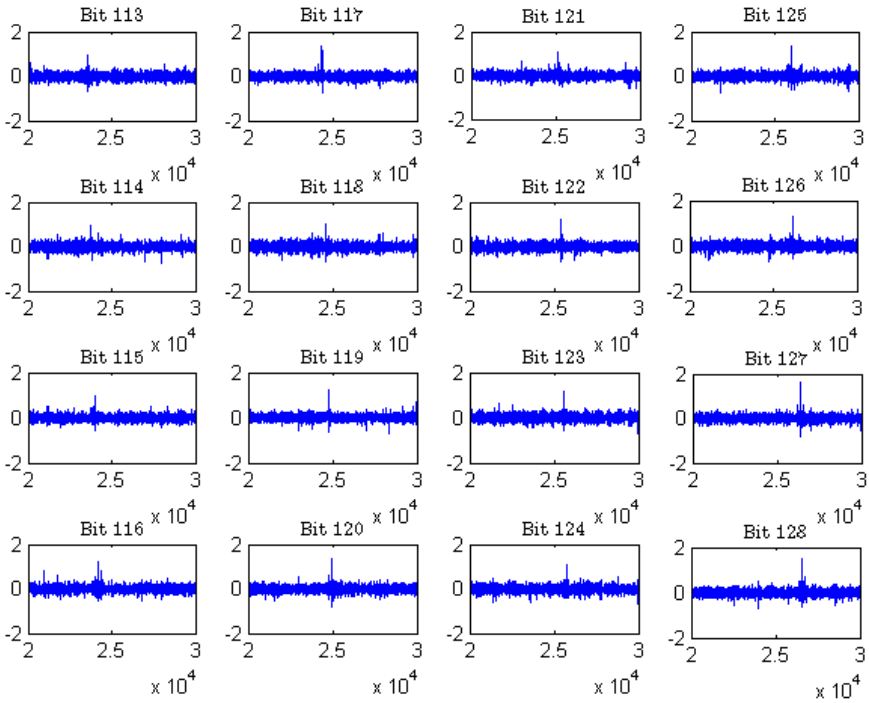

Figure 5.22: Plots for the correct key hypotheses in the attack on the last 16 key bits. The x-axes show the samples and the y-axes show the difference of means.

In figure 5.22 it shows an example of correct key guesses on the last 16 key bits in the Difference of Means analysis attack using 2000 measurements.

5.4.7 Difference of Means analysis with Hamming-Weight model

In this attack we use the Difference of Means analysis and the Hamming-Weight model. Both of these analysis method and model have been introduced before, so we ignore the details of process here. We only focus on the result.

Figure 5.23 shows the plots for the key hypotheses 0 and 1 of the attack based on 4000 measurements.

It is shown in the figure that no significant peak occurs for both key hypotheses at the second key bit position, thus 4000 power trace measurements are not sufficient to distinguish the correct key hypothesis from a wrong guess. We may therefore say that more power traces are needed for a successful attack.

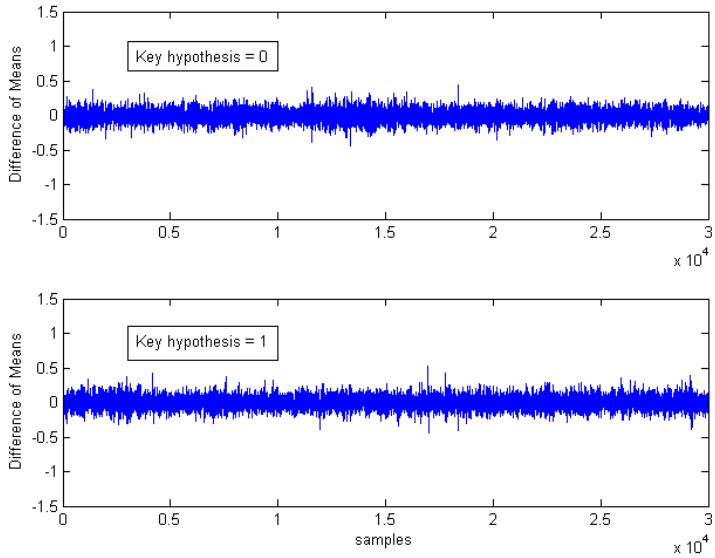

Figure 5.23: Key hypotheses at the 2nd key-bit.

