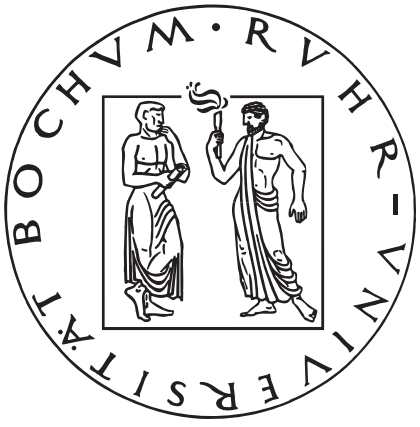
Side Channel Analysis Attacks on Stream Ciphers

Daehyun Strobel

23.03.2009

Masterarbeit
Ruhr-Universität Bochum



Lehrstuhl Embedded Security
Prof. Dr.-Ing. Christof Paar
Betreuer: Dipl.-Ing. Markus Kasper

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Bochum, 23.März 2009

Daehyun Strobel

Abstract

In this thesis, we present results from practical differential power analysis attacks on the stream ciphers Grain and Trivium.

While most published works on practical side channel analysis describe attacks on block ciphers, this work is among the first ones giving report on practical results of power analysis attacks on stream ciphers. Power analyses of stream ciphers require different methods than the ones used in todays most popular attacks. While for the majority of block ciphers it is sufficient to attack the first or last round only, to analyze a stream cipher typically the information leakages of many rounds have to be considered. Furthermore the analysis of hardware implementations of stream ciphers based on feedback shift registers inevitably leads to methods combining algebraic attacks with methods from the field of side channel analysis. Instead of a direct recovery of key bits, only terms composed of several key bits and bits from the initialization vector can be recovered. An attacker first has to identify a sufficient set of accessible terms to finally solve for the key bits.

On practical examples, we show how to successfully implement this kind of attacks for the recent stream ciphers Grain and Trivium. Therefore, we created a measurement setup that is ideally suited for acquiring power traces of the target device eSCARGOt, an ASIC including hardware implementations of both ciphers.

Contents

1 Introduction **1**
- 1.1 Previous Work . . . 2
- 1.2 Organization of this Thesis . . . 2

2 Statistical Methods **3**
- 2.1 Probability Space . . . 3
- 2.2 Discrete Random Variable . . . 3
- 2.3 Expected Value . . . 4
- 2.4 Variance and Standard Deviation . . . 4
- 2.5 Covariance and Correlation Coefficient . . . 5

3 Power Consumption of CMOS Circuits **7**
- 3.1 CMOS Background . . . 7
- 3.2 Power Consumption Components . . . 8

4 Introduction to Side Channel Attacks **11**
- 4.1 Basic Principles of Side Channel Attacks . . . 11
- 4.2 Timing Analysis . . . 12
- 4.3 Power Analysis . . . 13
  - 4.3.1 Power Models . . . 13
  - 4.3.2 Simple Power Analysis (SPA) . . . 14
  - 4.3.3 Differential Power Analysis (DPA) . . . 15

5 Stream Ciphers **19**
- 5.1 Introduction to Stream Ciphers . . . 19
  - 5.1.1 Feedback Shift Registers . . . 20
- 5.2 The eSTREAM Project . . . 21
- 5.3 eSCARGOt - European Stream Ciphers Are Ready (to) Go . . . 22
  - 5.3.1 Bit Order of the Key/IV Input and Keystream Output . . . 23

6 Acquisition of Power Traces **25**
- 6.1 Measurement Setup . . . 25
- 6.2 Communication Sequence . . . 31

6.3 Preprocessing of Measured Traces 32

7 Side Channel Analysis Attacks on Grain **35**
7.1 Design Specification of Grain 35
7.2 Adversary Model 36
7.3 Timing Analysis 37
7.4 Simple Power Analysis 37
7.5 Differential Power Analysis 37
7.5.1 Power Model 37
7.5.2 Theoretical Approach 38
7.5.3 Results 42

8 Side Channel Analysis Attacks on Trivium **45**
8.1 Design Specification of Trivium 45
8.2 Timing Analysis and Simple Power Analysis 47
8.3 Differential Power Analysis 47
8.3.1 Power Model 47
8.3.2 Theoretical Approach 47
8.3.3 Results 50

9 Summary **51**
9.1 Future Work 51

1 Introduction

First introduced by Paul C. Kocher in 1996 [Koc96], side channel attacks have become an important and wide area of cryptanalytic research. Instead of performing a mathematical attack on a cryptographic algorithm, side channel attacks can be categorized as physical attacks that exploit sources of information leakages of cryptographic devices to draw conclusions about the secret key. These attacks can be distinguished between active and passive attacks. Fault injection (FI) attacks are members of the active attacks and exploit the feedback gained from a device that is manipulated. These manipulations can induce a faulty behavior during the processing of a cryptographic algorithm that can then be used to disclose secrets. On the other hand, side channel analysis (SCA) attacks are passive attacks that work by analyzing side channels like power consumption [KJJ99] or electromagnetic radiation [QS01] of a cryptographic device.

One of the most powerful side channel analysis attack is the differential power analysis (DPA), where an adversary challenges an embedded device to encrypt a large number of plaintexts and measures the target's power consumption. By simulating the hypothetical power consumption based on the different plaintexts and applying statistical methods to correlate the hypothetical and measured power traces, it is possible to reveal secret information about the key.

This procedure is a well-known technique to attack block ciphers and there are many publications in scientific literature discussing sophisticated extensions to make it more efficient or to adapt it to different ciphers [BK02, BLW03, LSP04, OGOP04, Pro05, OMHT06, Jaf07, EKM+08]. Anyway, so far these attacks are mostly applied to block ciphers and not to stream ciphers. Stream ciphers generate a keystream, which is XORed to the plaintext during the encryption. An adversary faces the problem that new insights cannot be gained by modifying the used plaintext. In this case, a property of stream ciphers plays an important role. Analogous to block ciphers, their output depends on two quantities. For stream ciphers these are typically the key and the initialization vector (IV). While the key is fixed, the IVs vary with every keystream generation. This can be exploited to perform a similar attack as the above-mentioned attack on block ciphers.

In this thesis, we adapt the procedure of the DPA to perform practical attacks on hardware implementations of the two stream ciphers Trivium and Grain. For

these attacks, we create a measurement setup that is well-suited to acquire power traces from our target device, an application-specific integrated circuit (ASIC) called eSCARGOt. Finally, we show that it is possible to extract the whole key by analyzing power traces of only a few steps of the initialization phase.

1.1 Previous Work

Although power analysis attacks are known since the late 90's [KJJ99], in scientific literature, DPA attacks on stream ciphers still have not found much attention. Among the few results, in 2004 Lano *et al.* presented theoretical DPA attacks on the stream ciphers A5/1, used in GSM communications, and the bluetooth algorithm E0 [LMPV04]. Three years later, in 2007, Fischer *et al.* described a practical DPA of an FPGA implementation of Grain and a theoretical DPA of Trivium [FGKV07]. To recover the key of a Grain implementation, they propose three steps: In the first two steps, 34 and 16 values are extracted from the power traces. These values by themselves are not key bits, but define a set of linear equations that includes a subset of 50 of the 80 used key bits and that can be solved to extract all of them. The third step is an exhaustive key search with a complexity of the order 2^{30}. To reduce algorithmic noise, the authors take advantage of a chosen IV attack scenario.

An overview of the possible vulnerabilities allowing side channel analysis attacks on eSTREAM finalists is given in [GBC+08]. Gierlichs *et al.* analyzed all phase 3 candidates of both profiles with respect to their expected resistance to timing and power analysis attacks. So far there are no other works to our knowledge presenting practical SCA results for the eSTREAM ciphers.

1.2 Organization of this Thesis

This thesis is organized as follows. In chapters 2 to 5, some background information is given. After presenting a selection of fundamental statistical methods, we discuss the power consumption of CMOS circuits, which is widely-used for electronic devices. Side channel attacks, including a detailed description of differential power analysis, are introduced in Chapter 4. We close the theoretical part with a chapter on stream ciphers that also introduces the eSTREAM project and, as one result of the project, the target device eSCARGOt. The practical part of the thesis starts with the acquisition of power traces, described in Chapter 6, and ends with the side channel attacks on Grain (Chapter 7) and Trivium (Chapter 8). We summarize this thesis with Chapter 9 and give hints for future works.

2 Statistical Methods

The objective of this chapter is to depict the mathematical foundations for this thesis. It is a small selection of statistical concepts that are necessary to follow the differential power analysis used in the chapters 7 and 8. For a more detailed description see also [LM05].

2.1 Probability Space

A *probability space* is a term of the theory of probability and describes a random experiment. It is denoted as the triple $(\Omega, \mathcal{F}, P)$ and is defined as follows:

- Ω is the sample space - a set of all elementary events. For instance, the sample space of throwing a dice is $\{1, 2, 3, 4, 5, 6\}$.
- $\mathcal{F}$ is a subset of the power set of Ω with the properties
  1. $\Omega \in \mathcal{F}$,
  2. $A \in \mathcal{F} \Rightarrow \Omega \backslash A \in \mathcal{F}$,
  3. $A_1, A_2, \ldots \in \mathcal{F} \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$,

  where A is a set of elementary events.
- $P : \mathcal{F} \to \mathbb{R}$ is the probability measure and satisfies the following axioms:
  1. $P(A) \geq 0$,
  2. $P(A_1 \cup A_2 \cup \ldots) = P(A_1) + P(A_2) + \ldots$ for $A_j \cap A_k = \emptyset$, if $j \neq k$,
  3. $P(\Omega) = 1$.

2.2 Discrete Random Variable

A function X, mapping the sample space Ω to real numbers ($\Omega \to \mathbb{R}$), is called *(real) random variable*. It is defined as *discrete random variable*, if the members of X are denumerable, $x_1, x_2, \ldots, x_n$, with n denoting the number of members of X. In addition, the probability mass function p_i is given as

$$p_i = P(X = x_i) \text{ with } \sum_i p_i = 1.$$

2.3 Expected Value

The *expected value* of a discrete random variable is often confused with the *(arithmetic) mean*. Generally, these two terms can be distinguished by experimental and predicted appearance of the values. If the average is obtained from the results of an experiment of the past, it is denoted as *mean* μ and is given by the equation

$$\mu = AM(X) = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

with

X : a discrete random variable,
x_i : the members of X,
n : the number of members of X.

In contrast, the *expected value* conjectures the average value of a future experiment by taking the probability of the occurrence into account. Hence, the *expected value* of a discrete random variable can be calculated by

$$E(X) = \sum_{i=1}^{n} x_i p(x_i).$$

2.4 Variance and Standard Deviation

The average squared deviation of the expected value is called *variance*. It is defined as

$$\begin{aligned} Var(X) = \sigma^2 &= E\left((X - E(X))^2\right) \\ &= E(X^2) - E(X)^2, \end{aligned}$$

and describes how much a random variable X deviates from its expected value $E(X)$. The square root of the variance is also known as the *standard deviation* σ. An interesting property of the standard deviation can be seen in Figure 2.1. For normally distributed random variables,

- 68.27% of the values are within $\mu \pm \sigma$,
- 95.45% are within $\mu \pm 2\sigma$, and
- 99.73% are within $\mu \pm 3\sigma$.

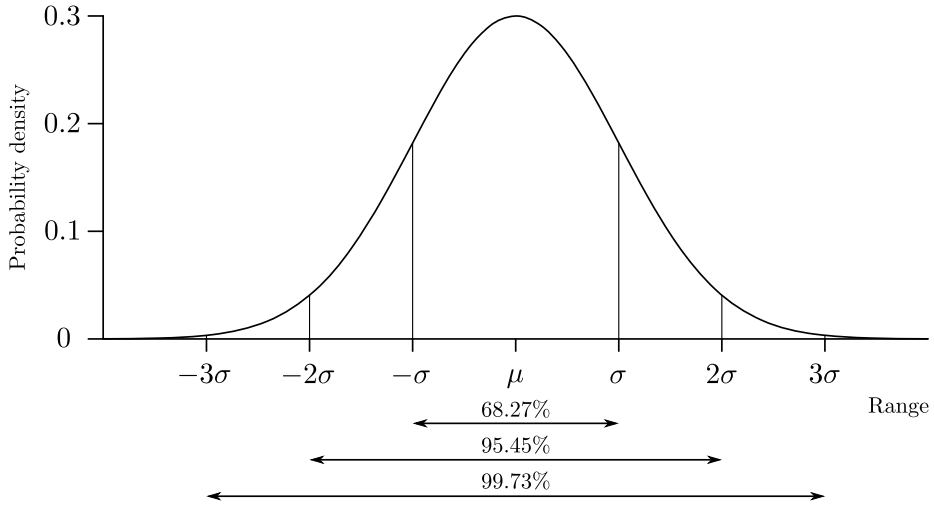


Figure 2.1: Standard deviation diagram of normally distributed random variables.

2.5 Covariance and Correlation Coefficient

The *covariance* can be used to measure the linear relationship between two random variables X and Y. It is defined as

$$\begin{aligned} Cov(X,Y) &= E\Big((X - E(X)) \cdot (Y - E(Y))\Big) \\ &= E(X \cdot Y) - E(X) \cdot E(Y), \end{aligned}$$

and is a more general form of the variance, since

$$\begin{aligned} Cov(X,X) &= E\Big((X - E(X)) \cdot (X - E(X))\Big) \\ &= Var(X). \end{aligned}$$

Depending on the outcome of the covariance, three cases may occur:

- A positive value of the covariance indicates a positive linear relationship of the variables.
- A negative value of the covariance indicates a negative linear relationship of the variables.
- In the case of $Cov(X,Y) = 0$, the variables X and Y are uncorrelated.

