FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Side-Channel Analysis: Efficient Attacks and Fault-Tolerant Countermeasures

by

Ing. Vojtěch Miškovský

A dissertation thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

Dissertation degree study programme: Informatics
Department of Digital Design

Prague, August 2019

Supervisor:
doc. Ing. Hana Kubátová, CSc.
Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Co-Supervisor:
Dr.-Ing. Martin Novotný
Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract and Contributions

Since side-channel analysis poses a serious threat to cryptographic devices, it became an extensively researched area. Two topics related to side-channel analysis are the main aim of this dissertation thesis: efficient attack implementations and fault-tolerance of SCA countermeasures.

Three approaches for time-efficient enhancements of correlation power analysis are proposed: efficient computations, trace aggregation, and edge-detection-based key candidate selection. All three approaches proved to be functional and useful by experimental evaluations. We also evaluate how different dependable architectures based on different kinds of redundancy affect resistance to side-channel analysis. The results show, that the influence is not statistically significant. In accordance with this result, architectures for dependable and secure design were proposed. These architectures are based on hardware modular redundancy and masking schemes. The case study shows, that the proposed architectures keep the simplicity of the modular redundancy architectures and the SCA resistance of masking schemes, while it significantly saves resources.

In particular, the main contributions of the dissertation thesis are as follows:

1. Evaluation of multiple approaches to first-order and higher-order correlation computations and a proposal of time-efficient and numerically stable implementations.

2. An efficient method for speeding up power analysis based on an aggregation of power traces.

3. Improved correlation distinguisher based on edge detection, especially useful in noisy environments.

4. Comprehensive evaluation of the influence of dependable architectures on the resistance against side-channel analysis.

5. Area-efficient architectures for dependability and the resistance to side-channel analysis.

Keywords:
Security, Dependability, Side-Channel Analysis, Differential Power Analysis, Correlation Power Analysis.

Acknowledgements

First of all, I would like to thank my supervisors, Hana Kubátová and Martin Novotný, for all of their guidance and assistance. I would also like to express my gratitude to my colleagues Petr Fišer and Petr Socha, who provided me with valuable feedback for this thesis, and also all other colleagues from the department of digital design for research cooperation and productive environment. Last but not least, my special thanks goes to my girlfriend, family, and friends for all of their support.

This research has been partially supported by grants SGS15/119/OHK3/1T/18, SGS16/121/OHK3/1T/18, and SGS17/017/OHK3/1T/18 of Czech Technical University in Prague, GA16-05179S of Czech Grant Agency, and CELSA/17/033 of Central European Leuven Strategic Alliance.

Contents

List of Figures

List of Tables

Abbreviations

Security

Attacks

SCA	Side-Channel Analysis
SPA	Simple Power Analysis
DPA	Differential Power Analysis
CPA	Correlation Power Analysis
MIA	Mutual Information Analysis
FA	Fault Analysis/Attack
FSA	Fault-Sensitivity Analysis
TVLA	Test Vector Leakage Assessment

Countermeasures

TI	Threshold Implementation
DOM	Domain-Oriented Masking
CMS	Consolidated Masking Schemes
DPL	Dual-rail with Precharge Logic
WDDL	Wave Dynamic Differential Logic
STTL	Secure Triple Track Logic
LRA	Leak-Resistant Arithmetic

Ciphers

AES	Advanced Encryption Standard
RSA	Rivest–Shamir–Adleman

Dependability

DMR	Dual Modular Redundancy
TMR	Triple Modular Redundancy
NMR	N-Modular Redundancy
EDC	Error Detection Code
ECC	Error Correction Code

Miscellaneous

LSB	Least Significant Bit
MSB	Most Significant Bit
SNR	Signal to Noise Ratio
CPU	Central Processing Unit
GPU	Graphics Processing Unit
FPGA	Field-Programmable Gate Array
LUT	Look-Up Table
CFGLUT	ConFiGurable Look-Up Table
CMOS	Complementary Metal–Oxide–Semiconductor
STL	Standard Template Library

Chapter **1**

Introduction

Computers, communication networks, and other electronic systems became an essential part of our everyday lives. We pay by cards, communicate by smartphones, drive smart cars, watch smart TVs, wear smart clothes and so on. As these systems can be connected over the Internet, they are making our lives easier as so as they are posing a threat to our privacy. As strangers reading our text messages or controlling our smart lightbulbs are usually undesirable, our data and communication need to be protected by cryptography. These smart devices have also pervaded areas, where human lives, health or property depend on the proper functionality of the devices, like automotive industry (e.g. smart cars) or medical appliances (e.g. pacemakers). Electronic systems utilized in these areas need to fulfill strict dependability properties.

1.1 Motivation

As implied above, some electronic systems need to be designed with regard to both the security and dependability. Cryptographic units need to be involved and protected against attackers to ensure the security properties, while fault-tolerant architectures need to be used for the implementation of all parts of these systems to make the whole systems dependable. Fault-tolerant design and attack countermeasures have a significant influence on physical properties like area or power consumption and the mutual influence of these design approaches is the aim of this thesis. It is mainly focused on the side-channel protection of dependable systems.

Numerous experimental evaluations are required to be performed in order to conduct the research. For this reason, we decided to dedicate a reasonable amount of research to efficient evaluations of attacks. Specifically, we focused on correlation power analysis and leakage assessment using Welch's t-test.

1.2 Problem Statement

Many approaches to make a design fault-tolerant exist. These fault-tolerant architectures affect physical properties of the original design, like area, power consumption, execution time and others. A change of any physical property of the design can affect its security, as these physical properties are exploited by recognized side-channel analysis. This influence needs to be explored and evaluated.

Another problem of devices designed to be both dependable and secure is a significant overhead. Fault-tolerant architectures usually demand high area and/or power and/or time overhead and the same applies to countermeasures against side-channel attacks. Therefore, when a design combines these two approaches, the overhead is cumulative. A proposition of composite architectures leading to a reduction of the overhead is desired.

For a comprehensive evaluation of side-channel resistance, a significant amount of statistical computations needs to be performed. The complexity of these computations depends on multiple characteristics of the evaluation. We decided to explore the possibilities of adjusting these characteristics to speed up these evaluations.

1.3 Goals of the Dissertation Thesis

1. Analysis of possibilities to speed up the correlation power analysis and test vector leakage assessment.

2. Comprehensive evaluation of the influence of fault-tolerant architectures on resistance against side-channel analysis.