5.4.8 Distance of Means analysis with Hamming-Distance model

In this approach, Distance of Means analysis is used as the statistical analysis method to determine the relationship between H and T. As we explained in the previous chapter, it is an improvement of Difference of Means analysis method. The preceding part of the process is the same as Difference of Means. But after calculating the difference between the mean value M1 and M0, we first divide this difference by the standard deviation as clearly described in equation 4.2 and 4.3, and then search for the peak value again. Algorithm 4 of this method is shown in the appendix.

It is shown in figure 5.24 that the correct key hypothesis can be distinguished from a wrong guess with 2000 measurements. The real value of this first key bit is 0 because the significant peak occurs in the upper subplot. Observing the figure 5.25, the dash-dot line starts to separate from the solid line after 1500 power traces being used, which means theoretically this bit can be recovered by using 1500 power traces. However, this may not be the case for the rest of the key bits. Based on a large amount of measurements, we conclude that about 2000 power traces are the least requirement for a successful attack.

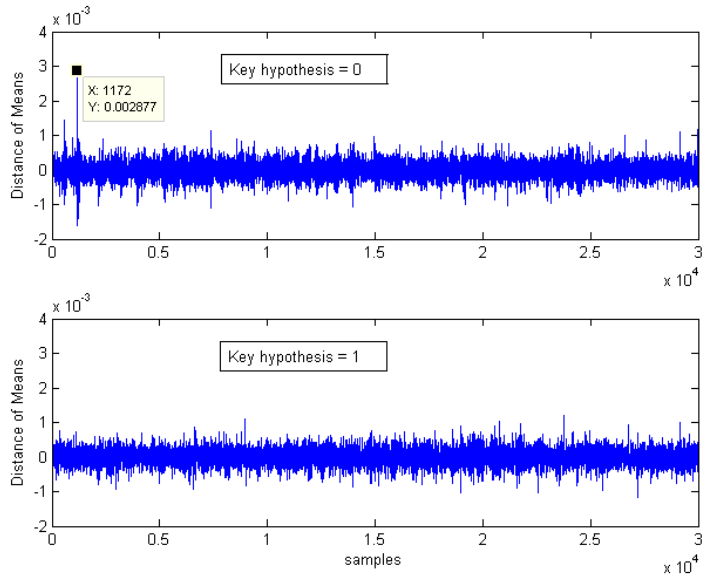

Figure 5.24: Key hypotheses at the 1st key-bit.

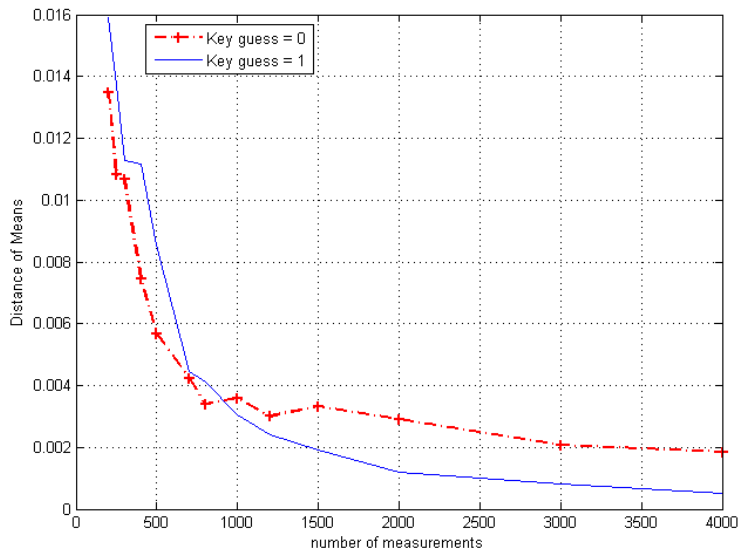

Figure 5.25: Distance of means as a funtion of the number of power traces.