To get a more precise description of their interdependency, the covariance is divided by the product of the standard deviation of the two variables. The result of this normalization is the *correlation coefficient*, which can take values between -1 and 1,

and is defined by

$$\begin{aligned} \varrho(X,Y) = r_{XY} &= \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \\ &= \frac{E((X-E(X))(Y-E(Y))}{\sqrt{Var(X)} \cdot \sqrt{Var(Y)}}. \end{aligned}$$

A high *correlation coefficient* indicates a strong positive linear relationship between X and Y, a strong negative linear relationship is given by values near -1. A value around 0 stands for low or no linear interdependency. Conversely, this does not mean that there is no relationship between X and Y at all. Nevertheless, a non-linear dependency may be given in this case.

To correlate two series of measurements G and H with the values $g_1, g_2, \ldots, g_n$ and $h_1, h_2, \ldots, h_n$, e.g., power traces, the *Pearson product-moment correlation coefficient* can be used for computation:

$$\varrho(G,H) = r_{GH} = \frac{\sum_{i=1}^{n}(g_i-\bar{g})(h_i-\bar{h})}{\sqrt{\sum_{i=1}^{n}(g_i-\bar{g})^2 \sum_{i=1}^{n}(h_i-\bar{h})^2}},$$

where $\bar{g} = \mu_G$ and $\bar{h} = \mu_H$ are the mean values of G and H.

3 Power Consumption of CMOS Circuits

CMOS (complementary metal-oxide-semiconductor) is a widespread technology to realize logical functions and is used, e.g., in microprocessors, RAM, ASICs, and other digital logic circuits. In this chapter we will concentrate on the power consumption of these circuits. Generally, the power consumption can be divided into a data-dependent and a data-independent part. For power analysis attacks the data-dependency plays a large role, because it can be exploited to obtain secret information from power traces. Hence, we will focus on this after a short introduction to the basic architecture of CMOS circuits.

3.1 CMOS Background

The basic components of CMOS circuits are MOSFETs (metal-oxide-semiconductor field-effect transistors), which can be regarded as electronic switches. Generally, we distinguish between two types of MOSFETs, p-type (PMOS) and n-type (NMOS) transistors (see Figure 3.1). The current flow from drain to source is controlled by

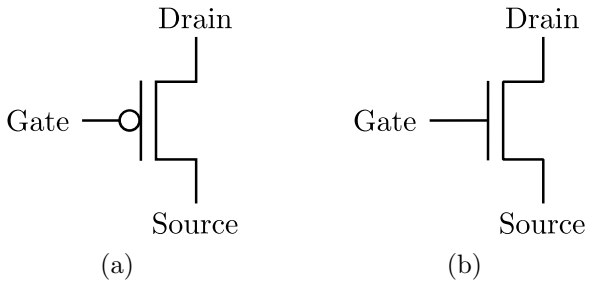


Figure 3.1: Symbols of a PMOS (a) and an NMOS (b) transistor.

the voltage between gate and source. While a PMOS transistor conducts when a negative voltage is applied, an NMOS transistor conducts if this voltage is positive. The complementary arrangement of both, PMOS and NMOS transistors, is the main

property of a CMOS logic style. As an example, a CMOS inverter cell is depicted in Figure 3.2.

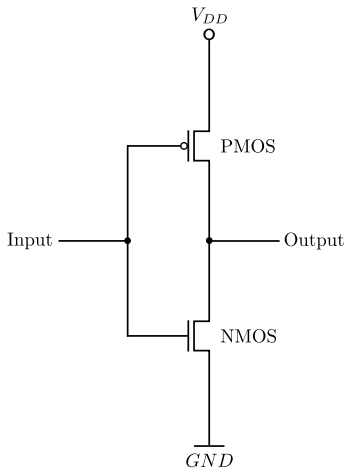


Figure 3.2: A simple circuit diagram of a CMOS inverter.

The circuit can be divided into a pull-up and a pull-down network. The connection of the output through the PMOS to the voltage source is called pull-up, the connection through the NMOS to the ground pull-down network. To accomplish the complementary effect, the gates of both MOSFETs are controlled by the same input. This makes sure that only one network conducts while the other one insulates. If V_{DD} (logical 1) is connected to the input, the NMOS conducts and the output is a logical 0. Otherwise, when the input signal is a logical 0, the conducting PMOS induces a logical 1 at the output.

3.2 Power Consumption Components

The data-dependent power consumption of CMOS circuits is the main source of information that can be exploited for power analysis attacks. The average power consumption P_{avg} is composed of three major sources and can be split into a dynamic (data-dependent) and a static (data-independent) part [CB95]:

$$P_{avg} = \underbrace{P_{leakage}}_{static} + \underbrace{P_{switching} + P_{short-circuit}}_{dynamic}.$$

For side channel attacks, the static part of the term is of minor importance. It remains constant during the complete time period and therefore contains no information about the processed data. One component of $P_{leakage}$ is the subthreshold leakage that is

characterized by a weak diffusion current of an insulating MOSFET between source and drain.

When switching the state of a CMOS circuit, the power consumption changes significantly for two reasons:

Capacitive Load: The wires and possibly gate electrodes of successive MOSFETs form a capacitor C_L. The size of C_L depends on the length of the wires and the number of successive CMOS cells. It is charged over the PMOS at every transition from a logical 0 to a logical 1 and discharged over the NMOS at every opposite transition. The average charging power of a CMOS cell at a clock rate of f_{CLK} is described with the equation [CB95]

$$P_{switching} = \alpha_{0\to1} \cdot C_L \cdot V_{DD}^2 \cdot f_{CLK},$$

where $\alpha_{0\to1}$ is defined as probability of occurrence of a power consuming transition $0 \to 1$. With regard to a measurement setup, this transition can only be noticed when measuring the voltage drop at V_{DD}. In the other case, when measuring at GND, the insulating NMOS prevents detecting this type of transition. Instead, a discharging current of C_L at transition $1 \to 0$ can be identified.

Short-Circuit Current: The second dissipation of a CMOS circuit is the short-circuit current $P_{short-circuit}$. Let V_{TN} and V_{TP} be the thresholds of NMOS and PMOS for insulating or conducting the path between source and drain. In practice, there is no instantaneous switching from one logical value to the other. This leads to a short time period during a transition where the input voltage V_{in} reaches a value that is exactly between both thresholds, $V_{TN} < V_{in} < V_{DD} - |V_{PN}|$ [CB95]. During this time, both transistors conduct and a short-circuit occurs.

In summary, in terms of analyzing the dynamic power consumption, we can conclude that the transitions $0 \to 1$ and $1 \to 0$ do not produce the same peak, because of the capacitive load described above. However, in most cases we can neglect this difference: Compared to the short-circuit that occurs in every transitions, the charging and discharging of C_L, respectively, represent only a small amount of the overall dynamic power consumption. Therefore, we only distinguish between the static power consumption, which is very low and can also be neglected, and the dynamic power consumption, depending on the processed data.

4 Introduction to Side Channel Attacks

This chapter gives a brief overview of the most common side channel attacks. After describing the basic principles, we will focus on passive attacks, especially on power analysis attacks.

4.1 Basic Principles of Side Channel Attacks

Regardless of the security of a cipher in theory, an implementation of this cipher can lead to new vulnerabilities. Ciphers that previously were considered safe can suddenly be attacked with simple methods. Side channel attacks do not target the encryption technique, but the secondary effects that occur during the execution of an implementation. For example, the attacks exploit that different types of operations require a different number of clock cycles (Section 4.2), or that the power consumption of a physical device varies, depending on the processed data (Section 4.3). These so-called leakage data can often be used to draw conclusions about the secret key. Figure 4.1 illustrates the different side channels of an encryption.

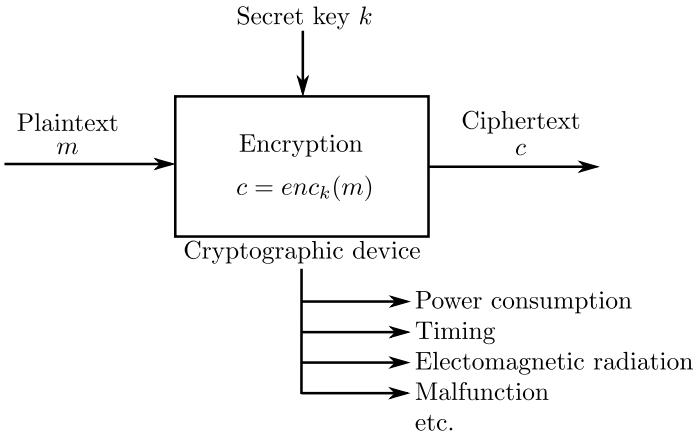


Figure 4.1: Possible side channels of a cryptographic device during an encryption.

Generally, one can differentiate between active and passive side channel attacks. Active attacks are the Fault Injection (FI) attacks [BS96], in which the adversary interferes with the encryption to force a malfunction during the computation. In certain operations, e.g., when generating an RSA signature using the Chinese remainder theorem, secret information about the key can be revealed. The most common sources to generate faults for an attack are [BECN+04]

- laser / light (see also [SA02]),
- power spikes (see also [AK96]),
- high temperature,
- overclocking,
- X-rays and ion beams.

Certainly, such manipulations can cause damages to the device. Another disadvantage is the mostly complex setup to induce the faults to the device. In contrast, passive attacks are rather simple to arrange. Two common types of passive attacks are presented in the following sections.

4.2 Timing Analysis

The timing analysis exploits the data-dependency of the timing behavior and was introduced by Kocher in 1996 [Koc96]. Basic requirements are some knowledge about the implementation and the data-dependency of the elapsed time, e.g., due to conditional branches during the computation.

A typical target for a timing attack is the RSA exponentiation using the square-and-multiply algorithm and the Montgomery reduction [DKL+00]:

Let m be the plaintext, n the RSA modulus, and k the secret key with

$$k = k_{known}||k_{unknown}.$$

The RSA exponentiation is given as $m^k \bmod n$. We assume that the first bit of the unknown part of the key is $k_i = 1$ and calculate

$$m^{k_{known}||k_i} \bmod n,$$

with the square-and-multiply algorithm. Depending on the plaintext m and the exponent $k_{known}||k_i$, in the last step of the square-and-multiply algorithm the Montgomery reduction is either performed or not. By doing this several times with changing m_i and a constant k_i, the elapsed times of the encryptions can be split into two sets:

F_1 : including all times where the reduction was performed in the last step,
F_2 : including all times without reduction in the last step.

To verify our assumption, the means of F_1 and F_2 are computed. If $\phi(F_1) = \phi(F_2)$, our separation was wrong and we can discard the assumption. Otherwise, if $\phi(F_1) > \phi(F_2)$, our separation and the assumption were correct. In both cases, the key bit is obtained and we can attack the next bit in the same way.

4.3 Power Analysis

Other attacks are based on power analyses. As explained in Section 3.2, the overall power consumption of a cryptographic device can be divided into a static and dynamic part. Since the dynamic power consumption is connected directly with the processed data, it is a potential target to detect the dependency between these two parameters. For that reason, power traces can be used to obtain secret information. There are mainly two attacks using this approach, the simple power analysis and the differential power analysis. Before we describe these two attacks, the connection of data and power consumption have to be clarified.

4.3.1 Power Models

To perform a successful attack, finding out the connection between processed data and power consumption is important. Considering this information it is possible to simulate power traces with varying data to compare them with the actually measured trace. However, it is not important to determine the exact power consumption, but rather the relative differences between the time intervals. In the following, the two most commonly used power models are explained.

Hamming Distance Model

The Hamming distance model simulates the power consumption in a digital circuit based on the number of transitions in a certain time interval, i.e., $0 \rightarrow 1$ and $1 \rightarrow 0$, respectively. We illustrate this using an example:

Typically, shift registers are realized with CMOS flip-flops that are connected in series and clocked synchronously. In Section 3.2 we have described that the power consumption changes significantly when changing the input of a circuit. This property can also be applied to flip-flops. Hence, if the input of a flip-flop stays the same, the power consumption is only composed of the static power consumption, which can be neglected. When changing the input, the power consumption of the flip-flop rises rapidly due to the dynamic power consumption. The Hamming distance model simulates the power trace based on the number of transitions in every clock cycle. In

the case of shift registers, this leads to a trace that usually has a strong correlation to the measured trace.

Other possible applications of the Hamming distance model are devices with long data buses that have a big capacitive load, for instance, microcontrollers [MPO]. The Hamming distance (HD) of a bus or a register can then be calculated with the Hamming weight (HW), which counts simply the number of 1s.

Let v_0 be the actual value and v_1 the successor. The Hamming distance is defined as

$$HD(v_0, v_1) = HW(v_0 \oplus v_1).$$

Hamming Weight Model

This model is much simpler than the Hamming distance model and is used when there is no knowledge about the internal structure of the device or consecutive values of some processes. It involves a relationship between the power consumption and the Hamming weight of the processed data. Generally, the Hamming weight model is not well-suited for simulating the consumption of a CMOS circuit. An example of use for this model is an AES implementation on a smart card.

4.3.2 Simple Power Analysis (SPA)

In a simple power analysis attack, only one or a few power traces are analyzed to determine hidden information. This information can be the type and length of an operation, how often and in which order they appear, the usage of conditional branches, or in certain cases the secret key. In most cases, an SPA attack requires a detailed knowledge of the algorithm.

As a simple example, we can again review the square-and-multiply algorithm. Generally, multiplications are more time-consuming than squarings. A closer look at a power trace can reveal whether a squaring or a multiplication is executed. From this, the adversary is able to detect every single bit of the exponent by distinguishing between a 0, which is only a squaring, or a 1, squaring with a subsequent multiplication.