3. Proposition of architectures combining fault-tolerance and resistance to side-channel analysis with regard to the overhead.

1.4 Structure of the Dissertation Thesis

The thesis is organized into 5 chapters as follows:

Chapter 1 — Introduction: Describes the motivation behind our efforts together with brief problem description. There is also a list of goals of this dissertation thesis.

Chapter 2 — Background and State-of-the-Art: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art.

Chapter 3 — Enhancements of Correlation Power Analysis: Describes our propositions of speeding up correlation power analysis and test vector leakage assessment.

Chapter 4 — Dependability vs. Security: Presents an evaluation of the influence of fault-tolerant architectures on the resistance against side-channel analysis.

Chapter 5 — Conclusions: Summarizes the results of our research, suggests possible topics for further research, and concludes the thesis.

Chapter 2

Background and State-of-the-Art

In this chapter, we present the work related to the topic of this thesis. In Section 2.1, an overview of the non-profiled side-channel attacks and a deeper description of correlation power analysis is presented as so is an overview of SCA countermeasures. Common dependable architectures based on various kinds of redundancy are described in Section 2.2. Existing research of the influence of fault-attack countermeasures on resistance against SCA is presented in Section 2.3. Differences between principles of fault-attack countermeasures (existing work) and dependable architectures (our approach) are also explained in this section.

2.1 Side-Channel Analysis

Side-channel analysis is a group of attacks exploiting physical properties (so-called side-channels) of a device under attack (usually an encryption module). Therefore, these attacks pose a threat even to cryptographically strong ciphers. The basic principle stands on an assumption that the activity of the side-channel is somehow dependent on the secret information (usually the cipher key) used by the device. Many exploitable side-channels exist, e.g. power consumption [2], electromagnetic radiation [3], time [4] or temperature [5]. The most common side-channel is power consumption, which is our primary focus in this research, but the further mentioned principles are also applicable to other side-channels, especially electromagnetic radiation [6].

2.1.1 Simple Power Analysis

In 1999, Paul Kocher et al. introduced two attacks to lay a foundation of side-channel analysis research — simple power analysis and differential power analysis [2]. **Simple power analysis** permitted to identify particular operations of the encryption from a very small amount of power traces (or even a single one) and extract the secret information. This attack is especially efficient against software implemented asymmetric ciphers based on modular exponentiation like RSA [7], where square and multiply algorithm permits us to visually identify the difference between an input bit being logical one (both square and multiply are processed) and logical zero (only the square is processed). Furthermore, this attack was later adjusted to other types of ciphers, e.g. in [8], simple power analysis of AES [9] exploiting key expansion procedure was presented.

2.1.2 Differential Power Analysis

The other attack presented by Kocher et al., **differential power analysis**, proved itself to be even more beneficial for side-channel analysis research. In comparison with SPA, it is a more complex attack, but also a more powerful attack. DPA exploits the fact that the power consumption of the device under attack at a certain point in time is proportional to the currently processed data. Even a difference in a single bit leads to minor differences in power consumption. The attack assumes that an attacker is able to measure the power consumption of the device during encryption and that he is also able to feed the device with arbitrary input data (plaintext).

At first, the attacker collects power traces (the actual amount depends on the device under attack, it can be from tens up to millions or even more) during the encryption while the device is fed with random plaintexts. The collected power traces are then analyzed in the following way. A hypothetical value — an internal value of the encryption algorithm, which is predicted using a function of (a part of) a key and (a part of) plaintext or ciphertext — is computed for each part of the key (e.g. byte of the key), each key candidate and each measured encryption. (The partitioning of the key is important to keep the amount of possible key candidates feasible.) Then, for each part of the key and each key candidate, the power traces are partitioned into two sets according to the value of one of the bits (e.g. LSB or MSB). Power traces in these two sets are averaged and the two average traces are subtracted. In the case of a wrong key candidate, the trace resulting from the subtraction is nearly constant at zero (with some insignificant peaks caused by noise), while in the case of the correct key candidate, a major peak appears in time points, where the hypothetical value is processed. The whole cipher key is extracted this way.

2.1.3 DPA Derivatives

Differential power analysis became a highly researched topic and many enhancements and derivatives were proposed. An attack built on the DPA basis, but using electromagnetic radiation as a side-channel, was proposed in [3]. Messerges et al. [10] proposed a multi-bit DPA which uses partitioning of the power traces into two sets by Hamming weight of hypothetical value. Another approach is to correlate power consumption estimates (which are based on a model of the power consumption, e.g. Hamming weight of a hypothetical value or Hamming distance between hypothetical values in two consequent states of encryption) to the measured power consumption [11]. This approach is usually called correlation power analysis and its comprehensive description is presented further. In [12], Gierlichs et al. proposed a ranking of the key candidates using mutual information between the hypothetical leakage and the measured power consumption (mutual information analysis).

In summary, all these (and other) works generalized differential power analysis to a family of attacks, where an attacker chooses a side-channel (usually power or electromagnetic radiation), a model of the side-channel (e.g. power model), and a distinguisher — statistical method for ranking of the key candidates (difference of means (the original DPA), correlation (CPA), mutual information (MIA), euclidean distance, etc). An overview of distinguishers is presented in [13] while the comparison of them was recently published in [14].

2.1.4 Correlation Power Analysis

This section contains a comprehensive description of correlation power analysis for which we present multiple enhancements in Chapter 3 of this thesis.

The first step of the attack is a power measurement. The total number of n power traces is measured containing s samples each, feeding the device under attack with random plaintexts. We can consider the collected power traces a matrix $n \times s$.

In the next step, a power model needs to be chosen. For example, in the case of AES, it is usually the Hamming weight of the first S-box output (in the case of a software implementation) or the Hamming distance between the output of the last round (the ciphertext) and the output of the last but one round (in the case of a hardware implementation). The power model is used for calculation of hypothetical leakage for each of n encryptions and for each of hypothetical key candidates. The key and the plaintext (or the ciphertext) gets split into b smaller parts (e.g. 16 bytes in the case of AES-128). We calculate the hypothetical leakage value for each of n plaintexts (or ciphertexts), each of c possible key candidates (e.g. 256 possible values of a byte) and we repeat this for each of b parts of the key. As a result, we have b matrices $n \times c$ of hypothetical values.