5.4.9 Distance of Means analysis with Hamming-Weight model

We ignore the details of the analysis and modeling processes and only focus on the result. Figure 5.26 shows the plots for the key hypotheses 0 and 1 of our attack based on 4000 measurements.

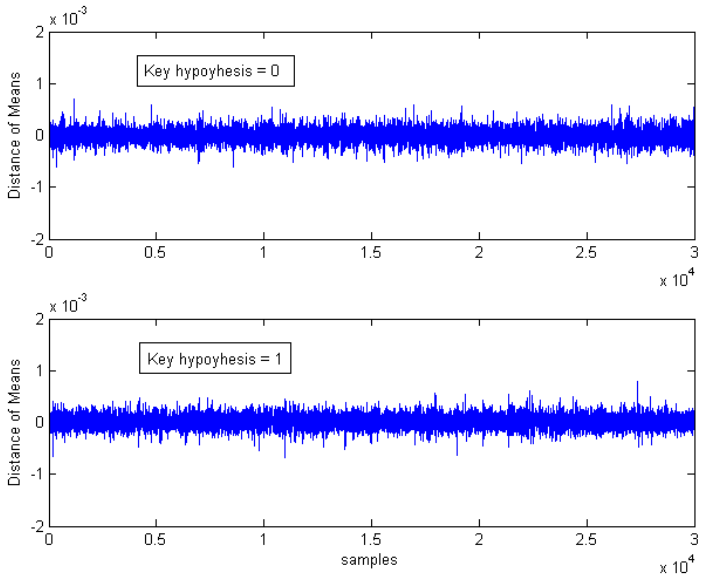

Figure 5.26: Key hypotheses at the 4th key-bit.

Obviously, there is no clear peak to distinguish the two key hypotheses. Especially, at the position of the 4th key bit there is no significant difference between the two subplots to reveal this secret key bit. Therefore, we can conclude that for a successful attack, more power traces are required.

5.5 Summary

In this chapter we have introduced all the attacks we performed. At the beginning we tried different power traces and different intermediate values. According to the comparison of them, we decided to use the power traces which are measured at V_{DD} of the ASIC, and chose the states of R and S registers together as the intermediate values. Afterwards, based on these two decisions we utilized different statistical analyses and power models to attack the stream cipher, where the results are shown in table 5.1.

	Correlation Coefficient	Difference of Means	Distance of Means
Hamming-Distance	1000	2000	2000
Hamming-Weight	more than 4000	more than 4000	more than 4000

Table 5.1: Number of measurements needed to find the correct key for different DPA techniques.

As it can be seen in the table, the Correlation Coefficient is the best statistical analysis method compared to the Difference of Means and Distance of Means analyses. Theoretically, the Correlation Coefficient and Distance of Means analyses consider the difference and variance

of the values together, so the attacks based on those analyses should require about the same number of power traces, meanwhile require fewer power traces than the attacks based on the Difference of Means. The reason is that the Difference of Means only takes the difference of values into account. In our case the Correlation Coefficient minimally requires 1000 power traces for a successful attack while the Difference of Means and Distance of Means require 2000 traces at least. Most of them work ideally except the Distance of Means analysis requires more traces than what we expect.

Among different power models, actually we only succeeded with the Hamming-Distance model. We tried Hamming-Weight model with different statistical analyses but all of them require more than 4000 power traces. We can conclude the Hamming-Distance model describes the power consumption much better than the case of Hamming-Weight model. We also implemented the Zero-Value model in our attack, but we think it is not suitable for attacking MICKEY due to the reason which is explained at the very beginning of this chapter.

Chapter 6

Conclusions and future work

6.1 Conclusions

In this thesis two main objectives have been addressed. In one hand we developed a measurement setup between a PC and the eSCARGOt ASIC. On the other hand we performed the DPA attacks on the MICKEY stream cipher which is implemented on the ASIC.