In [MPO], Mangard *et al.* differentiate between *single-shot SPA* attacks and *multiple-shot SPA* attacks. In single-shot SPA attacks, only one power trace can be recorded by the adversary. This requires a highly noise-reduced generation of the trace. In multiple-shot SPA attacks, multiple traces can be used to reduce the noise afterwards, e.g., by averaging the traces.

4.3.3 Differential Power Analysis (DPA)

In contrast to an SPA attack, a DPA needs a large number of traces and applies statistical methods to reveal the secret key. Due to the large number it is possible to extract information even from extremely noisy traces. A precondition of a DPA is that the adversary has knowledge either of the plaintext or the ciphertext and is able to predict key-dependent intermediate values of the attacked algorithm. In the following, we present the most common strategies of a DPA.

DPA with Difference of Means Test

Let $f(d, k)$ be an intermediate result that only depends on the known plaintext d and a part of the secret key k, e.g., an output of an S-box. The first step of the DPA is the measurement phase. For random inputs $d_1, \ldots d_D$, the power traces $t_1, \ldots t_D$ are recorded using the unknown key. In a second step, the adversary selects a so-called *Boolean selection function* b. This can be, for instance, a function that returns one defined bit of the intermediate value. Then the key guessing phase begins. Assuming one key $\tilde{k}$, the adversary computes $b(f(d_i, \tilde{k}))$, for $i = 1 \ldots D$ and partitions the traces recorded in the first step in two sets:

- $\mathcal{S}_0$, containing all traces for which $b(f(d_i, \tilde{k})) = 0$,
- $\mathcal{S}_1$, containing all traces for which $b(f(d_i, \tilde{k})) = 1$.

After all D traces have been allocated, the difference between the mean values of the two sets is evaluated by calculating

$$\Delta_{\tilde{k}} = \frac{\sum\limits_{i \in \mathcal{S}_1} t_i}{|\mathcal{S}_1|} - \frac{\sum\limits_{i \in \mathcal{S}_0} t_i}{|\mathcal{S}_0|}.$$

Every wrong key guess leads to a trace near zero for all time periods. In contrast, $\tilde{k} = k$ results to a trace containing a peak at time period τ. This is exactly that time, in which the computation of $f(d_i, k)$ takes place.

The occurrence of the peak is based on the chosen power model. Suppose that f is a software implementation on an 8-bit processor. Selecting the Boolean selection function as mentioned above, results to the sets $\mathcal{S}_0$ with seven uniformly distributed bits plus one 0, and $\mathcal{S}_1$ with, again, seven uniformly distributed bits, but plus an additional 1. Hence, the expected Hamming weights of these two sets are $HW(\mathcal{S}_0) = 3.5$ and $HW(\mathcal{S}_1) = 4.5$, provided that the traces have been correctly partitioned due to the correct key guess. If the power consumption obeys the Hamming weight model, this leads to the peak at time period τ.

This method was firstly introduced by Kocher *et al.* in 1999 [KJJ99]. By choosing an intermediate result that only depends on a small part of the key, the adversary

pursues the *divide-and-conquer* strategy. In this strategy the adversary divides one big problem into several smaller ones, e.g., the revealing of an 128-bit key is achieved by attacking 16 8-bit S-boxes. Hence, the effective key space decreases from 2^{128} to $16 \cdot 2^8 = 2^{12}$.

DPA with Correlation Coefficients

Another approach based on Kocher's method is the DPA with correlation coefficients as criterion for correct key guesses. Mangard *et al.* described this attack in detail by using five steps [MPO]. Basically, the first three steps are also performed in the DPA from Kocher.

Step 1: Choosing an Intermediate Result of the Executed Algorithm. In this step we first choose the function f with the same properties discussed above. The intermediate result is denoted as $f(d, k)$.

Step 2: Measuring the Power Consumption. Using the unknown key, we encrypt or decrypt random inputs $d_1, \ldots d_D$. As result, we get D power traces $t_1, \ldots t_D$, which can be combined to a $D \times T$ matrix, where T is the number of measurement points per trace.

$$\mathcal{T} = \begin{pmatrix} t_{1,1} & t_{1,2} & \ldots & t_{1,T} \\ \vdots & & & \vdots \\ t_{D,1} & t_{D,2} & \ldots & t_{D,T} \end{pmatrix}$$

It is important that these traces are perfectly aligned, which means that the measurement points of one column are recorded exactly in the same time period of the computations for every trace.

Step 3: Calculating Hypothetical Intermediate Values. For every key hypothesis

$$\mathcal{K} = k_1, k_2, \ldots, k_K,$$

with K denoting the number of possible keys, all intermediate values $f(d_i, k_j)$ for $i = 1, \ldots, D$ and $j = 1, \ldots, K$ are computed. The matrix we obtain has the size $D \times K$.

$$\mathcal{V} = \begin{pmatrix} v_{1,1} & v_{1,2} & \ldots & v_{1,k} \\ \vdots & & & \vdots \\ v_{D,1} & v_{D,2} & \ldots & v_{D,K} \end{pmatrix}$$

Note that one key hypothesis of $\mathcal{K}$ is the correct key used in Step 2. Hence, one column of matrix $\mathcal{V}$ contains the intermediate results that produced the recorded traces.

Step 4: Mapping Intermediate Values to Power Consumption Values. In this step, we select an appropriate power model to simulate the power consumption in dependency of the intermediate values. The choice of the correct power model is decisive for the efficiency of the DPA. Two common power models have been discussed in Section 4.3.1, the Hamming distance model and the Hamming weight model. The mapping of

$$v_{i,j} \rightarrow h_{i,j} \text{ for } i = 1, \ldots, D \text{ and } j = 1, \ldots, K$$

results to the $D \times K$ matrix $\mathcal{H}$:

$$\mathcal{V} = \begin{pmatrix} v_{1,1} & v_{1,2} & \ldots & v_{1,k} \\ \vdots & & & \vdots \\ v_{D,1} & v_{D,2} & \ldots & v_{D,K} \end{pmatrix} \rightarrow \mathcal{H} = \begin{pmatrix} h_{1,1} & h_{1,2} & \ldots & h_{1,k} \\ \vdots & & & \vdots \\ h_{D,1} & h_{D,2} & \ldots & h_{D,K} \end{pmatrix}.$$

Step 5: Comparing the Hypothetical Power Consumption Values with the Power Traces. In Step 4, we have simulated the power consumption with all possible key hypotheses for every input value used to generate the power traces. Hence, one column of our power hypotheses matrix $\mathcal{H}$ strongly correlates with the leakage point. All we have to do is comparing the two matrices $\mathcal{T}$ and $\mathcal{H}$ column by column by applying the correlation coefficient (see Section 2.5). Again, we can summarize the result of the computations

$$r_{j,l} = corr(\mathcal{H}_j, \mathcal{T}_l),$$

for $j = 1, \ldots, K$ and $l = 1, \ldots, T$, in a matrix of size $K \times T$:

$$\mathcal{R} = \begin{pmatrix} r_{1,1} & r_{1,2} & \ldots & r_{1,T} \\ \vdots & & & \vdots \\ r_{K,1} & r_{K,2} & \ldots & r_{K,T} \end{pmatrix}.$$

The value with the highest correlation coefficient, $r_{k,\tau}$, reveals the secret key k and the time of the leakage τ.

5 Stream Ciphers

In the following, an introduction to stream ciphers is given. In Section 5.2, we describe the multi-year project eSTREAM that selected promising new stream ciphers in three evaluation phases. Afterwards, one result of this project, an application-specific integrated circuit (ASIC) called eSCARGOt, is presented, which contains hardware implementations of the last evaluation phase.

5.1 Introduction to Stream Ciphers

Stream ciphers are very popular for real-time applications because of their low hardware complexity and high performance. Compared to block ciphers, they do not have a predefined size of plaintext that has to be encrypted. As a consequence, the plaintext can be encrypted immediately without latency. This is important for real-time applications, for instance, the A5/1 algorithm used for mobile phone communication.

A characteristic of stream ciphers is that each bit is encrypted individually. Basically, stream ciphers can be considered as pseudo-random number generators (PRNGs) that generate a keystream from a short input key. During encryption, the sender XORs the keystream bit by bit with the plaintext. The receiver owns the same input key to reconstruct the keystream and obtain the plaintext by also XORing the keystream to the ciphertext. A general encryption and decryption process is illustrated in Figure 5.1.

A popular example for a stream cipher is the One Time Pad (OTP), co-invented in 1917 by G. Vernam and J. Mauborgne [Sch95]. In 1949, C. E. Shannon proved that the OTP has the property of *perfect secrecy* [Sha49]. This means that an adversary is not able to gain any new insights with the possession of the ciphertext. The OTP uses, in contrast to the above-mentioned stream ciphers, a true random number generator (TRNG) to create a keystream that is only allowed to be used once. However, the keystream generation has a great negative effect. Because of the true randomness, the keystream cannot be reproduced. Hence, the whole keystream can be seen as secret key, which has the same length as the encrypted plaintext.

Because the OTP is perfectly secure, it follows that the security of stream ciphers depends on the non-predictability of the pseudo-random function that creates the

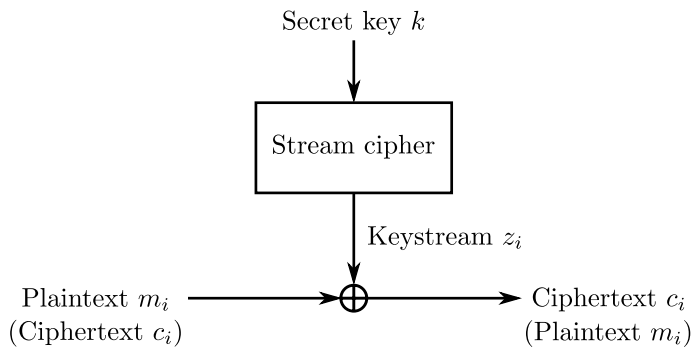


Figure 5.1: Encryption (decryption) scheme of a stream cipher.

keystream. An often used building block to realize a keystream generator is a combination of shift registers.

5.1.1 Feedback Shift Registers

A feedback shift register is a register of an arbitrary size n that moves its content bits synchronously to one direction. This implicates that there is an output and an input bit - the one falling out of the register and the one filling the gap. The maximum length of an output sequence is $2^n - 1$, the number of possible states of the register. Then, at the latest, it starts from the beginning. The input bit is the result of a feedback function. Depending on this kind of function, the register is also denoted as *linear feedback shift register* (LFSR) or *non-linear feedback shift register*(NLFSR). An example for an LFSR is depicted in Figure 5.2. Here, the input of the register is given by the states of the bits z_0, z_3, and z_7.

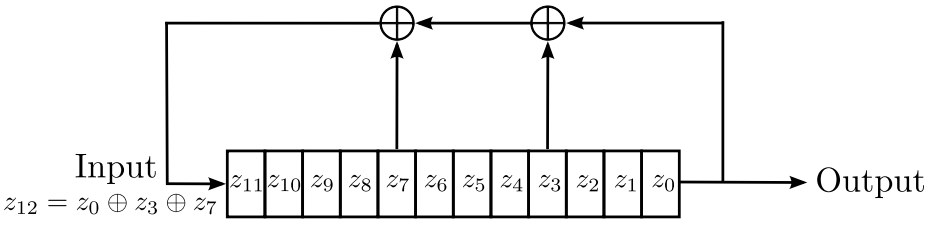


Figure 5.2: An example for a Linear Feedback Shift Register (LFSR).

Initially, a register is filled with the so-called *seed* or *initialization vector* (IV). In case of an LFSR, this must be non-zero to prevent a zero-only state.

Linear feedback shift registers, individually, are extremely insecure. For this reason, they are often combined, e.g., with non-linear combining functions. Another

possibility to enhance the security is based on the *alternating stop-and-go generator*. While usually all registers are clocked at the same time, the alternating stop-and-go generator uses an irregular clocking. This is realized by a register that, depending on the output, decides which of the successive registers is clocked.

5.2 The eSTREAM Project

The eSTREAM project was founded in 2004 with the intention to find stream ciphers that are suitable for widespread adoption [ECRa]. The initiator was the 4-year network ECRYPT, European Network of Excellence for Cryptology, that has taken up the cause to "intensify the collaboration of European researchers in information security, and more in particular in cryptology and digital watermarking" [ECRb]. The successor, ECRYPT II, started in August 2008.

After their call for primitives, 34 candidates had been submitted to eSTREAM. The ciphers were partitioned into two profiles [ECRa]:

- Profile 1: Stream ciphers for software applications with high throughput requirements.
- Profile 2: Stream ciphers for hardware applications with restricted resources such as limited storage, gate count, or power consumption

In three evaluation phases, the ciphers were analyzed, e.g., with respect to security, performance, and simplicity. All in all, 8 stream ciphers made it to the final portfolio (see also Table 5.1).

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

Table 5.1: The eSTREAM Portfolio [ECRa].

In the same year of the announcement, M. Hell and T. Johansson published a cryptanalytic attack against the cipher F-FSCR-H v2 [HJ08], which causes the ECRYPT to revise the portfolio and eliminate F-FSCR-H v2 from the list.

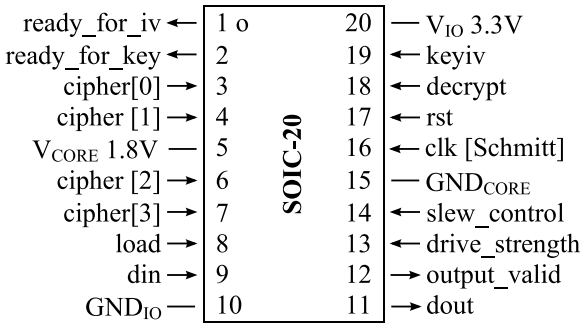


Figure 5.3: Interface of the eSCARGOt ASIC taken from [GB08].