Since we assume a linear dependence between the measured variables [11], Pearson correlation coefficient is used. The Pearson correlation coefficient between random variables X and Y is defined as follows [15]:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y}, \tag{2.1}$$

with $\operatorname{cov}(X,Y)$ being the covariance between variables X and Y, and σ_X, σ_Y being standard deviations of variables X, Y respectively.

The measured power trace matrix ($n \times s$) is correlated with hypothetical value matrix ($n \times c$). This is repeated for each of b parts of key resulting in b correlation matrices $c \times s$. A vector of all samples corresponding to one key candidate is a correlation trace. All b parts of the key can be extracted e.g. by looking for the highest (absolute) value in each of correlation matrices or by a visual examination of each correlation trace (while looking for high peaks, sharp edges or another anomaly).

2.1.5 Higher-Order Analysis

This section presents a brief description of multivariate and univariate higher-order analysis.

2.1.5.1 Multivariate Higher-Order Analysis

Higher-order differential power analysis was first mentioned by Kocher et al. [2] as an attack that combines one or more samples within a single power trace. This kind of attack is useful when related intermediate cipher values are processed at a different time, e.g. in the case of masked software implementations. It is referred to as multivariate higher-order analysis. Nevertheless, finding these time points to combine is a complex task [16].

2.1.5.2 Univariate Higher-Order Analysis

Different situation may occur in hardware implementations, where the intermediate values may be masked in parallel (as discussed in Section 2.1.6.1), i.e. they manifest themselves at the same time. Attacker can overcome this by performing a univariate higher-order analysis, which is performed at each sample independently. For univariate second-order analysis, every power sample gets mean-free squared [17]:

$$t'_i = (t_i - \mu_t)^2, \tag{2.2}$$

where μ_t is a sample mean at given sampling point. For univariate d-order analysis, where $d > 2$, every power sample gets additionally standardized:

$$t'_i = (\frac{t_i - \mu_t}{s_t})^d, \tag{2.3}$$

where s_t is a sample standard deviation at given sampling point.

2.1.6 Countermeasures

Many countermeasures against side-channel attacks were proposed over the years. We focus on the ones implemented in hardware. These can be divided into two main categories: hiding and masking. In this section, we provide a brief overview of hiding, while we look deeper into masking, as masking schemes serve as a basis to our dependable architectures presented in Section 4.2.

2.1.6.1 Masking

Masking countermeasures are based on randomizing internal values of encryption so that these do not correlate with any of the input values. The most common method is Boolean masking [18]. Alternatively, multiplicative [19] or affine [20] masking can be used.

In Boolean masking, each intermediate value x is split into n shares x_i, where

$$x = \bigoplus_{i=1}^{n} x_i \tag{2.4}$$

. Multiple masking schemes resistant to side-channel analysis at arbitrary order even at presence of glitches were proposed, for example threshold implementation [21, 22, 23], domain-oriented masking [24] or consolidated masking schemes [25]. These schemes are also proved to be resistant to fault-sensitivity analysis [26, 27]. We provide deeper description of threshold implementation since we use it for a case study of architectures presented in Section 4.2.

Threshold Implementation Threshold implementation is a Boolean masking scheme provably secure against side-channel analysis of arbitrary order d even in presence of glitches. While the information is split into n shares x_i (as in Equation 2.4), all the transformations applied to the shares need to satisfy specific properties.

In the case of **linear transformations**, the implementation is straightforward. For every transformation $z = L(x)$ over $GF(2^m)$, which is linear over $GF(2)$, an equation

$$z = \bigoplus_{i=1}^{n} z_i = \bigoplus_{i=1}^{n} L(x_i) = L(\bigoplus_{i=1}^{n} x_i) = L(x) \tag{2.5}$$

holds. Therefore, each share can be independently processed by linear transformations causing no side-channel leakage.

Situation is more complicated regarding **non-linear transformations**. Each transformation $z = N(x, y, \ldots)$, which is not linear over $GF(2)$, is realized using shared functions $f_1, f_2, \ldots, f_n$ satisfying three properties: non-completeness, correctness, and uniformity.

Non-completeness: Every function is independent of at least one share of each of the input variables (x, y, ...), for example:

$$\begin{aligned} z_1 &= f_1(x_2, x_3, \ldots, x_n, y_2, y_3, \ldots, y_n, \ldots) \\ z_2 &= f_2(x_1, x_3, \ldots, x_n, y_1, y_3, \ldots, y_n, \ldots) \\ &\ldots \\ z_n &= f_n(x_1, x_2, \ldots, x_{n-1}, y_1, y_2, \ldots, y_{n-1}, \ldots) \end{aligned} \tag{2.6}$$

Correctness: Sum of the outputs of the functions is the correct output of the original transformation:

$$z = \bigoplus_{i=1}^{n} z_i = \bigoplus_{i=1}^{n} f_i(\ldots) = N(x) \tag{2.7}$$

Uniformity: For each shared function, for each unshared input combination, each shared output value must be equally probable.

2.1.6.2 Hiding

Basic area-efficient hiding countermeasures can be deployed using simple signal-to-noise ratio reduction — switching capacitors, noise generation, pipelining [28], etc. Another approach is to hide information in time, which is usually implemented in software [29], but it can be also used in hardware implementations [30].

Another kind of hiding technique is based on so-called dual-rail with precharge logic. Design protected by these countermeasures is built using special logic gates providing the same switching activity for any logic transition. This makes the power consumption caused by the switching activity constant and independent of the processed internal values. There are many schemes using this principle, e.g. WDDL [31, 32] or STTL [33]. An overview of DPL techniques is presented in [34].

Countermeasures based on dynamic reconfiguration are entangled with reconfigurable logic, especially FPGAs. There are multiple approaches to dynamic reconfiguration as a side-channel analysis countermeasure. For example, a coarse grain reconfigurable architecture based on leak-resistant arithmetic [35] is proposed in [36]. Approach based on temporal jitter is explained in [37]. In [38], multiple countermeasures using FPGA-specific design units are proposed. Countermeasure based on a dynamically adjustable decomposition of S-box using configurable look-up table (CFGLUT) is presented in [39].