The measurement setup is used to measure the power consumption of the ASIC for encryptions of MICKEY. The PC controls the whole measurement by generating the Initial Vectors, KEY and plaintext and collecting the power traces at the end. The ASIC is a hardware implementation of the stream ciphers. We developed an interface between them on an FPGA in order to provide the communication between the PC and the ASIC. In other words, our job is to control the ASIC to run the target stream cipher on as much different data blocks as we need, in the meanwhile measure and store the corresponding power traces for the further attacks. So far, our interface is applicable for three of the stream ciphers which are integrated in the ASIC, which are MICKEY, Trivium 128 and Gain 80.

Among these three stream ciphers, our attacks focus on MICKEY. Based on the measured power traces, we performed different DPA attacks to look for the most efficient way to reveal the entire secret key. We tried different intermediate values, different power traces from *GND* as well as V_{DD}, different power models and different statistical analyses. The power models and statistical analyses influence the attack result significantly. After comparing different attack results, we can conclude that, the attack based on the Hamming-Distance power model and the Correlation Coefficient statistical analysis works most efficiently. It requires the least number of power traces, which is 1000, to reveal the entire secret key successfully. Other options always require more than 1000 power traces or are unsuitable for the attacks on the MICKEY stream cipher.

6.2 Future work

We make several suggestions for the future work:

1. The measurement setup in our thesis is only applicable for three stream ciphers. More

effort should be put into the interface to make the selection of the remaining stream ciphers possible and also perform side-channel analysis on the other stream ciphers.

2. We use internal registers as the intermediate values during our attack on MICKEY, so try to find other intermediate values which are strongly related to the power consumption is another option.

3. We perform the attack on the ASIC bit by bit. Maybe attack one byte of the key each time is also possible.

4. We suggest applying other power models and statistical analyses, such as Maximum-Likelihood analysis, etc.

Bibliography

[1] *eSCARGOt full datasheet v1.3*, Department of Electronic and Electrical Engineering, University of Sheffield, 2008. [Online]. Available: http://www.shef.ac.uk/eee/escargot/

[2] F. K. Grkaynak, P. Luethi, N. Bernold, R. Blattmann, V. Goode, M. Marghitola, H. Kaeslin, N. Felber, and W. Fichtner, "Hardware evaluation of eSTREAM candidates: Achterbahn, Grain, MICKEY, MOSQUITO, SFINKS, Trivium, VEST, ZK-Crypt," ETH Zurich, CH-8092 Zurich, Tech. Rep.

[3] E. D. Mulder, S. B. Ors, B. Preneel, and I. Verbauwhede, "Differential power and electromagnetic attacks on a FPGA implementation of elliptic curve cryptosystems," *Computers & Electrical Engineering*, vol. 33, no. 5-6, pp. 367–382, 2007.

[4] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography.* Boca Raton, FL, USA: CRC Press, Inc., 1996.

[5] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[6] Keylength - cryptographic key length recommendation. [Online]. Available: http://www.keylength.com/en/8/

[7] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology.* London, UK: Springer-Verlag, 1999, pp. 388–397.

[8] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *CHES '01: Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2001, pp. 251–261.

[9] J.-J. Quisquater and D. Samyde, "Electromagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards," in *E-SMART '01: Proceedings of the International Conference on Research in Smart Cards.* London, UK: Springer-Verlag, 2001, pp. 200–210.

[10] P. C. Kocher, "Timing Attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *CRYPTO*, 1996, pp. 104–113.

[11] S. B. Örs, F. Gürkaynak, E. Oswald, and B. Preneel, "Power-Analysis Attack on an ASIC AES implementation," in *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2.* Washington, DC, USA: IEEE Computer Society, 2004, p. 546.

[12] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs: State-of-the-art implementations and attacks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 534–574, 2004.

[13] C. H. Gebotys, "Security-driven exploration of cryptography in DSP cores," in *ISSS '02: Proceedings of the 15th international symposium on System Synthesis.* New York, NY, USA: ACM, 2002, pp. 80–85.