5.3 eSCARGOt - European Stream Ciphers Are Ready (to) Go

The target device is a 0.18 μm ASIC called eSCARGOt. Designed by T.Good and M. Benaissa, it contains the implementation of all hardware profile stream cipher candidates of Phase 3 submitted to eSTREAM [GB08], which are

- Moustique,
- Edon80,
- Trivium,
- Decim, Decim-128,
- F-FCSR-H, F-FCSR-16,
- Grain, Grain-128,
- Mickey, Mickey-128,
- Pomaranch, Pomaranch-128.

In addition, accelerated designs for Grain and Trivium are implemented:

- Grain (x8 internally),
- Trivium (x8 internally).

The pin assignment is given in Figure 5.3. The eSCARGOt requires two supply voltages, 3.3 V for I/O and 1.8 V for the internal core of the chip. It is clocked by an external clock with a frequency of maximum 50 MHz on Pin 16. Pins 3, 4, 6, and 7 (*cipher*[0] to *cipher*[3]) are the input for the cipher selection. Most of the protocols, e.g., for transmitting the key or the IV, are carried out with handshaking. Pins 1, 2, 8, and 12 are intended for this use.

In addition to the keystream generation, the eSCARGOt provides the possibility to directly XOR the keystream to the input data supplied to Pin 9 (*din*). The output is

given on Pin 11 (*dout*). For further information about the interface and the operation modes we refer to the data sheet [GB08].

5.3.1 Bit Order of the Key/IV Input and Keystream Output

A peculiarity can be found in conjunction with the key or IV transfer, especially for Trivium. Due to the non-standardization of the bit order, the input of the key and IV and the output of the keystream are not identical for all ciphers. For the implementation, T.Good and M. Benaissa chose an order that complies with the default test vectors. The result is given in Table 5.2 [GB08].

cipher	key/IV	keystream
Moustique	normal	normal
Trivium	quad byte swapped	quad byte swapped
Pomaranch	normal (but 18-bit hex values)	normal
Mickey	normal	normal
Grain	bits in 8-bit bytes reversed	bits in 8-bit bytes reversed
F-SCSR-H	bytes reversed	normal
F-SCSR-16	bits in 16-bit bytes reversed	byte pairs swapped
Edon80	normal	normal
Decim	bits reversed	bits reversed

Table 5.2: Input and output bit order of the eSCARGOt stream ciphers [GB08].

For this thesis, three bit orders were tested, which are

- *bits in 8-bit bytes reversed*, for Grain,
- *quad byte swapped*, for Trivium, and
- *normal*, as a reference (e.g., used for Mickey).

The *normal* bit order is simply the serial input/output of the bits from the most significant bit (MSB) to the least significant bit (LSB). For the other two orders, the description is a little bit misleading. *Bits in 8-bit bytes reversed* is actually the same as *normal*. The order used for Trivium, *quad byte swapped*, corresponds to the

bit order description	hexadecimal	binary
normal	1A 2B 3C 4D	00011010 00101011 00111100 01001101
bits in 8-bit bytes reversed	1A 2B 3C 4D	00011010 00101011 00111100 01001101
quad byte swapped	4D 3C 2B 1A	01001101 00111100 00101011 00011010

Table 5.3: Comparison of the applied bit orders for Grain and Trivium.

little-endian representation with an 8-bit access. The bits are taken byte-wise from right to left. Table 5.3 gives an example of the three different transfer possibilities.

6 Acquisition of Power Traces

In the last chapters, we presented the theoretical background of our work. In this chapter, we describe the experimental part, which starts with the measurement setup for generating power traces. After this, we propose how the traces are preprocessed for the DPA.

6.1 Measurement Setup

The measurement setup consists of four parts, which are the four basic components PC, oscilloscope, microcontroller, and ASIC. In the following, we describe the components and how they interact with each other.

Personal Computer (PC)

The PC can be regarded as the control unit that transmits the configuration data to the oscilloscope and microcontroller. For this purpose, we use a standard desktop PC with no special features. The only requirement is a hard disk with enough free disk space to save the traces. The communication with the oscilloscope is done via an Ethernet interface. For this reason, a so-called VISA (virtual instrument software architecture) session is established to transfer configuration data like time window, sample rate, etc.

Digital Sampling Oscilloscope

The power traces are measured using an Agilent Infiniium 54832D oscilloscope. It is a 1 GHz Mixed-Signal Oscilloscope (MSO), with 4 scope channels, 16 timing channels, and an analog/digital vertical resolution of 8 bits. For our setup, at least two probes are needed: One for the measurement of the power traces and one to detect a suitable trigger signal that indicates the beginning of the measurement. We decided to set the time window to 4 ms and the sample rate to 1 GS/s.

ATmega32L

As an interface between the eSCARGOt and the PC, we chose the Atmel ATmega32L, which is an 8-bit microcontroller with a RISC architecture. Most instructions can be executed in a single clock cycle. Hence, at a speed of 8 MHz up to 8 MIPS can be achieved. Additionally, a 32 KB flash memory and a 2 KB internal SRAM are integrated.

The difference between the ATmega32 and the ATmega32L is primarily the operating voltages. While the ATmega32 is designed for voltages from 4.5V to 5.5V, the ATmega32L is a low-power version that even works with voltages as low as 2.7V. This is a great advantage, since it is compatible to the supplying voltage of the eSCARGOt chip, which is 3.3V.

In order to operate with the microcontroller, we designed a printed circuit board (PCB). The schematic is given in Figure 6.1, the corresponding board layout in Figure 6.2. Basically, we have two main components in the schematic. To the right, we have the microcontroller with two connectors (top left) connecting Port A and C with the eSCARGOt for the data exchange. Below these connectors, also belonging to the microcontroller, there is a reset button to reset the microcontroller and an 8 MHz quartz crystal as clock source. Although the ATmega32L provides an internal oscillator, it is recommended to use this quartz crystal instead, due to its higher stability. Relying on the internal oscillator can lead to jitters that is a decisive factor of an unsuccessful DPA due to the misalignment of power traces. In addition to the required layout, three LEDs are connected to PD4, PD6, and PD7. These LEDs are for debugging purposes only.

The other main component is responsible for the communication with the PC. The FT232RL is a USB to serial UART interface which supports data transfer rates from 300 baud to 3 Megabaud. We included a voltage regulator that reduces the input voltage of 5V coming from the USB to 3.3V. The pins TXD and RXD are used to transmit and receive data, RTS# and CTS# for the hardware handshaking.

On the PC side we installed a virtual COM port driver that causes the USB device to appear as an additional COM port. The configuration for the communication with the FT232RL is given as follows:

- Baud rate: 9600 Bd,
- Parity: none,
- Data bits: 8,
- Stop bits: 1.

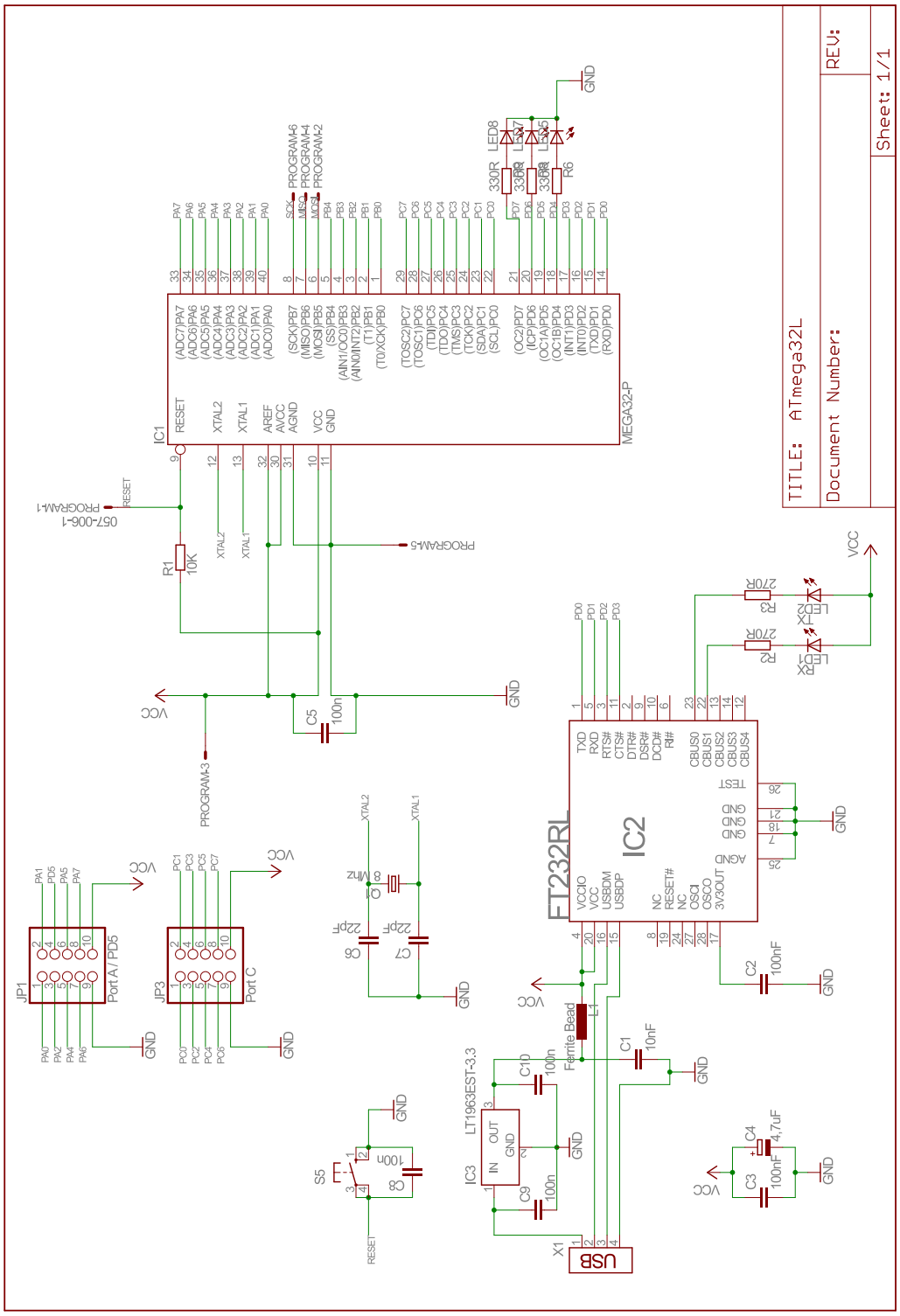


Figure 6.1: Schematic of the PCB for the ATmega32L

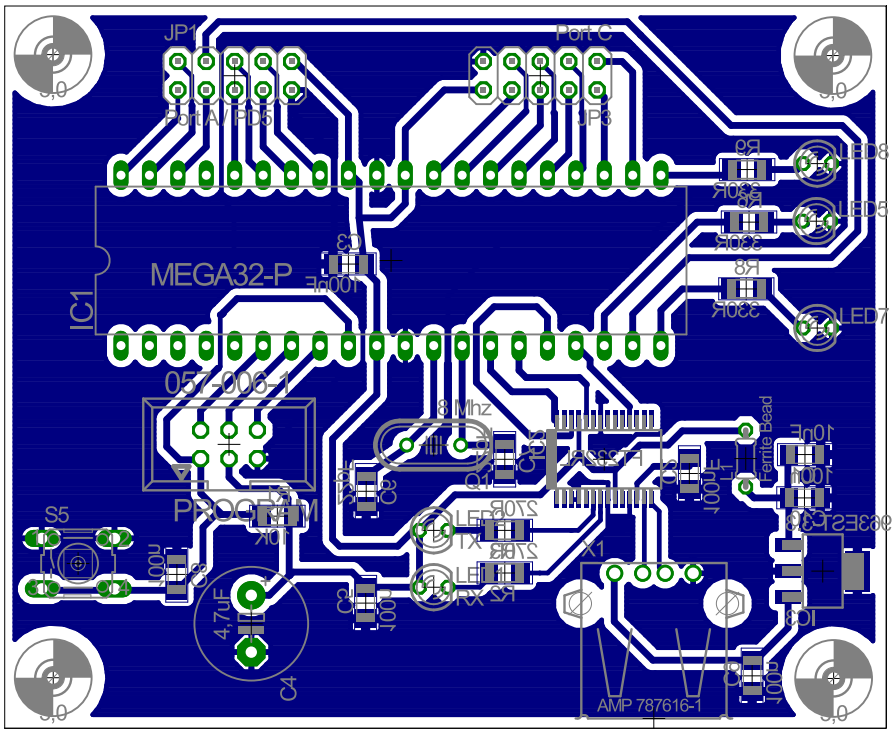


Figure 6.2: Board layout of the PCB for the ATmega32L

Applied Software. Atmel provides a helpful integrated development environment (IDE), which is called AVR Studio. It includes a project management tool, a source file editor, a debugger, and a chip simulator. The supported languages are Pascal, BASIC, Assembly, and C. For the programming of the microcontroller, we chose the actual version AVR Studio 4 and the language C. After the compilation, the program was transmitted via a programmer to the microcontroller. The tool that accomplished the transmission was AVRdude. To activate the external 8 MHz quartz crystal, some fuse bits of the microcontroller have to be enabled/disabled[1]. This can also be achieved with AVRdude, or alternatively with PonyProg.

For the PCB, we used a layout editor from CadSoft, called EAGLE (version 5). The software provides three main components, which are

- a schematic editor to interconnect the electronic devices,
- a layout editor, where the wiring is automatically adopted from the schematic editor,
- and an autorouter that suggests suitable routing possibilities.

[1]For further information see also the microcontroller data sheet.