2.1.7 Leakage Assessment

Side-channel analysis became a well-known threat and many countermeasures were proposed, therefore a need for evaluation and comparison of the countermeasures emerged. At first, statistical tests based on mutual information were proposed [40, 41]. Nevertheless, these tests need to estimate the probability distribution of the leakage and they cannot target a certain statistical order. Later, tests based on Student's t-distribution were proposed [42]. In multiple works, e.g. [43] or [23], Welch's t-test was used for leakage assessment without a deeper examination of the procedure. In [16], Schneider and Moradi provide an extensive evaluation of leakage assessment (including multivariate and higher-order evaluation) using Welch's t-test, which became a widely used method for leakage evaluation. More methods for leakage assessment were recently published: chi-square test-based [44] or deep learning-based [45].

2.1.7.1 Non-Specific Welch's t-test

We focus on non-specific, fixed vs. random Welch's t-test in this thesis, as it is proposed in [16].

Welch's t-test serves to test a null hypothesis that two population sets have the same mean, hence we use it to test whether samples of both sets were drawn from the same population. It is a generalization of Student's t-test for populations with different variances [46]. While μ_1 and μ_2 are sample means of the two sets, s_1^2 and s_2^2 are sample variances of the sets and n_1 and n_2 are cardinalities of the sets, we can compute Welch's statistic t as

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{2.8}$$

and degrees of freedom v as

$$v = \frac{(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2})^2}{\frac{(\frac{s_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{s_2^2}{n_2})^2}{n_2 - 1}} \tag{2.9}$$

.

In the case of non-specific test, no assumption on the implementation is needed in contrast to the specific test. Fixed vs. random test means, that one population set contains traces measured when fixed inputs are fed to the device, while the other set consists of traces acquired using random inputs. This approach demands a specific manner of trace collection. Before each encryption, either (preselected) fixed or random input is chosen with the same probability (a coin is flipped). Having the two sets of power traces containing s samples each, the t characteristic and degrees of freedom v are computed for each of s sample points independently. For simplicity, usually a threshold $|t| > 4.5$ is defined to reject the null hypothesis (that the samples from both fixed and random sets are drawn from populations with the same means) without considering the degrees of freedom.

2.2 Dependable Architectures

Dependable systems are an intensively researched area. Fault-tolerant digital design needs to be deployed in order to make the system dependable. In this research, we mostly focus on the most prominent redundancy architectures. Usually, we distinguish three types of redundancy: space redundancy, time redundancy, and information redundancy [47], [48].

2.2.1 Space Redundancy

Space redundancy architectures deploy additional hardware resources (i.e. area overhead) to provide the desired dependability properties. The most simple example of space redundancy is modular redundancy. Modular redundancy is based on the replication of the whole design and evaluation of the outputs using various comparison logic. Regarding the number of modules, three basic modular redundancy architectures can be distinguished: duplex (or also dual modular redundancy), triple modular redundancy, and N-modular redundancy.

2.2.1.1 Duplex

Duplex (DMR) scheme consists of two copies of the original design. Outputs of these two modules are compared and if they differ, an error is reported. This scheme is able to detect at least a single fault, while its overhead is more than 100% (one extra module and a comparator). A typical implementation of duplex can be seen in Figure 2.1

2.2.1.2 Triple Modular Redundancy

TMR uses three copies of the original design. The outputs are processed using majority voter(s), providing the correct result in the case of at least one fault. To protect the design also against a fault in the voters, three majority voters need to be deployed. The overhead

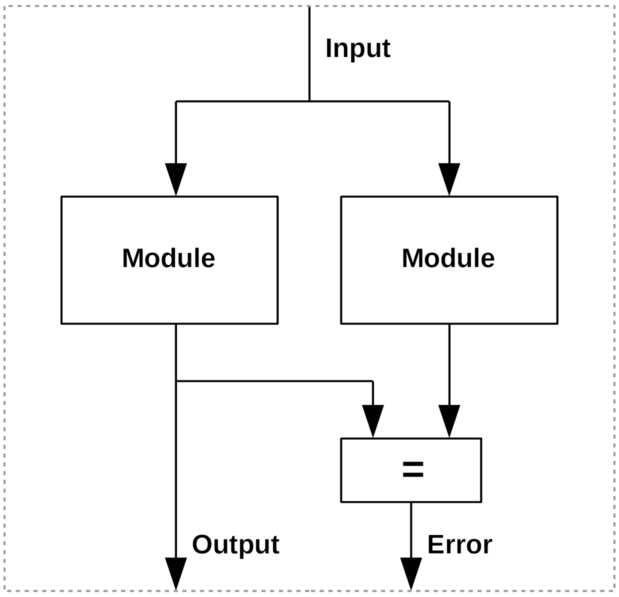


Figure 2.1: Diagram of typical implementation of DMR

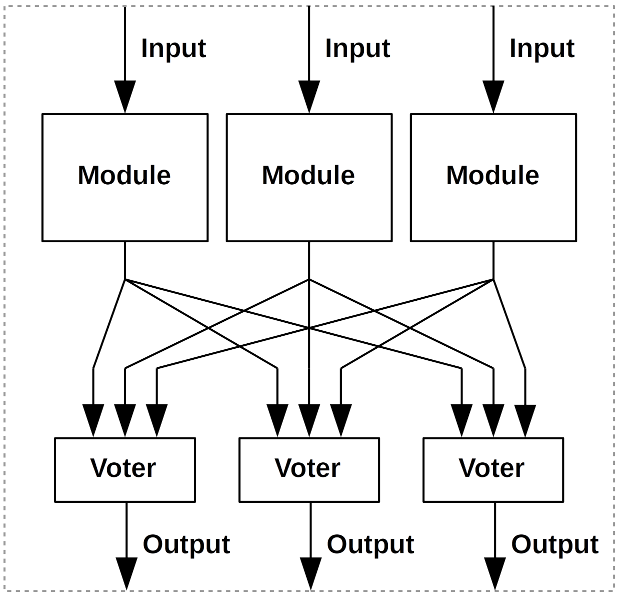


Figure 2.2: Diagram of typical implementation of TMR

of this scheme is more than 200% (2 extra modules and voters). A typical implementation of TMR can be seen in Figure 2.2

2.2.1.3 N-Modular Redundancy

NMR is a generalization of TMR. It consists of N modules and it is able to mask at least $n = (N-1)/2$ faults. The overhead of this scheme is at least $(N-1) \times 100\%$.