[14] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture.* New York, NY, USA: ACM, 2007, pp. 494–505.

[15] J. Daemen and V. Rijmen, *The Design of Rijndael.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

[16] Ecrypt stream cipher project. [Online]. Available: http://www.ecrypt.eu.org/stream/

[17] ECRYPT II - European Network of Excellence for Cryptology II. [Online]. Available: http://www.ecrypt.eu.org/

[18] W. Fischer, B. M. Gammel, O. Kniffler, and J. Velten, "Differential power analysis of stream ciphers." in *CT-RSA*, ser. Lecture Notes in Computer Science, M. Abe, Ed., vol. 4377. Springer, 2007, pp. 257–270. [Online]. Available: http://dblp.uni-trier.de/db/conf/ctrsa/ctrsa2007.html#FischerGKV07

[19] C. De Cannière and B. Preneel, "Trivium," *New Stream Cipher Designs: The eSTREAM Finalists*, pp. 244–266, 2008.

[20] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86–93, 2007.

[21] European Stream Ciphers Are Ready to GO. [Online]. Available: http://www.shef.ac.uk/eee/escargot/

[22] *Xilinx University Program Virtex-II Pro Development System Hardware Reference Manual*, XILINX, 2008. [Online]. Available: http://www.xilinx.com/univ/xupv2p.html

[23] *Processor Local Bus (PLB) v3.4 (v1.02a)*, Xilinx, 2009. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/plb_v34.pdf

[24] P. Schaumont. Gezel version 2. [Online]. Available: http://rijndael.ece.vt.edu/gezel2/index.php/Main_Page

[25] M. D. Steve Babbage, Ed., *The stream cipher MICKEY-128 2.0.* Vodafone Group R&D, Newbury, UK, August 2006. [Online]. Available: http://www.ecrypt.eu.org/stream/mickey128p2.html

[26] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2003, pp. 2–12.

[27] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *CHES*, 2004, pp. 16–29.

[28] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks Revealing the Secrets of Smart Cards.* Springer, 2007.

[29] D. Agrawal, J. R. Rao, and P. Rohatgi, "Multi-channel attacks," in *CHES*, 2003, pp. 2–16.

[30] J.-J. Quisquater and D. Samyde, Eds., *A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions: the SEMA and DEMA methods.* Eurocrypt rump session, 2000.

[31] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM Side-Channel(s)," in *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2003, pp. 29–45.

[32] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *WOST'99: Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology.* Berkeley, CA, USA: USENIX Association, 1999, pp. 17–17.

Appendix A

Algorithms for different DPA attack techniques

Algorithm 1 Pseudocode for DPA with the Hamming-Distance power model and the Correlation Coefficient analysis.

```
for r = 1 to 128 do
  for i = 1 to D do
    h_{i,0} = HD(V_{pre0}, V_{current0})
    h_{i,1} = HD(V_{pre1}, V_{current1})
  end for
  calculate the correlation:
  for t = 1 to T (length of the power trace) do
    r_{0,t} = C(h_{*,0}, t_{*,t})
    r_{1,t} = C(h_{*,1}, t_{*,t})
  end for
  Accept the key hypothesis, for which the correlation coefficient is maximal.
end for
```

Algorithm 2 Pseudocode for DPA with the Hamming-Weight power model and the Correlation Coefficient analysis.

```
for r = 1 to 128 do
  for i = 1 to D do
    h_{i,0} = HW(V_{current0})
    h_{i,1} = HW(V_{current1})
  end for
  calculate the correlation:
  for t = 1 to T (length of the power trace) do
    r_{0,t} = C(h_{*,0}, t_{*,t})
    r_{1,t} = C(h_{*,1}, t_{*,t})
  end for
  Accept the key hypothesis, for which the correlation coefficient is maximal.
end for
```