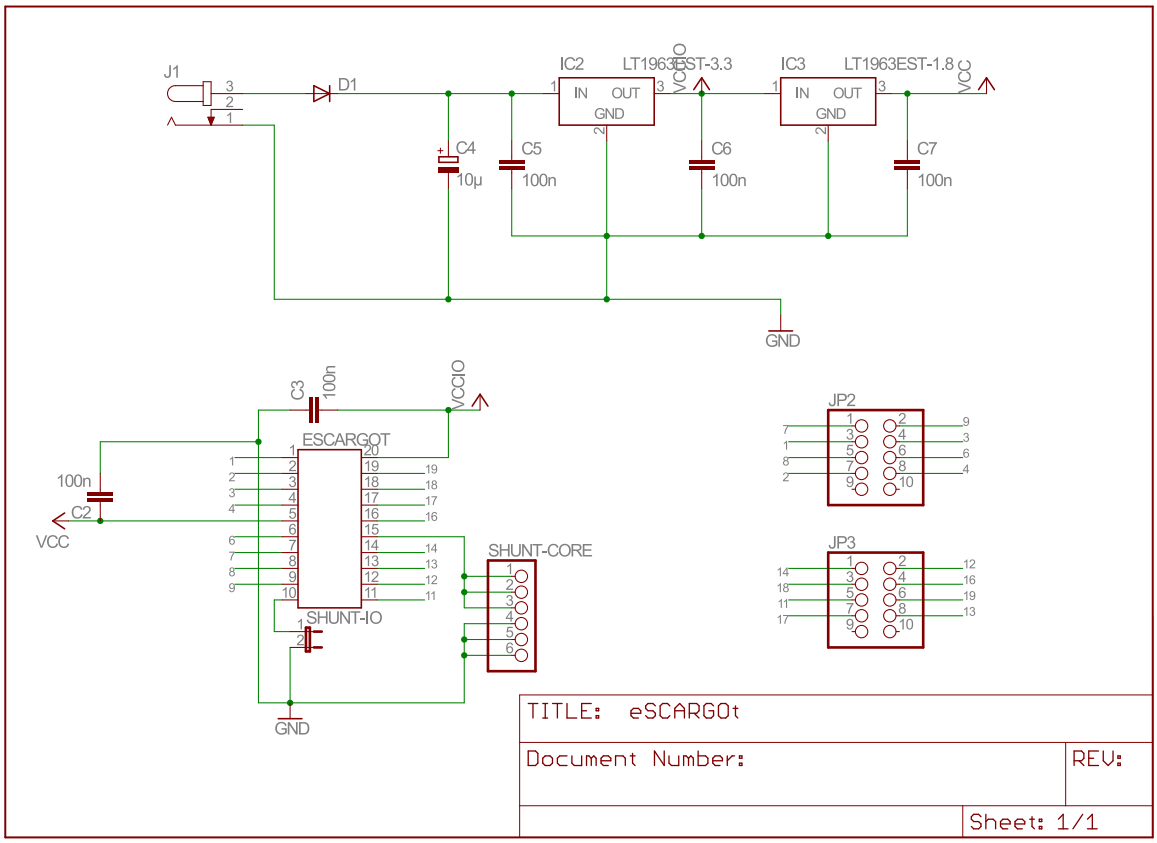


Figure 6.3: Schematic of the PCB for the eSCARGOt.

The figures 6.1, 6.2, 6.3, and 6.4 result from this software.

The communication through the virtual serial port between PC and microcontroller has been realized using Matlab R2008a[2]. Matlab is a high-level language for technical computing that has its strength in mathematical functions, such as statistics. Hence, it is well-suited for future DPA attacks.

eSCARGOt

As for the microcontroller, we also etched a PCB for the eSCARGOt chip, which is presented in Figure 6.3 (schematic) and Figure 6.4 (layout).

The ASIC is powered by a stabilized power supply. Two voltage regulators make sure that the input voltages comply with the specification (3.3V and 1.8V). As a measure of precaution, we decided not to solder the eSCARGOt to the board. In-

[2]See also http://www.mathworks.com.

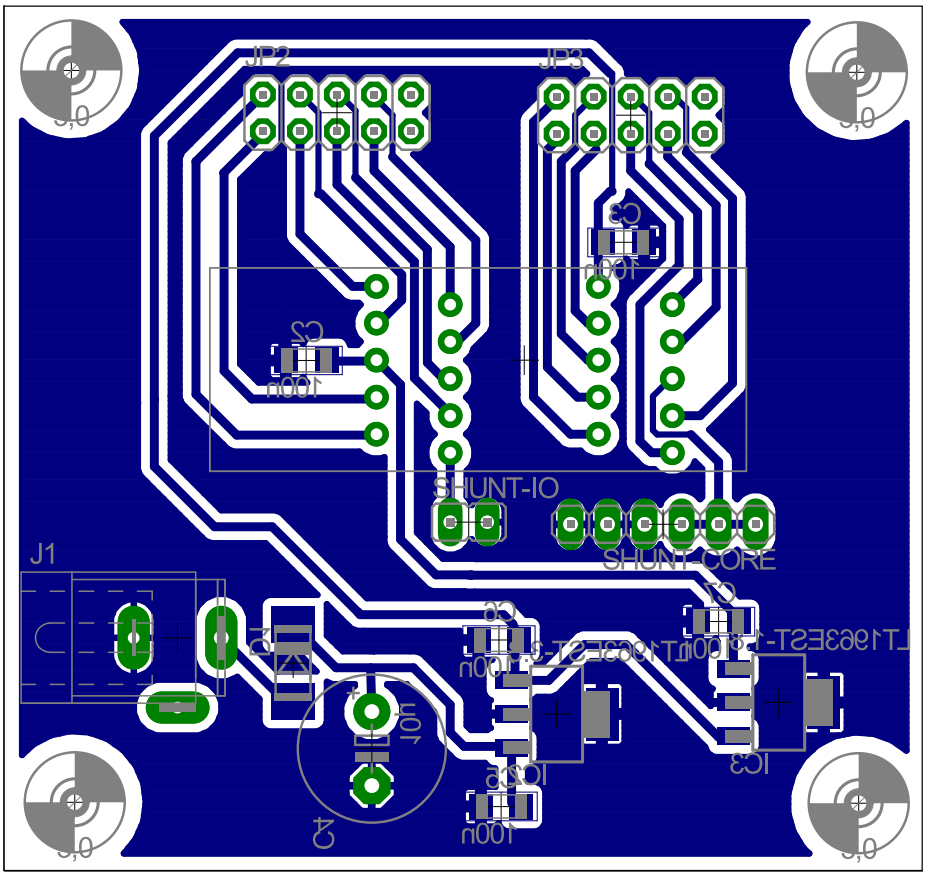


Figure 6.4: Board layout of the PCB for the ATmega32L

stead, an SOIC burn-in test socket is applied, which allows a clean removal of the eSCARGOt for other experiments.

In order to measure the power consumption, the PCB provides a power measurement circuit. Because digital oscilloscopes only have the ability to measure voltages, a shunt resistor is inserted between GND and GND_{CORE}. The voltage drop of the created circuit has now the desired proportionality to the actual current, since $u(t) = R \cdot i(t)$. As shunt resistor we use 18 Ω for all trace acquisitions.

The clock signal is generated by the microcontroller at a frequency of 125 kHz. As trigger signal for the oscilloscope, we choose a falling edge on pin 1, which is $ready\_for\_iv$. This indicates that the IV has been transmitted correctly and the initialization of the stream cipher begins.

6.2 Communication Sequence

For every trace generation basically five steps are passed. An overview is given in Figure 6.5.

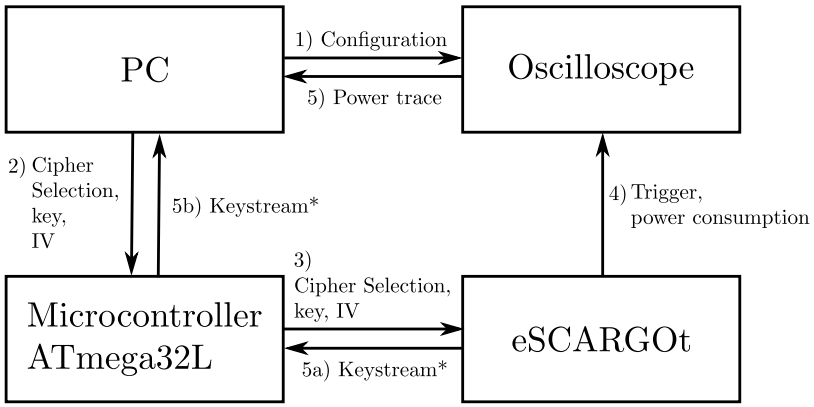


Figure 6.5: Communication of the components used for gaining power traces

1. The PC establishes a VISA session and transmits the configuration data to the oscilloscope. After this, the oscilloscope is armed and waits for the trigger signal to start the measurement.
2. Now, the serial port between PC and microcontroller is opened to transfer the cipher selection, a predefined key and a randomly generated IV. Both, the key and the IV are given in hexadecimal notation. To reduce the noise during the measurement, the serial port is closed afterwards.
3. The microcontroller selects the cipher by applying the 4-bit number to the input pins $cipher[0]$ to $cipher[3]$ and resetting the eSCARGOt for at least one clock cycle. When the pin $ready\_for\_key$ is asserted high, the microcontroller sends the key, after the conversion to binary notation, to the eSCARGOt. The same procedure is done for the IV.
4. Before the eSCARGOt starts with the initialization, it releases the trigger signal with the falling edge of the pin $ready\_for\_iv$ and triggers the oscilloscope to measure.
5. After the measurement, the digitalized trace is transmitted to the PC, where it is saved in one file together with the key and IV used in Step 2. Optionally, the keystream generated by the eSCARGOt is sent via the microcontroller (a) to the PC (b).

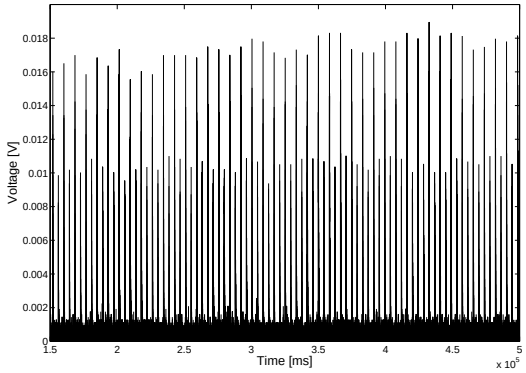


Figure 6.6: Enlarged detail of a measured example trace (TRIVIUM initialization).

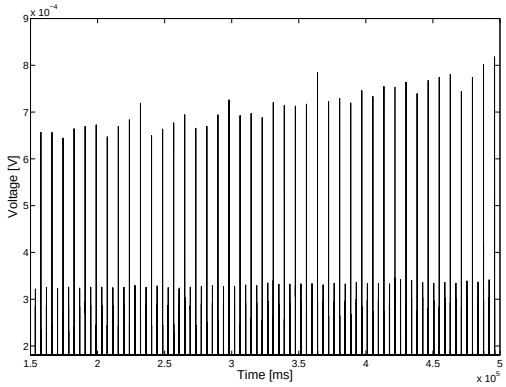


Figure 6.7: Variance of 100 traces (TRIVIUM initialization).

6.3 Preprocessing of Measured Traces

If the six steps are finished successfully, we get much more information as we need. Figure 6.6 shows an example trace of the initialization of TRIVIUM. Mounting a DPA on complete traces would lead to a huge amount of computations due to the inclusion of nonrelevant values. For efficiency reasons, we concentrate only on the important part of the traces.

With a sample rate of 1 GS/s and a time window of 4 ms, every trace consists of 4,000,000 measuring points compared to 160 steps of the initialization of GRAIN and 1152 steps of the initialization of TRIVIUM, respectively. Hence, we have to differentiate which of these points carry the most important information.

The trace can be divided into three parts, namely two types of peaks and the space in-between. When taking the clock signal into account, the higher peaks can be allocated to the rising edges and the lower peaks to the falling edges. To compress the trace to the bare minimum, it is necessary to detect which parts of the traces contains the most important information. In our case this is the dynamic power consumption depending on the processed data. By calculating the variance of several traces, we can infer that the peaks of the rising edges are more distributed than the peaks of the falling edges (see Figure 6.7), which indicates a higher information content at the rising edges. For this reason, we decided to use only these peaks for the power analysis.

In order to extract the peaks, there are mainly two common possibilities that can be applied:

- After defining an appropriate threshold, the power consumption values between

positively crossing and negatively crossing the threshold are summed up.

- Only the highest value of a peak is considered.

For this compression we adapt an existing Matlab script to our measurement, which makes use of the second method. The algorithm can be described as follows:

1. Starting at point t, find the highest value in a predefined window size e around t and save this peak.
2. From this peak, skip n values and set this point to t. Continue with Point 1.

Before this algorithm begins, the starting point t, the window size e, and the step size n have to be assessed manually. At a clock frequency of 125 kHz, we expect a rising edge every 0.008 ms. Considering a sample rate of 1 GS/s, the step size n can be estimated to 8000. The result of the peak extraction of our example trace is given in Figure 6.8.