2.2.2 Time Redundancy

Time redundancy improves dependability properties using additional time. An elementary example is repeating the whole algorithm. The main disadvantage of time redundancy techniques is the fact, that usually only transient faults can be detected or corrected.

2.2.3 Information Redundancy

Even though information redundancy can demand both area and time, it is considered a different type of redundancy. Information redundancy techniques primarily include error detection/correction codes like parity, Hamming code, cyclic codes, etc.

2.3 Fault Attack Countermeasures vs. Side-Channel Resistance

Both fault-tolerant and attack-resistant techniques are deeply researched, but there is a lack of research about their mutual influence and interplay. To the best of our knowledge, the only publications somehow related to this topic are dedicated to the influence of fault attack countermeasures on side-channel resistance. These are presented in this section.

Fault-tolerant techniques are similar to countermeasures against fault-injection attacks. Nevertheless, there are significant differences. The main objective of fault-tolerance as a dependability property serves primarily to protect the device against spontaneously introduced faults and to provide the correct output even if a fault occurs (or at least to detect that the output is incorrect). It needs to protect against both transient and permanent faults in any part of the design. On the other hand, fault-tolerance as a countermeasure against fault attacks serves to protect against intentionally introduced faults. Its purpose is rather to prevent an attacker from revealing secret information than providing the correct output. Also, not all parts of the design are usable for fault injection. For this reason, fault-injection countermeasures often protect only some parts of the design (usually the non-linear ones). Finally, most of the fault attacks are based on transient faults only.

Regazzoni et al. [49],[50] focused on influence of fault attack countermeasures on power analysis resistance. They compared differential power analysis resistance of several AES S-box implementations protected by various error detection and error correction codes (parity-based codes, residue-based codes, Hamming codes). These implementations were synthesized using ST-Microelectronics 90nm CMOS standard cell library. They performed differential power analysis using noise-free traces generated by the SPICE simulator. They concluded that used fault attack countermeasures are making the device more vulnerable to differential power analysis.

A more comprehensive comparison was introduced in [51]. This research was focused on implementations of S-box and MixColumns functions protected by space redundancy and information redundancy architectures. These protected functions were part of the whole AES design (in contrast with Regazzoni et al.) implemented in Xilinx Spartan-6

FPGA on Sakura-G board. These implementations were compared by the minimal number of power traces necessary to reveal the correct cipher key. Authors concluded that these methods have a negative influence on attack-resistance.

In contrast with the above-mentioned work, **we focus on fault-tolerance in the context of the reliability in this work. Fault attacks are not a part of our main concern!**

Chapter 3

Enhancements of Correlation Power Analysis

Side-channel analysis poses a serious threat to electronic devices. Fast execution of these attacks is crucial for their experimental evaluations (e.g. the ones presented in Chapter 4 of this thesis). In this chapter, we propose three approaches to increase the time efficiency of correlation power analysis. The time complexity of both parts of the attack (the measurement part and the computational part) is dependent on the number of measurements n and the number of samples per power trace s. In Section 3.1, we deal with efficiency of the correlation computations and Welch's t-test. We present an evaluation and a comparison of three algorithms for the first-order analysis, evaluation, and comparison of two approaches to higher-order analysis and we present results of our OpenMP and OpenCL based implementations. A time optimization of correlation evaluation based on decreasing the number of samples s by aggregation of the samples is presented in Section 3.2. Finally, in Section 3.3, we propose an improvement of the correlation distinguisher based on advanced identification of the correct key candidate in order to decrease the number of traces n.

The work presented in Section 3.1 was done in cooperation with Petr Socha (a master student under the supervision of the author of this thesis). It was partially presented at international conferences DDECS 2017 [A.2] and MECO [A.7].

The work introduced in Section 3.2 was presented at an international conference MECO 2018 [A.5].

The work presented in Section 3.3 was done in cooperation with Petr Socha (a master student under the supervision of the author of this thesis). It was partially presented at an international conference DSD 2018 [A.6] and it was invited to Microprocessors and Microsystems journal, where an extended version was recently accepted for publication [A.8].

All implementations and improvements discussed in Sections 3.1 and 3.3 are included in our open-source side-channel toolkit SICAK [60], allowing anyone for further usage and improvements. This toolkit was presented at workshop Trudevice 2019 [A.17].

3.1 Efficient Computations

To perform a power attack against modern low-power devices, we need to obtain a huge amount of power traces. Also, higher-order power analysis demands more power traces than the first-order analysis [52], [53]. Therefore, the correlation calculation can get highly time demanding. Several implementations of correlation power analysis have been published (e.g. [54], [55]), however the methods used for computing correlation coefficients may suffer from poor numerical stability [56]. In [57], various aspects of CPA are discussed, some algorithms are suggested and compared; however, presented methods may be numerically unstable as well. A numerically stable approach of computing correlations for side-channel attacks is introduced in [17]; however, no implementation or performance comparison of discussed methods is presented.

To the best of our knowledge, no comparison of hereby presented Pearson correlation coefficient computation methods, that would provide tangible figures and discuss their properties, is available in the open literature.

In this section, we present multiple approaches to the univariate first-order and higher-order analysis. We focus on the correlation power analysis and test vector leakage assessment using a non-specific methodology based on Welch's t-test [42]. We describe the principles of these methods, discuss their advantages and limitations and we examine the computational and memory demands of the approaches using our open-source CPU and GPU-accelerated implementations.

In Section 3.1.1, we present methods for first-order analysis using Pearson correlation coefficient, introduce the mathematical background for each method, discuss their advantages and disadvantages and propose efficient implementations. A GPU-accelerated implementation is also described. In Section 3.1.2, two computational approaches to higher-order analysis are discussed. An implementation of first-order and higher order Welch's t-test is described in Section 3.1.3. A performance evaluation and a comparison of the methods for attacking AES cipher are presented in Section 3.1.4 and we conclude our results in Section 3.1.5.

3.1.1 First-Order Analysis

In the process of obtaining the correct key, a Pearson correlation coefficient between each sample and the power model must be computed as described in Section 2.1.4.

Each of n measurements of power traces is done with s samples per trace, resulting in a vector of random variables $\mathbf{X} = [X_1, ..., X_s]$, one for each sample.