Algorithm 3 Pseudocode for DPA with the Hamming-Distance power model and the Difference of Means analysis.

```
for r = 1 to 128 do
  for i = 1 to D do
    h_{i,0} = HD(V_{pre0}, V_{current0})
    h_{i,1} = HD(V_{pre1}, V_{current1})
    if h_{i,0} or h_{i,1} ≥ 160 then
      put the corresponding power trace into set A_1, record the number of power traces c_1
      in this set.
    else
      put the corresponding power trace into set A_0, record the number of power traces c_0
      in this set.
    end if
  end for
  calculate the mean m1 = A_1/c_1 and m0 = A_0/c_0 for each key hypothesis.
  calculate the difference m1 - m0 for each key hypothesis as well.
  Accept the key hypothesis, for which there shows a spike in the difference.
end for
```

Algorithm 4 Pseudocode for DPA with the Hamming-Distance power model and the Distance of Means analysis.

```
for r = 1 to 128 do
  for i = 1 to D do
    h_{i,0} = HD(V_{pre0}, V_{current0})
    h_{i,1} = HD(V_{pre1}, V_{current1})
    if h_{i,0} or h_{i,1} ≥ 160 then
      put the corresponding power trace into set A_1, record the number of power traces c_1
      in this set.
    else
      put the corresponding power trace into set A_0, record the number of power traces c_0
      in this set.
    end if
  end for
  calculate the mean m1 = A_1/c_1 and m0 = A_0/c_0 for each key hypothesis.
  calculate the difference m1 - m0 for each key hypothesis as well, then further divide the
  difference by the standard deviation s_{m1,m0}.
  Accept the key hypothesis, for which there shows a spike in the result of division.
end for
```