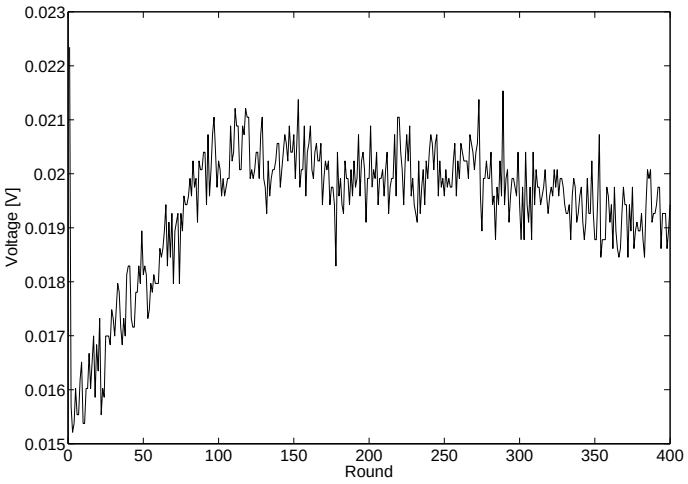


Figure 6.8: Extracted peaks of an example trace (Trivium initialization)

7 Side Channel Analysis Attacks on Grain

Grain was designed by Martin Hell, Thomas Johansson, and Willi Meier. In this chapter, we describe the second version, denoted as version 1 [HJM05]. After submitting an initial version to eSTREAM, smaller changes were made due to security flaws. In the following sections we examine the stream cipher Grain with respect to side channel vulnerabilities and present the theoretical background of our implemented DPA attack. We close this chapter with practical results of the attack on the eSCARGOt ASIC in Section 7.5.3.

7.1 Design Specification of Grain

The stream cipher Grain is based on two 80-bit shift registers: A non-linear feedback shift register (NLFSR), denoted as $b_i, b_{i+1}, \ldots, b_{i+79}$, and a linear feedback shift register (LFSR), denoted as $s_i, s_{i+1}, \ldots, s_{i+79}$. These registers are coupled via a non-linear function H.

Before initialization, an 80-bit key is loaded into the NLFSR and a 64-bit IV into the LFSR. The remaining 16 bits of the LFSR are filled with ones to avoid a weak always-zero state.

The algorithm is illustrated in Figure 7.1. During initialization ($\delta = 1$), no keystream is generated and the output function z_i is XORed with the input of the feedback registers. Hence, the update functions b_{i+80} and s_{i+80} are defined as

$$\begin{aligned} b_{i+80} &= s_i \oplus g_i \oplus z_i \cdot \delta \text{ and} \\ s_{i+80} &= f_i \oplus z_i \cdot \delta, \end{aligned}$$

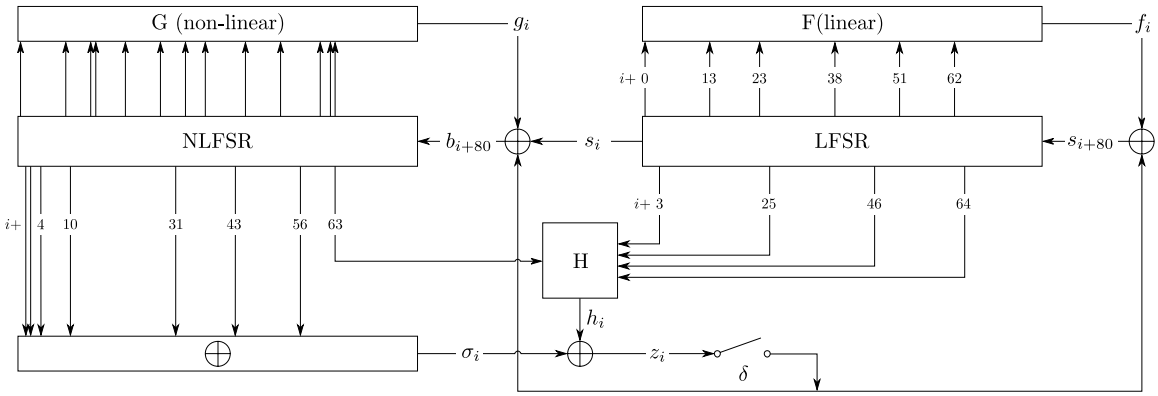


Figure 7.1: GRAIN (based on [FGKV07]).

with

$$\begin{aligned} g_i &= b_{i+62} \oplus b_{i+60} \oplus b_{i+52} \oplus b_{i+45} \oplus b_{i+37} \oplus b_{i+33} \oplus b_{i+28} \oplus b_{i+21} \\ &\oplus b_{i+14} \oplus b_{i+9} \oplus b_i \oplus b_{i+63}b_{i+60} \oplus b_{i+37}b_{i+33} \oplus b_{i+15}b_{i+9} \\ &\oplus b_{i+60}b_{i+52}b_{i+45} \oplus b_{i+33}b_{i+28}b_{i+21} \oplus b_{i+63}b_{i+45}b_{i+28}b_{i+9} \\ &\oplus b_{i+60}b_{i+52}b_{i+37}b_{i+33} \oplus b_{i+63}b_{i+60}b_{i+21}b_{i+15} \\ &\oplus b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} \oplus b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} \\ &\oplus b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}, \end{aligned}$$

$$f_i = s_{i+62} \oplus s_{i+51} \oplus s_{i+38} \oplus s_{i+23} \oplus s_{i+13} \oplus s_i.$$

The output function z_i and the non-linear function h_i are defined as

$$\begin{aligned} z_i &= \sigma_i \oplus h_i \\ &= b_{i+1} \oplus b_{i+2} \oplus b_{i+4} \oplus b_{i+10} \oplus b_{i+31} \oplus b_{i+43} \oplus b_{i+56} \oplus h_i, \end{aligned}$$

$$\begin{aligned} h_i &= s_{i+25} \oplus b_{i+63} \oplus s_{i+3}s_{i+64} \oplus s_{i+46}s_{i+64} \oplus s_{i+64}b_{i+63} \\ &\oplus s_{i+3}s_{i+25}s_{i+46} \oplus s_{i+3}s_{i+46}s_{i+64} \oplus s_{i+3}s_{i+46}b_{i+63} \\ &\oplus s_{i+25}s_{i+46}b_{i+63} \oplus s_{i+46}s_{i+64}b_{i+63}. \end{aligned}$$

The initialization is finished after 160 of these rounds ($\delta = 0$), and the keystream generation begins.

7.2 Adversary Model

In this thesis, we assess the following adversary model: The adversary has knowledge about the IVs. A strong adversary model would allow her to choose the IVs, but this is not necessary in our case. We suppose that the adversary has the possibility to get

hold of the encryption device allowing any exhaustive analysis during the keystream generation.

7.3 Timing Analysis

To apply a timing analysis attack, a dependency of the data to the timing behavior has to be given, e.g., if the cipher includes conditional branches or table look-ups. In the case of Grain, this is not the case. Also, any straightforward hardware implementation of this stream cipher should be constant in time, regardless of the internal state, and hence, prevent a timing analysis attack.

7.4 Simple Power Analysis

In [GBC+08], the authors evaluated Grain with respect to possible side-channel vulnerabilities. They figured out that the bit-serialized implementation makes it possible to mount an SPA attack in theory. Therefore, the registers have to be reset to a defined state before starting the key setup. By measuring the Hamming distances of subsequent key bits, the adversary is able to recover the whole key.

7.5 Differential Power Analysis

In this section, we describe our DPA attack on Grain. To develop this attack, we started by attacking ideal simulated traces. For this reason, we introduce our assumed power model.

7.5.1 Power Model

A DPA requires an adversary to map hypothetical intermediate values to hypothetical power consumptions. Therefore, the power consumption of the analyzed device has to be modeled in a suitable manner. The main source of leakage of a hardware implementation is the dynamic power consumption resulting from switching CMOS flip-flops. In Grain these are used to implement the required shift registers. As already figured out in Section 3.2, we can assume that

$$P_{FF}(0,1) = P_{FF}(1,0) >> P_{FF}(0,0) = P_{FF}(1,1) \approx 0,$$

where $P_{FF}(x,y)$ describes the power consumption of a flip-flop changing its state from x to y. Based on this property, the Hamming distance model seems to be an

appropriate choice to simulate the power consumption. As figured out in 4.3.1, the Hamming distance model simulates the power consumption in a digital circuit by counting the number of transitions of register bits in a certain time interval. Hence, the hypothetical power consumption for each clock cycle i is modeled by

$$\begin{aligned} P^i_{hyp} &= \sum P_{FF}(0,1) + \sum P_{FF}(1,0) \\ &= \sum_{j=i}^{i+79} ((b_j \oplus b_{j-1}) + (s_j \oplus s_{j-1})). \end{aligned}$$

7.5.2 Theoretical Approach

Our proposed attack is divided into 4 steps. The first two steps allow an extraction of 17 key bits. The remaining 63 key bits can then be recovered in a subsequent step that involves solving algebraic equations with respect to extracted side channel information.

Step 1: Round 1 to 16. Our attack begins similar to the first two steps of the attack by Fischer *et al.* described in [FGKV07]. Basically, we perform a correlation DPA attack for every round to extract key bits from the power consumption of the first 16 rounds. Beginning with round $i = 1$, we work through the following points that complies to the description of a DPA in Section 4.3.3:

1. As intermediate result, we choose to predict the states of the internal NLFSR and LFSR. These states depend on the feedback bits b_{i+80} and s_{i+80}. Because initially the state of the LFSR is completely known, b_{i+80} and s_{i+80} can be written as functions u and v of known variables and unknown variables that have to be determined by the adversary:

$$\begin{aligned} b_{i+80} &= u(\underbrace{s_i, s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}}_{known}, \underbrace{g_i, b_{i+63}, \sigma_i}_{unknown}), \\ s_{i+80} &= v(\underbrace{s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, f_i}_{known}, \underbrace{b_{i+63}, \sigma_i}_{unknown}). \end{aligned}$$

   Compared to Section 4.3.3, this complies with the known non-constant data d and the unknown key k.
2. Using several random values for the known part, which are actually bits of the initialization vector, we start a series of measurements. Note that these measurements only have to be recorded once for all rounds.

Algorithm 1: Step 1 of the DPA of Grain

Input : IV : initialization vector, $T(i)$: vector containing all values of round i of measured power traces, n : number of IVs, m : number of hypotheses

Output: $\sigma_1, \ldots, \sigma_{16}$, $b_{64}, \ldots, b_{79}$, and $g_1, \ldots, g_{16}$

```
1  begin
2      b_80 ← 0
3      for i = 1 to 16 do
4          for j = 1 to n do
5              hyp ← 1
6              for all hypotheses (σ_i, b_{i+63}, g_i) do
7                  Using IV_j and known values of rounds 0 to i − 1, compute state of
                   NLFSR and LFSR of round i and
8                  HD_hyp(j) = HD of last i bits of NLFSR + HD of LFSR
9                  hyp ← hyp + 1
10             end
11         end
12         for l = 1 to m do
13             Compute correlation coefficient corr_l(HD_l, T(i))
14         end
15         Accept hypothesis with highest correlation coefficient
16     end
17 end
```

3. The intermediate results are computed for all possible hypotheses for (g_1, b_{64}, σ_1) and the used random values for $(s_1, s_4, s_{26}, s_{47}, s_{65})$.
4. As most of the NLFSR bits b_i are independent of the used IVs for the first rounds, their contribution to the Hamming distance can be omitted. This does not effect our result as constant offsets will not effect the correlation coefficient anyway. Hence, it is only necessary to consider the Hamming distances of the LFSR and the IV-dependent part of the NLFSR. To estimate the Hamming distance between b_{80} and b_{81}, we make the arbitrary assumption that $b_{80} = 0$.
5. Comparing the Hamming distances with the power traces results to the correct values for (g_1, b_{64}, σ_1).

These five points are repeated for the rounds 2 to 16 to obtain the values $\sigma_2, \ldots, \sigma_{16}$, $b_{65}, \ldots, b_{79}$, and $g_2, \ldots, g_{16}$. Remember that all these values still depend on the initial assumption of b_{80}, which will be validated to keep or correct the extracted information during the next step. The whole procedure of Step 1 is summarized in Algorithm 1.

Step 2: Round 17. The arbitrary assumption of $b_{80} = 0$ in Step 1, is verified in Step 2 of the attack. For this reason, round 17 is analyzed twice. In both runs, hypotheses for the tuple (σ_{17}, g_{17}) are tested. First, the assumption $b_{80} = 0$ is retained and the result with the highest correlation coefficient is stored. Then, b_{80} is set to 1 and the analysis is repeated. Because b_{80} effects all previously calculated values of g_i, these values have to be inverted for this second calculation of the Hamming distances. Depending on the correctness of the initial assumption, either the first or the second run produces a wrong output of the function H. Again, the highest correlation coefficient allows to determine the correct result and to invert the previously extracted values if necessary.

Step 3: Round 18 to 62. So far the considered values of g_i were unique and independent of the respective IVs. Every bit of the NLFSR used to compute g_i has been one of the initial key bits. This changes in round 18. From this round on, the input bit b_{i+63} of the non-linear function G is given as

$$b_{i+63} = g_{i-17} \oplus \sigma_{i-17} \oplus h_{i-17} \oplus s_{i-17}, \text{ for } i \geq 18.$$

As a consequence, not all input bits of G are IV-independent constants anymore.

The following procedure varies with respect to the amount of input bits of the function G that depend on the IV. An overview of the known and unknown bits of the NLFSR at round 18 is given in Figure 7.2.

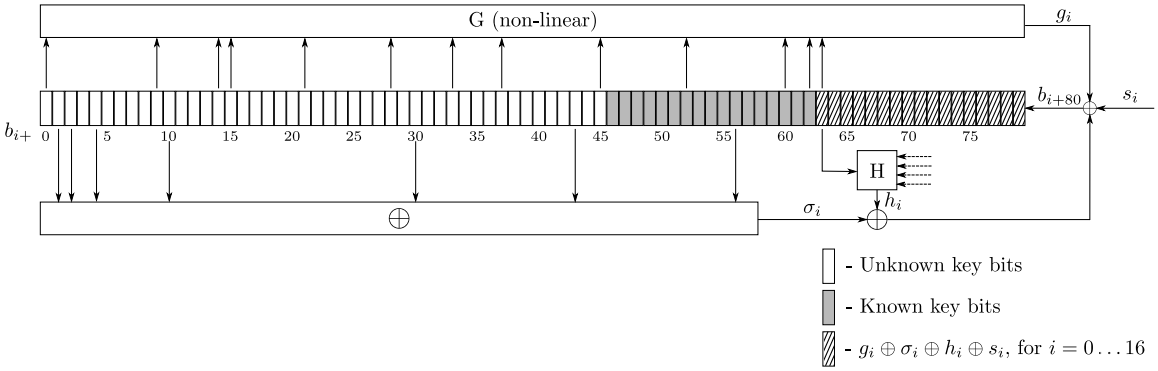


Figure 7.2: State of NLFSR at round 18: Overview of revealed key bits.