The plaintext/ciphertext used while measuring the traces, after being processed by an appropriate power model, results in a vector of variables $\mathbf{Y} = [Y_1, ..., Y_c]$, one for each of c key candidates.

To obtain the correct key, the correlation coefficient between each sample per trace and each key candidate must be computed. The result of the algorithm is the correlation matrix $\mathbf{C}$ of size $s \times c$, where $C_{i,j} = \rho_{X_i,Y_j}$, computed for each part of key separately.

The correct key can be found in the correlation matrix **C** simply by searching for the maximal and the minimal correlation coefficient, or by some other statistical method.

Different approaches to implementing the computation of the correlation coefficient on a statistical sample will be described in the following subsections since the selected method of processing the data has a significant effect on both time and memory performance, as well as on some other properties of the calculation. Additionally, a description of our implementation is presented for each approach as so as a description of a GPU-accelerated implementation.

3.1.1.1 Two-Pass Approach

Given that $\operatorname{cov}(X,Y) = \mathrm{E}[(X-\mu_X)(Y-\mu_Y)]$ and $\sigma_X^2 = \mathrm{E}[(X-\mu_X)^2]$, with μ_X, μ_Y being means of X, Y respectively [15], we can express the Pearson correlation coefficient in a format more suitable for processing of the statistical sample:

$$\rho_{X,Y} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{\mathrm{E}[(X-\mu_X)^2]}\sqrt{\mathrm{E}[(Y-\mu_Y)^2]}}. \tag{3.1}$$

Based on Equation 3.1, a two-pass algorithm for the correlation computation can be easily designed. When the correlation is computed on a statistical sample, letter r will be used instead of letter ρ. Assuming we have datasets $X = \{x_1, ..., x_n\}$ and $Y = \{y_1, ..., y_n\}$ with cardinalities n and sample means $\overline{x}$ and $\overline{y}$, we can write:

$$\mathrm{r}_{X,Y} = \frac{\sum\limits_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum\limits_{i=1}^{n}(x_i-\overline{x})^2}\sqrt{\sum\limits_{i=1}^{n}(y_i-\overline{y})^2}}. \tag{3.2}$$

The first time the data sets are passed, the sample means $\overline{x}$ and $\overline{y}$ are calculated. With the second pass, the variances (thus standard deviations) and the covariance are calculated, allowing to easily compute the final correlation coefficients.

Unless the number of statistical samples n is very large, this method is numerically stable and it is giving accurate results [56]. However, the two-pass (adj.) computation is undesirable, especially for large data sets.

For once, the time necessary for the calculation grows with more statistical samples in the set, thus the time performance issue gets significant where more power traces may be needed (on FPGAs, when processing higher-order analysis, etc.) [57]. Multi-pass algorithms are impractical in this case since the distributed memory access may dominate the computation time and memory consumption may be unbearable.

Also, the final application asks for the ability to stop the computation, check the results and, in the case of need, continue the computation with more measured statistical samples (i.e. power traces), allowing e.g. to find out the number of power traces necessary to obtain the correct key candidate. A new added statistical sample in this case requires to pass through all the already processed data once again.

Implementation By going through all the data, a vector $\overline{\mathbf{X}} = [\overline{x_1}, ..., \overline{x_s}]$ of s sample means, one for each of s samples per trace, and a vector $\overline{\mathbf{Y}} = [\overline{y_1}, ..., \overline{y_c}]$ of c sample means, one for each of c key candidates, are calculated.

In the second pass, with the precomputed sample means, vectors containing variances $\sigma^{\mathbf{2}}_{\mathbf{X}} = [\sigma^2_{X_1}, ..., \sigma^2_{X_s}]$ and $\sigma^{\mathbf{2}}_{\mathbf{Y}} = [\sigma^2_{Y_1}, ..., \sigma^2_{Y_c}]$, as well as a covariance matrix $\mathbf{K}$ of size $s \times c$, where $K_{i,j} = \mathrm{cov}(X_i, Y_j)$, are easily computed, from which the final correlation matrix $\mathbf{C}$ is easily derived.

The whole process is repeated for each part of the key with the same measured power traces (X), but a different power model, based on the plaintext/ciphertext used (Y).

Our early implementation of the two-pass algorithm, written in C++ using standard template library, including the power-model computation, is unbearably slow for a large amount of traces, while still faster than some universal mathematical software (e.g. Wolfram Mathematica).

3.1.1.2 Naïve One-Pass Approach

When we expand the Equation 3.1 using the linearity of the expectation and the variance [15], we obtain:

$$\rho_{X,Y} = \frac{\mathrm{E}[XY] - \mathrm{E}[X]\mathrm{E}[Y]}{\sqrt{\mathrm{E}[X^2] - \mathrm{E}[X]^2}\sqrt{\mathrm{E}[Y^2] - \mathrm{E}[Y]^2}}, \tag{3.3}$$

giving us a convenient one-pass approach to the computation of the correlation coefficient:

$$\mathrm{r}_{X,Y} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2}}. \tag{3.4}$$

In a single pass through all the data (statistical samples), the sums $\sum_{i=1}^n x_i y_i$, $\sum_{i=1}^n x_i$, $\sum_{i=1}^n x_i^2$, $\sum_{i=1}^n y_i$ and $\sum_{i=1}^n y_i^2$ are computed, allowing to easily calculate the sample means, the variances, and the covariance, and finally the correlation coefficients.

This approach promises a better time performance of the implemented calculation since all the data need to be accessed only once. Also, since the summation of disjoint pools of statistical samples (measured power traces) is an independent operation, this kind of one-pass calculation allows for a straightforward parallelization.

Since the critical part of the computation is the summation, the algorithm can be stopped at any time, providing the answer based on processed samples, and then resumed with a little extra effort.

However, this method suffers from poor numerical stability [56]. Consider the denominator of the Equation 3.4, where the sample variance is computed as a difference of two positive floating-point numbers. Due to this subtraction, severe cancellations may occur, leaving the result dominated by the roundoff. A similar situation may occur in the numerator of the Equation 3.4.