Appendix B

The GEZEL implementation of the interface

```
dp interface_n ( in ready_for_IV : ns(1);
in ready_for_KEY : ns(1);
in output_valid : ns(1);
in dout : ns(1);
out cipher_0 : ns(1);
out cipher_1 : ns(1);
out cipher_2 : ns(1);
out cipher_3 : ns(1);
out load : ns(1);
out din : ns(1);
out keyiv : ns(1);
out decrypt : ns(1);
out rstchip : ns(1);
out slew_control : ns(1);
out drive_strength : ns(1);
out trigger : ns(1))

{

reg rfi, rfk, ov : ns(1); // ready_for_iv, ready_for_key and output_valid
reg cipher0, cipher1, cipher2, cipher3 : ns(1); // registers to select a specific cipher
reg key, iv : ns(128);
reg output : ns(9); // register to store ciphertext
reg input : ns(8); // register to store plaintext
reg flag : ns(1); // register to indicate the start of a new encryption session
reg counter_input : ns(4);
reg counter_keyiv : ns(8);

sfg init {
rfi = ready_for_IV;
rfk = ready_for_KEY;
```

```
ov = output_valid;
cipher_0 = 0;
cipher_1 = 0;
cipher_2 = 0;
cipher_3 = 0;
load = 0;
din = 0;
keyiv = 0;
decrypt = 0;
drive_strength = 0;
slew_control = 0;
rstchip = 0;
trigger = 0;
}

sfg cipher_select { // selecting a specific cipher
cipher_0 = cipher0;
cipher_1 = cipher1;
cipher_2 = cipher2;
cipher_3 = cipher3;
load = 0;
din = 0;
keyiv = 0;
decrypt = 0;
drive_strength = 0;
slew_control = 0;
rstchip = 0;
trigger = 0;
}

sfg reset { // resetting the selected cipher
rstchip = 1;
load = 0;
din = 0;
keyiv = 0;
decrypt = 0;
drive_strength = 0;
slew_control = 0;
cipher_0 = cipher0;
cipher_1 = cipher1;
cipher_2 = cipher2;
cipher_3 = cipher3;
trigger = 0;
}

sfg update { // register update
rfi = ready_for_IV;
rfk = ready_for_KEY;
```

```
ov = output_valid;
counter_input = 0;
load = 0;
din = 0;
keyiv = 0;
decrypt = 0;
drive_strength = 0;
slew_control = 0;
cipher_0 = cipher0;
cipher_1 = cipher1;
cipher_2 = cipher2;
cipher_3 = cipher3;
rstchip = 0;
trigger = 0;
counter_keyiv = 0;
}
sfg key_load { // loading key
rfi = ready_for_IV;
rfk = ready_for_KEY;
ov = output_valid;
keyiv = key[0];
key = key >> 1;
load = 1;
din = 0;
decrypt = 0;
drive_strength = 0;
slew_control = 0;
cipher_0 = cipher0;
cipher_1 = cipher1;
cipher_2 = cipher2;
cipher_3 = cipher3;
rstchip = 0;
trigger = 1;
counter_keyiv = counter_keyiv + 1;
}
sfg iv_load { // loading IV
rfi = ready_for_IV;
rfk = ready_for_KEY;
ov = output_valid;
keyiv = iv[0];
iv = iv >> 1;
load = 1;
din = 0;
decrypt = 0;
drive_strength = 0;
slew_control = 0;
```

```
cipher_0 = cipher0;
cipher_1 = cipher1;
cipher_2 = cipher2;
cipher_3 = cipher3;
rstchip = 0;
trigger = 0;
counter_keyiv = counter_keyiv + 1;
}

sfg load_fetch { // reading ciphertext
output = (output | dout) << 1;
}

sfg input_text{ // loading plaintext
din = input[0];
input = input >> 1;
counter_input = counter_input + 1;
load = 1;
keyiv = 0;
decrypt = 0;
drive_strength = 0;
slew_control = 0;
cipher_0 = cipher0;
cipher_1 = cipher1;
cipher_2 = cipher2;
cipher_3 = cipher3;
rstchip = 0;
trigger = output[0];
}

sfg start {
rfi = ready_for_IV;
rfk = ready_for_KEY;
ov = output_valid;
cipher_0 = cipher0;
cipher_1 = cipher1;
cipher_2 = cipher2;
cipher_3 = cipher3;
load = 0;
din = 0;
keyiv = 0;
decrypt = 0;
drive_strength = 0;
slew_control = 0;
rstchip = 0;
trigger = 0;
}
}
```

```
fsm interface_n_ctlr(interface_n) {

initial s0;
state s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11;

@s0 if (flag == 1) then (start) -> s1;

else (init) -> s0;

@s1 (cipher_select) -> s2;

@s2 (reset) -> s3;

@s3 (reset) -> s4;

@s4 (reset) -> s5;

@s5 (reset) -> s6;

@s6 (reset) -> s7;

@s7 if (rfk == 1 | rfi == 1) then (cipher_select) -> s8;

else (update) -> s7;

@s8 if (cipher0 == 1 & cipher1 == 1 & cipher2 == 0 & cipher3 == 0 & rfk == 1 &
counter_keyiv < 80) then (key_load) -> s8; // Trivium

else if (cipher0 == 1 & cipher1 == 0 & cipher2 == 0 & cipher3 == 1 & rfk == 1 &
counter_keyiv < 128) then (key_load) -> s8; // Grain-128

else if (cipher0 == 1 & cipher1 == 1 & cipher2 == 0 & cipher3 == 1 & rfi == 1 & counter_keyiv
< 128) then (iv_load) -> s8; // MICKEY-128

else (update) -> s9;

@s9 if (cipher0 == 1 & cipher1 == 1 & cipher2 == 0 & cipher3 == 0 & rfi == 1 &
counter_keyiv < 80) then (iv_load) -> s9;

else if (cipher0 == 1 & cipher1 == 0 & cipher2 == 0 & cipher3 == 1 & rfi == 1 & counter_keyiv
< 96) then (iv_load) -> s9;

else if (cipher0 == 1 & cipher1 == 1 & cipher2 == 0 & cipher3 == 1 & rfk == 1 &
counter_keyiv < 128) then (key_load) -> s9;

else (cipher_select) -> s10;

@s10 if (ov == 0) then (update) -> s10;

else if (ov == 1 & counter_input < 8) then (input_text, load_fetch) -> s10; // loading plaintext

else (update, load_fetch) -> s11;

@s11 if (flag == 0) then (init) -> s0;

else (update) -> s11;
}
```