The graphic illustrates that 9 of 13 input bits of the function G are unknown key bits, while 3 bits have been disclosed in the previous steps and 1 bit depends on the IV. A closer look at the function G shows that the non-linear part of the function does not include all 13 input bits. Instead, a few bits, b_{i+62}, b_{i+14}, and b_i, are only XORed and do not appear in any AND operation. We will summarize these bits in a

new variable denoted as $gconst_i$. Furthermore, all summands independent of the IV are also included in $gconst_i$. This leads to a much simpler equation for g_i. To give an example, in round 18,

$$g_i = gconst_i \oplus b_{i+63}b_{i+60} \oplus b_{i+63}b_{i+45}b_{i+28}b_{i+9} \\ \oplus b_{i+63}b_{i+60}b_{i+21}b_{i+15} \oplus b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37}, \text{ for } i = 18.$$

Now hypotheses for the variables $(\sigma_i, b_{i+9}, b_{i+15}, b_{i+21}, b_{i+28}, b_{i+37}, b_{i+45}, gconst_i)$ can be used to model a power consumption and to discriminate valid guesses by means of the correlation coefficient. Remember that the values of b_{i+60} and b_{i+52} are already known from steps 1 and 2 and therefore do not have to be guessed (compare Figure 7.2). Figure 7.3 depicts the correlation coefficient in Round 18 of an attack on simulated ideal traces[1]. Note that although there can obviously only be one correct guess, there are several hypotheses sharing the highest correlation coefficient. This is due to the fact that all these guesses lead to the same leakage.

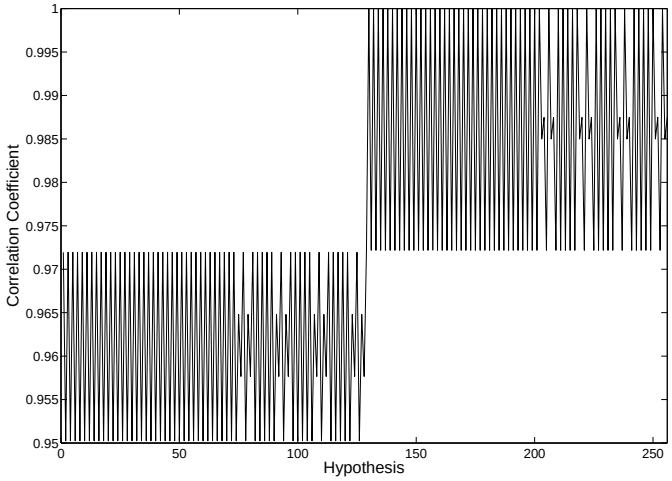


Figure 7.3: Correlation coefficients of hypothetical and simulated power consumption of round 18.

Note that the device only leaks information due to changes of the Hamming distance caused by b_{i+80} and s_{i+80}. The value of σ_i linearly effect both of these bits, while all other guessed values only contribute to b_{i+80}. Therefore, a false guess of σ_i cannot be compensated by the other values. In addition, σ_i effects the correlation coefficient with a doubled weight compared to the other variables. Hence, the correct choice of σ_i can still be discriminated with high accuracy as suggested by the simulation results shown.

[1]The simulated traces are calculated from the Hamming distances using the power model.

Another challenge concerning σ_i is that from Round 25 on, it also depends on at least one IV-dependent input. To circumvent this problem, only the unknown constant key bits are considered in our hypotheses to get a unique value for σ_i. We can redefine the result as

$$\sigma_i^* = \begin{cases} b_{i+1} \oplus b_{i+2} \oplus b_{i+4} \oplus b_{i+10} \oplus b_{i+31} \oplus b_{i+43} \oplus b_{i+56}, & \text{for } i = 1, \ldots, 7, \\ b_{i+1} \oplus b_{i+2} \oplus b_{i+4} \oplus b_{i+10} \oplus b_{i+31} \oplus b_{i+43}, & \text{for } i = 8, \ldots, 20, \\ b_{i+1} \oplus b_{i+2} \oplus b_{i+4} \oplus b_{i+10} \oplus b_{i+31}, & \text{for } i = 21, \ldots, 32, \\ b_{i+1} \oplus b_{i+2} \oplus b_{i+4} \oplus b_{i+10}, & \text{for } i = 33, \ldots, 53, \\ b_{i+1} \oplus b_{i+2} \oplus b_{i+4} & \text{for } i = 54, \ldots, 59, \\ b_{i+1} \oplus b_{i+2} & \text{for } i = 60, 61, \\ b_{i+1} & \text{for } i = 62. \end{cases}$$

The method illustrated for round 18 can also be used for subsequent rounds. In every round where the dependency of g_i on the input bits changes, the respective equation has to be updated. The number of bits that have to be hypothesized decreases with preceding shifting of the known bits to the left side of the NLFSR.

The last round where this method is applied is round 62. Here the hypothesis consists of only two bits, σ_i^* and $gconst_i$.

Step 4: Key computations. Having extracted the values $\sigma_1^*, \ldots, \sigma_{62}^*$ from the power traces, $k_2, \ldots, k_{63}$ can be calculated by solving the above-mentioned set of equations for σ_i^* starting with b_{63}. The key bits $k_{64}, \ldots, k_{80}$ are already known from Step 1 and 2. Note that k_i is equivalent to the corresponding b_i. Therefore, the extraction of b_i reveals the correct key.

7.5.3 Results

The attack proposed in Section 7.5.2 succeeded with as much as 2600 power traces. The correlation coefficients of right and wrong guesses are very close to each other, such that less power traces lead to false detections. This bad separation of correlation coefficients results from the corresponding leakage: In each round, the wrong and right guesses differ only by the 2 values of the calculated feedback bits b_{i+80} and s_{i+80}. The overall leakage of the device is the leakage of all 160 register bits plus noise and contributions of feedback function's logic. Therefore, although the correlation coefficient is as high as 0.64, there are still only small differences that can be used to discriminate.

In addition, there is also a leakage from the internal circuitry that counts the number of processed rounds. This source of leakage can be identified when analyzing

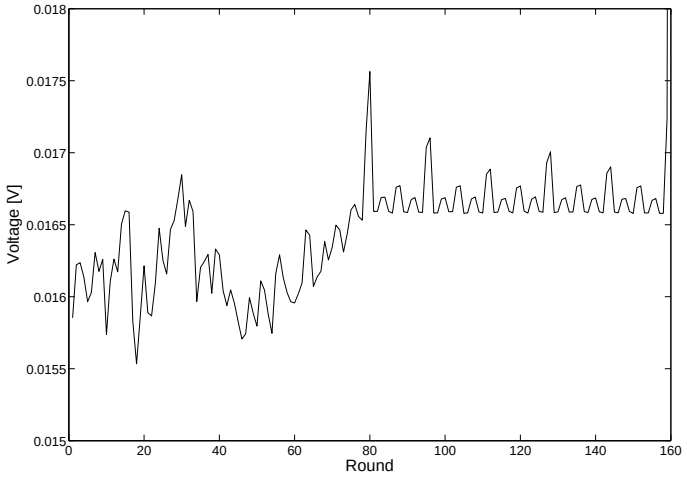


Figure 7.4: Average power consumption of the initialization of Grain.

the average power consumption of all 2600 traces (see also Figure 7.4): Once the initial state evolved to a pseudo random state, a periodical leakage pattern can be identified. This counter or state machine starts when delivering the key into the NLFSR and counts thenceforward 320 rounds until the initialization has finished, including 80 rounds to load the key, 64 rounds to load the IV, 16 rounds to fill the remaining bits of the LFSR with ones and finally 160 rounds of the actual initialization process. This will not have any effect on our results, because the leakage of this counter can be treated as a constant offset for every point of a measurement. Nevertheless, this observation indicates that attacks exploiting multiple points of the traces, e.g., higher order attacks, require a more sophisticated power model than the Hamming distance model.

8 Side Channel Analysis Attacks on Trivium

The second stream cipher we attack is Trivium, which was submitted to eSTREAM by Christophe De Cannière and Bart Preneel. After a short description, we focus on the DPA. For this attack, we use the same adversary model as for the attack on Grain. Finally, we present the result of the DPA of the eSCARGOt implementation.

8.1 Design Specification of Trivium

Trivium makes use of an 80-bit secret key and an 80-bit IV. The algorithm is realized by three circularly arranged NLFSRs of different lengths (93, 84 and 111 bits). These registers' values are referred to as $s_1, ..., s_{93}$ for the first, $s_{94}, ..., s_{177}$ for the second, and $s_{178}, ..., s_{288}$ for the third register.

The algorithm starts by writing the key and the IV to the leading 80 bits of the first and the second register, respectively. All remaining bits are filled with zeros, except for s_{286}, s_{287}, and s_{288}, which are set to one. After loading the key and the IV, the initialization process is executed for $4 \cdot 288$ clock cycles. Once the initialization has finished, the algorithm outputs its keystream. The setup and initialization process is illustrated by the following pseudo code [CP06]:

$(s_1, s_2, \ldots, s_{93}) \leftarrow (k_1, \ldots, k_{80}, 0, \ldots, 0)$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (iv_1, \ldots, iv_{80}, 0, \ldots, 0)$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

for $i = 1$ to $4 \cdot 288$ **do**
  $t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$
  $t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$
  $t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end

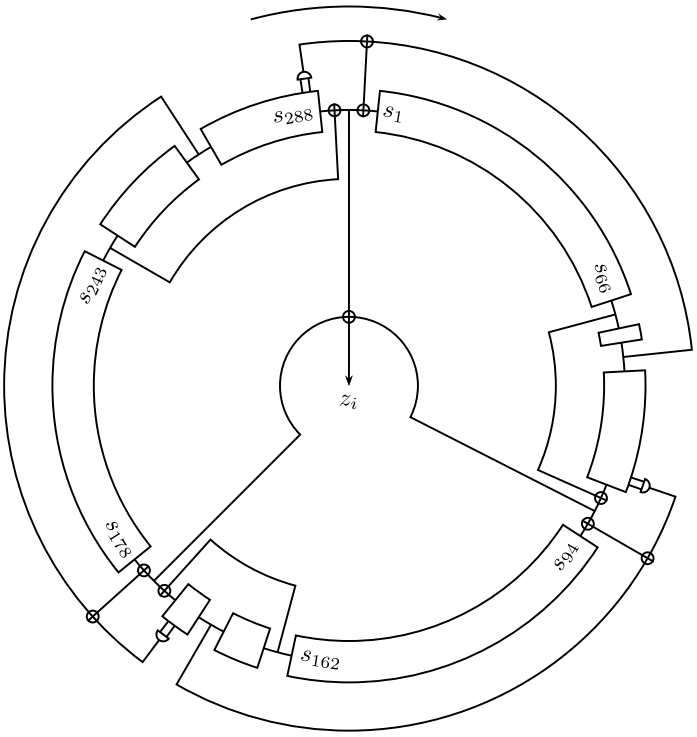


Figure 8.1: Keystream generation of Trivium [CP06].

The keystream generation can be described in a very similar way. Here, the intermediate results of t_1, t_2, and t_3 are combined by a logical XOR function to generate the output bits z_i. The structure of Trivium can be found in Figure 8.1, the algorithm for the key generation is given as follows [CP06]:

for $i = 1$ to n **do**
$t_1 \leftarrow s_{66} + s_{93}$
$t_2 \leftarrow s_{162} + s_{177}$
$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end

8.2 Timing Analysis and Simple Power Analysis

Concerning the data-dependent timing behavior and the susceptibility to simple power analysis, TRIVIUM has exactly the same characteristics as GRAIN. There are no conditional branches or table look-ups, which could be exploited for a timing analysis, but the key setup allows a theoretical approach for a simple power analysis.

8.3 Differential Power Analysis

Again, we will begin with the power model. After describing the attack in detail, we will present our result by testing it on the eSCARGOt chip.

8.3.1 Power Model

As done in GRAIN, we also choose the Hamming distance model for the same reason. Due to the simple construction of TRIVIUM, the power model can be set to

$$P^i_{hyp} = \sum_{j=1}^{288} (s_j(i) \oplus s_j(i-1)),$$

where again P^i_{hyp} denotes the hypothetical power consumption of round i.

8.3.2 Theoretical Approach

The attack is an alternating capturing of key bits directly and capturing variables for non-linear equations that can be solved to disclose the key bits. Again, our attack uses a correlation DPA step for each single round of the initialization. This way, the the number of unknown values that have to be considered to predict hypothetical intermediate values can be kept very small.