Implementation In a single pass through the measured data and power model, a vector of sums $\mathbf{S_X} = [\sum_{k=1}^{n} x_{1_k}, ..., \sum_{k=1}^{n} x_{s_k}]$ of power traces for each sample and a vector of their powers $\mathbf{S_{X^2}} = [\sum_{k=1}^{n} x_{1_k}^2, ..., \sum_{k=1}^{n} x_{s_k}^2]$, as well as vectors of sums of the power model $\mathbf{S_Y} = [\sum_{k=1}^{n} y_{1_k}, ..., \sum_{k=1}^{n} y_{c_k}]$ and $\mathbf{S_{Y^2}} = [\sum_{k=1}^{n} y_{1_k}^2, ..., \sum_{k=1}^{n} y_{c_k}^2]$ are computed, as well as a matrix of sums $\mathbf{S_{XY}}$ of size $s \times c$, where $S_{XY_{i,j}} = \sum_{k=1}^{n} x_{i_k} y_{j_k}$. From these, based on the Equation 3.4, the final correlation matrix $\mathbf{C}$ is easily computed.

The bottleneck of the whole computation is the multiplication and the summation while creating the matrix $\mathbf{S_{XY}}$. Fine-grain parallelization of this operation results in a very good scalability of this algorithm. The minimum amount of the working memory needed is bounded only by the size of vectors and the matrix of sums, mentioned above.

Our implementation, written in C/C++ and using OpenMP, gets much faster than our two-pass implementation, also thanks to the better memory usage. The final correlation computation was separated from the power model computation, where a rearrangement of the data for a better correlation computation performance also happens. The time performance of the power model computation is well satisfactory on its own, as well as it is scalable.

3.1.1.3 Incremental One-Pass Approach

The poor numerical stability of the naïve one-pass algorithm, combined with the necessity to process a lot of statistical samples in a reasonable time, demands a better approach. This new approach should preferably maintain both the time complexity of the naïve one-pass algorithm and the numerical stability of the two-pass algorithm.

Let's assume that we have datasets $X = \{x_1, ..., x_n\}$ and $Y = \{y_1, ..., y_n\}$ with cardinalities n. Let's assume that $X' = X \cup \{x_{n+1}\}$ and $Y' = Y \cup \{y_{n+1}\}$ are the datasets created by adding some new samples to the sets.

A recurrent formula for the sample mean can be easily obtained and is widely known:

$$\overline{x'} = \overline{x} + \frac{x_{n+1} - \overline{x}}{n+1}. \tag{3.5}$$

Now, let's assume we have the sample variance defined as $\sigma_X^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$, where n is the cardinality of the dataset. Let $M_{2,X} = \sum_{i=1}^{n}(x_i - \overline{x})^2$, therefore $\sigma_X^2 = \frac{1}{n}M_{2,X}$.

For the dataset X' created by adding a new statistical sample to the set X, we would like to calculate the variance as $\sigma_{X'}^2 = \frac{1}{n+1}M_{2,X'}$. Using the Equation 3.5, a recurrent formula for the sum $M_{2,X'}$ can be directly used [58]:

$$M_{2,X'} = M_{2,X} + (x_{x+1} - \overline{x})(x_{x+1} - \overline{x'}). \tag{3.6}$$

For the covariance, a similar recurrent formula exists. Assume a dataset consisting of pairs $S = \{(x_i, y_i) | x_i \in X, y_i \in Y\}$, representing a pair of random variables X and Y, with the cardinality n and the sample means $\overline{x}$ and $\overline{y}$. The sample covariance is defined as $\mathrm{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})$. Let $C_{2,S} = \sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})$, then $\mathrm{cov}(X,Y) = \frac{1}{n}C_{2,S}$.

Let's assume that $S' = S \cup \{(x_{n+1}, y_{n+1})\} = \{(x'_i, y'_i) | x'_i \in X', y'_i \in Y'\}$ is a new dataset created by adding a new pair of samples. The covariance of the dataset S' can be then computed as $\mathrm{cov}(X', Y') = \frac{1}{n} C_{2,S'}$, where [59]:

$$C_{2,S'} = C_{2,S} + \frac{n}{n+1}(x_{n+1} - \overline{x})(y_{n+1} - \overline{y}). \tag{3.7}$$

Substituting the sums in the Equation 3.2 for $C_{2,S}$, $M_{2,X}$ and $M_{2,Y}$, we get the final correlation coefficient:

$$r_{X,Y} = \frac{C_{2,S}}{\sqrt{M_{2,X}}\sqrt{M_{2,Y}}}. \tag{3.8}$$

Using these recurrent formulas, one can calculate correlation coefficients, even for a large number of statistical samples (measured power traces), assuring the numerical stability of the computation [59].

This approach, allowing online and direct update of all the values every time a new sample is added to the dataset, is well-suited for parallel CPU computations. This algorithm can be parallelized just as easily, as the naïve one-pass algorithm.

Implementation The incremental one-pass algorithm keeps a matrix and vectors of the same values as the two-pass algorithm does (unlike naïve one-pass algorithm, which keeps sums of powers), but rather than computing them in two passes, it uses recurrent update formulas, mentioned earlier.

In a one-pass, sample mean vectors $\overline{\mathbf{X}} = [\overline{x_1}, ..., \overline{x_s}]$ and $\overline{\mathbf{Y}} = [\overline{y_1}, ..., \overline{y_c}]$ are calculated using the Equation 3.5. In the same pass, variance vectors $\mathbf{M_{2,X}} = [M_{2,X_1}, ..., M_{2,X_s}]$ and $\mathbf{M_{2,Y}} = [M_{2,Y_1}, ..., M_{2,Y_c}]$ are computed using Equations 3.5 and 3.6 and the covariance matrix $\mathbf{K}$, where $K_{i,j} = C_{2,S_{i,j}}$ and $S_{i,j} = \{(x, y) | x \in X_i, y \in Y_j\}$, gets computed using the Equation 3.7. The final matrix $\mathbf{C}$ with Pearson correlation coefficients is then obtained using the Equation 3.8.

While this algorithm is numerically stable [59], it keeps the online character of the naïve one-pass algorithm and allows optimized distributed memory access and thus a good time performance, as well as a good memory space performance. The bottleneck of the implementation remains in the update of the covariance matrix $\mathbf{K}$, allowing to parallelize easily (e.g. using OpenMP). The algorithm is also capable of stopping, giving out the results and continuing the computation with more statistical samples, thus, for example, allowing to find out the number of traces needed to obtain the correct key.

3.1.1.4 GPU-Accelerated Implementation

As the correlation computations for side-channel analysis require a large amount of the same computations over different data, i.e. it follows single instruction, multiple data principle, using a GPU to accelerate the computations is an obvious option.