For the first round, the state of the registers is given as follows:

$$\begin{aligned}(s_1, s_2, \ldots, s_{93}) &\leftarrow (k_{69}, k_1, k_2, k_3, \ldots)\\ (s_{94}, s_{95}, \ldots, s_{177}) &\leftarrow (k_{66} + iv_{78}, iv_1, iv_2, iv_3, \ldots)\\ (s_{178}, s_{179}, \ldots, s_{288}) &\leftarrow (iv_{69}, 0, 0, 0, \ldots)\end{aligned}$$

Since the first register is completely independent of the IVs, it has no influence to the correlation coefficient and can be neglected during the computation of the Hamming distances. This property does not change until round 67.

round	key bits revealed
1 - 12	$k_{66}, \ldots, k_{55}$
13 - 27	$\sigma_{13}, \ldots, \sigma_{27}$
28 - 37	$k_{39}, \ldots, k_{30}$
38 - 54	$\sigma_{38}, \ldots, \sigma_{54}$
55 - 62	$k_{12}, \ldots, k_5$
63 - 66	$\sigma_{63}, \ldots, \sigma_{66}$
67 - 73	$k_4, \ldots, k_1, k_{69}, \ldots, k_{67}, \sigma_{67}, \ldots, \sigma_{73}$
74 - 78	$\sigma_{74}, \ldots, \sigma_{78}$

Table 8.1: Alternating disclosure of keys and key terms.

In round 1, all values of the second and third register are known with k_{66} being the only exception. Calculating two hypothetical power consumptions using both values $k_{66} = 0$ and $k_{66} = 1$, the correct key bit can be determined by means of the correlation coefficient. The result can be used to calculate the internal state of the next round of the cipher, in which k_{65} is the only unknown bit. Following this strategy, the 12 key bits $k_{66}, \ldots, k_{55}$ are revealed.

Instead of key bits, in rounds 13 to 27, values of combinations of key bits are extracted from the side channel information. For example, in round 13 the leading bit of the second register can be calculated by the term:

$$s_{94} = k_{54} \oplus k_{79} \cdot k_{80} \oplus iv_{66}.$$

Because none of these key bits are already disclosed, the used hypotheses have to target the combination of these bits. For this reason, σ_i is introduced as

$$\sigma_i = s_{66} \oplus s_{91} \cdot s_{92} \oplus s_{93},$$

which in the currently discussed round 13 would be:

$$\sigma_{13} = k_{54} \oplus k_{79} \cdot k_{80}.$$

Sticking to this technique, key bits and sigmas can alternately be extracted. The schedule of this attack is given in Table 8.1. The only exception to this pattern occurs in rounds 67 to 73. Here, the first register contains an IV-dependent part, which has to be taken into account when determining the Hamming distances. In round 67, the leading bits of this register are

$$(s_1, s_2, \ldots, s_{93}) \leftarrow (iv_{69} \oplus k_3, k_4, k_5, k_6, \ldots).$$

As a consequence, two bits have to be hypothesized, k_3 from the first and σ_{67} from the second register. Computing the Hamming distance of the leading bit of register 1 needs an assumption for k_4. In this attack $k_4 = 0$ is chosen.

The processing of rounds 68 to 72 is now straightforward and round 73 can be used to prove the initial assumption made for k_4:

$$\begin{array}{lcl}(s_1, s_2, \ldots) & \leftarrow & (iv_{63} \oplus iv_{76} \cdot iv_{77} \oplus iv_{78} \oplus k_{66}, iv_{64} \oplus iv_{77} \cdot iv_{78} + \oplus iv_{79} \oplus k_{67}, \ldots) \\ (s_{94}, s_{95}, \ldots) & \leftarrow & (k_{63} \oplus k_{19} \cdot k_{20} \oplus k_{21} \oplus iv_6, k_{64} \oplus k_{20} \cdot k_{21} \oplus k_{22} \oplus iv_7, \ldots) \\ (s_{178}, s_{179}, \ldots) & \leftarrow & (k_{63} \oplus iv_{75} \oplus iv_{10} \cdot iv_{11} \oplus iv_{12}, k_{64} \oplus iv_{76} \oplus iv_{11} \cdot iv_{12} \oplus iv_{13}, \ldots).\end{array}$$

By reevaluating the value of k_{66}, the result can be compared to the initially recovered value from round 1. If the recovered values do not match, all key bits revealed from the moment of our assumption have to be inverted to get back to a consistent solution. The remaining key bits can be obtained using the equations of σ_i:

for $i = 25$ to 27 and 52 to 54 **do**
  $k_{67-i} = \sigma_i + k_{92-i} \cdot k_{93-i} + k_{94-i}$
end
for $i = 78$ to 70 **do**
  $k_{94-i} = \sigma_i + k_{92-i} \cdot k_{93-i} + k_{136-i}$
end
for $i = 69$ to 68 **do**
  $k_{94-i} = \sigma_i + k_{92-i} \cdot k_{93-i} + k_{136-i} + 1$
end
$k_{27} = \sigma_{67} + k_{25} \cdot k_{26} + k_{69}$
for $i = 66$ to 65, 51 to 40, and 24 to 14 **do**
  $k_{94-i} = \sigma_i + k_{92-i} \cdot k_{93-i} + k_{67-i}$
end

Improvement of the Attack

In the previous section, we proposed an attack that analyzes each round separately. Therefore, in all rounds except of rounds 67 to 73, only one unknown bit has to be hypothesized.

A more sophisticated approach exploits that each feedback bit not only effects the Hamming distance of the round where it is generated, but also the leakage of subsequent rounds. In this improvement, two key bits, k_i and k_{i+1}, are revealed by attacking round $i + 3$. In the first 66 rounds, this means that 2^4 hypotheses are built for all possible values of four bits. A correct guess of the first two key bits can result in a deviation of the Hamming distance of no more than 2 bits, independent of the

hypotheses for the remaining two bits. Instead, a wrong key guess of k_i and k_{i+1} can be off by a distance of up to 4. This leads to an improvement of discrimination of correct and wrong key hypotheses using the correlation coefficient compared to the previously introduced attack.

8.3.3 Results

Figure 8.2 shows the average power consumption of the eSCARGOt during the initialization of Trivium. As already seen in Grain, the power consumption pattern of a counter can easily be identified, indicating the need of sophisticated power models for higher order attacks. Without this counter, the power trace should be at a nearly constant level from round 159 on. This is the first time where all bits of the registers depend on the IV. The increasing power consumption at the beginning can be explained by the initial state of the registers. Especially responsible is the third register, which contains 108 zeros that are arranged in a consecutive order.

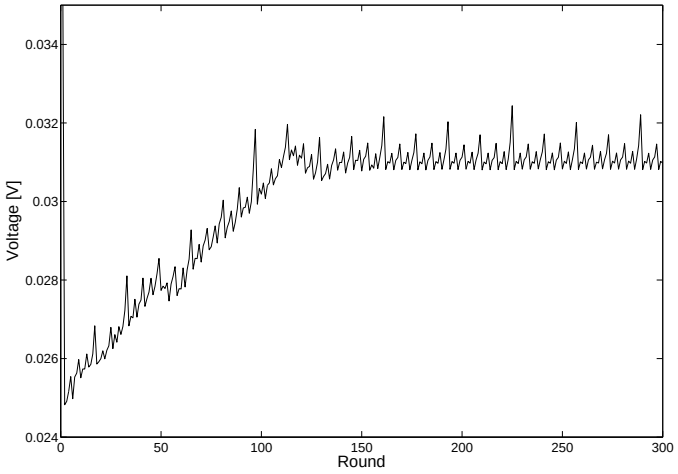


Figure 8.2: Average power consumption of the initialization of Trivium.

The high peak of round 1 interferes with the falling edge of the external control signal *ready_for_iv*, provided by the *eSCARGOt* chip to indicate the beginning of the initialization process. Our improved attack proposed in Section 8.3.2 allows to neglect this peak since it starts by analyzing the leakage of round 4. The attack using 550 power traces was suitable to reveal all 80 bits of the key.

9 Summary

In this thesis, we evaluated practical side channel attacks on hardware implementations of stream ciphers. While demonstrating two practical attacks on hardware implementations of the two stream ciphers Trivium and Grain, we highlight the additional efforts necessary to mount DPAs on stream ciphers compared to the commonly used methods established for block ciphers. We practically showed that in DPA, methods can be adapted to stream ciphers, but also that a direct discrimination of the key is not always possible. We overcame these challenges by introducing new variables for intermediate states that accumulate to algebraic systems of equations. Our work also illustrates that instead of gathering a huge set of equations, it is very feasible to randomly assume unpredictable values and to correct results afterwards in case of inconsistencies. This thesis is one of the first works reporting on practical results of DPA attacks on stream ciphers and the first to provide results using the *eSCARGOt* evaluation ASIC. Also the use of side channel analysis results in algebraic systems is not a common practice in the field of side channel analysis yet.

9.1 Future Work

This work indicates a need for further investigations of side channel analysis attacks on stream ciphers to develop methods that work as efficient as their counterparts on block ciphers. Especially concerning hardware implementations, our results have shown that for successful higher order attacks a more sophisticated power model is required. The possibility of DPA attacks on stream ciphers also raises the need for proposals of countermeasures that are applicable to stream ciphers.

Bibliography

[AK96] Ross Anderson and Markus Kuhn. Tamper Resistance - A Cautionary Note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, 1996.

[BECN+04] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. Cryptology ePrint Archive, Report 2004/100, 2004. `http://eprint.iacr.org/`.

[BK02] Régis Bevan and Erik Knudsen. Ways to Enhance Differential Power Analysis. In *ICISC*, pages 327–342, 2002.

[BLW03] Bert den Boer, Kerstin Lemke, and Guntram Wicke. A DPA Attack against the Modular Reduction within a CRT Implementation of RSA. In *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 228–243, London, UK, 2003. Springer-Verlag.

[BS96] Eli Biham and Adi Shamir. A New Cryptanalytic Attack on DES. preprint, 1996.

[CB95] Anantha P. Chandrakasan and Robert W. Brodersen. Minimizing Power Consumption in CMOS Circuits, 1995. `http://bwrc.eecs.berkeley.edu/php/pubs/pubs.php/418.html`.

[CP06] Christophe De Cannière and Bart Preneel. Trivium Specifications. *eSTREAM, ECRYPT Stream Cipher Project*, 2006.

[DKL+00] Jean-François Dhem, François Koeune, Philippe-Alexandre Leroux, Patrick Mestré, Jean-Jacques Quisquater, and Jean-Louis Willems. A Practical Implementation of the Timing Attack. In *CARDIS '98: Proceedings of the The International Conference on Smart Card Research and Applications*, pages 167–182, London, UK, 2000. Springer-Verlag.

[ECRa] ECRYPT. eSTREAM Portfolio. eSTREAM, ECRYPT Stream Cipher Project. `http://www.ecrypt.eu.org/stream/finallist.html`.

[ECRb] ECRYPT. Network of Excellence in Cryptology. `http://www.ecrypt.eu.org/ecrypt1/`.

[EKM+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoqCode Hopping Scheme. In *CRYPTO*, pages 203–220, 2008.

[FGKV07] W. Fischer, B. M. Gammel, O. Kniffler, and J. Velten. Differential Power Analysis of Stream Ciphers. In *of Lecture Notes in Computer Science*, pages 257–270. Springer, 2007.

[GB08] Tim Good and M. Benaissa. European Stream Ciphers Are Ready to GO, 2008. `http://www.shef.ac.uk/eee/escargot/`.

[GBC+08] Benedikt Gierlichs, Lejla Batina, Christophe Clavier, Thomas Eisenbarth, Aline Gouget, Helena Handschuh, Timo Kasper, Kerstin Lemke-Rust, Stefan Mangard, Amir Moradi, and Elisabeth Oswald. Susceptibility of eSTREAM Candidates towards Side Channel Analysis. In *The State of the Art of Stream Ciphers - SASC 2008*. Lausanne, Switzerland, February 2008.

[HJ08] Martin Hell and Thomas Johansson. Breaking the F-FCSR-H Stream Cipher in Real Time. In *ASIACRYPT '08: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, pages 557–569, Berlin, Heidelberg, 2008. Springer-Verlag.

[HJM05] Martin Hell, Thomas Johansson, and Willi Meier. Grain - A Stream Cipher for Constrained Environments. *eSTREAM, ECRYPT Stream Cipher Project*, 2005.

[Jaf07] Josh Jaffe. A First-Order DPA Attack Against AES in Counter Mode with Unknown Initial Counter. In *CHES '07: Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems*, pages 1–13, Berlin, Heidelberg, 2007. Springer-Verlag.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 388–397, London, UK, 1999. Springer-Verlag.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, pages 104–113, London, UK, 1996. Springer-Verlag.

[LM05] Richard J. Larsen and Morris L. Marx. *An Introduction to Mathematical Statistics and Its Applications*. Prentice Hall, 4th edition, 2005.

[LMPV04] Joseph Lano, Nele Mentens, Bart Preneel, and Ingrid Verbauwhede.

Power Analysis of Synchronous Stream Ciphers with Resynchronization Mechanism. In *SASC 2004 - The State of the Art of Stream Ciphers*, Workshop Record, pages 327–333, 2004. `http://www.ecrypt.eu.org/stvl/sasc/record.html`.

[LSP04] Kerstin Lemke, Kai Schramm, and Christof Paar. DPA on n-Bit Sized Boolean and Arithmetic Operations and Its Application to IDEA, RC6, and the HMAC-Construction. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 205–219. Springer-Verlag, 2004.

[MPO] Stefan Mangard, Thomas Popp, and Elisabeth Oswald. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer-Verlag.

[OGOP04] Siddika Berna Örs, Frank Gürkaynak, Elisabeth Oswald, and Bart 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*, page 546, Washington, DC, USA, 2004. IEEE Computer Society.

[OMHT06] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006*, Lecture Notes in Computer Science, pages 192 – 207. Springer, 2006.

[Pro05] Emmanuel Prouff. DPA Attacks and S-Boxes. In *FSE*, pages 424–441, 2005.

[QS01] Jean-Jacques Quisquater and David 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*, pages 200–210, London, UK, 2001. Springer-Verlag.

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. Optical Fault Induction Attacks. pages 2–12. Springer-Verlag, 2002.

[Sch95] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. 2nd edition, 1995.

[Sha49] Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell Systems Technical Journal*, 28:656–715, 1949.