Unfortunately, the most promising incremental one-pass approach proved not to be ideal for implementation on GPU, as it requires to update all the variables (averages, variances, covariances) with every measurement processed, resulting in poor workload balance.

Therefore, an algorithm combining the two-pass approach and the incremental one-pass approach is proposed instead. In the first pass, sample mean vectors $\overline{\mathbf{X}} = [\overline{x_1}, ..., \overline{x_s}]$ and $\overline{\mathbf{Y}} = [\overline{y_1}, ..., \overline{y_c}]$ are calculated using the Equation 3.5. In the same pass, variance vectors $\mathbf{M_{2,X}} = [M_{2,X_1}, ..., M_{2,X_s}]$ and $\mathbf{M_{2,Y}} = [M_{2,Y_1}, ..., M_{2,Y_c}]$ are computed using Equations 3.5 and 3.6, as in the case of the incremental one-pass implementation described in Section 3.1.1.3. In the second pass, the covariance matrix $\mathbf{K}$ of size $s \times c$, where $K_{i,j} = \text{cov}(X_i, Y_j)$ is computed and the correlation matrix $\mathbf{C}$ is easily derived, as it is described in Section 3.1.1.1.

Since the GPU acceleration is done using OpenCL with multiplatformity in mind, there are no vendor-specific optimizations used. It is fair to assume that with such optimizations present (e.g. using cuBLAS, the nVidia computational library optimized at assembler level for the maximum performance), the running time would be lower. Also, low-end GPUs are not well suited for double-precision computation; single-precision computation may be much faster, however, numerical instabilities may occur for large datasets.

3.1.2 Higher-Order Analysis

In this section, we focus on univariate higher-order analysis, as it is described in Section 2.1.5.2. Two different approaches to the univariate higher-order analysis are described in the following subsections.

3.1.2.1 Naïve Multiple-Pass Approach

The most straightforward way to perform any kind of higher-order analysis is to preprocess the power traces according to the Equations 2.2 and 2.3 mentioned in Section 2.1.5.2, and then run the first-order analysis algorithm (e.g. CPA or t-test). The biggest drawback of this approach is the same as in the case of any multi-pass statistical algorithm: adding new statistical samples to the statistical set requires launching the whole computation from the beginning (since the sample mean, used for the preprocessing, changes with new data).

3.1.2.2 Incremental One-Pass Approach

Formulas allowing for robust one-pass higher-order analysis are given in [17, 16]. These take advantage of the fact that **mean and variance of the higher-order preprocessed power traces** can be expressed using **central moments of the raw power traces**. To estimate the n-th central moment, formulas similar to the ones mentioned in Section 3.1.1.3 are presented, allowing to estimate the n-th central moment of the set created by adding one or more new traces to an existing set. These formulas operate with mean-free values as well and therefore avoid numerical instabilities.

Unfortunately, to update the estimate of n-th central moment, all the i-th central moments, $2 <= i < n$, are required. To express the variance of the d-th-order power traces, the $2d$-th and d-th central moments are necessary. In consequence, for the d-th order analysis, all the $2d$ central moments need to be kept and updated with every new power trace.

To estimate the covariance between the higher-order power traces and power predictions, similar formulas are given as well. For a d-th order analysis, d "adjusted central sums" need to be kept and updated.

These facts represent both time and memory penalties in comparison with the preprocessing approach – a cost for not being required to run the whole computation again with new power traces available. In the case of very large datasets, this approach would be a preferred option. Unlike the preprocessing approach, these formulas allow for online processing of power traces. It is also notable, that correlation matrices/t-values from central moments can be derived from first-order up to the order of the computation; unlike the preprocessing approach, where only the selected order is computed.

3.1.3 Welch's t-test

For computation of Welch's t characteristic and degrees of freedom v, sample means and variances need to be computed (or higher moments in the case of higher-order test). The implementation of Welch's t-test is easily derived from first-order incremental one-pass approach described in Section 3.1.1.3 or higher-order incremental one-pass approach described in Section 3.1.2.2 respectively.

In comparison with correlation analysis, Welch's t-test does not demand covariances to be computed, which is the bottleneck of correlation analysis computations. Therefore, the evaluation of Welch's t-test is much less computational demanding.

3.1.4 Results

In this section, we compare the time performance and the scalability of implementations stated earlier. The CPU running times were partially measured on two different CPUs: Intel i5 2400 (hereafter referred to as CPU1) and Intel Xeon E312xx (CPU2). Additionally, nVidia GeForce GTX 1050 was used for GPU computations using OpenCL. Power traces are signed 16-bit integers, power predictions unsigned 8-bit integers, the computation is done in double-precision floating-point. All the algorithms are implemented in C/C++ for the sake of the best performance. Wall-clock time is presented and hereafter referred to as running time. Parallel computational threads are hereafter referred to as threads.

3.1.4.1 First-Order CPA

In this section, we compare the time performance and the scalability of the above-described first-order implementations of correlation power analysis of AES-128 using three different approaches.

Two-Pass Approach Our original two-pass implementation, written in C++ using STL, combines the computation of correlations and the power model computation. The

Table 3.1: Running time of the two-pass algorithm for various numbers of power traces and samples, in seconds (CPU1)

# of traces / # of samples per trace	100	1k	10k	100k
10	0.009	0.127	1.882	73.56
100	0.084	1.361	16.70	794.5
1,000	0.704	11.58	149.6	6 964

Table 3.2: Running time of the naïve one-pass algorithm for various numbers of power traces and 1,000 samples per trace, in seconds (CPU1)

# of traces / # of threads	100	1k	10k	100k	1M
1	0.253	2.035	19.80	198.7	2006
2	0.162	1.097	10.42	103.7	1049
4	0.116	0.617	5.648	56.47	587.7

library containers and a poor memory cache usage, caused by the organization of the computation, lead to a very poor time performance.

Table 3.1 presents the running time for various numbers of power traces and samples per trace, using a single thread.

As can be seen, for 100,000 power traces and 1,000 samples per trace, the computing time gets unbearably high. This leads to the need for other implementations.

Naïve One-Pass Approach The optimized naïve one-pass algorithm, implemented in C language, takes the measured data and the pre-computed power model (see Section 3.1.4.1), and results with the correlation matrix for each part of key candidate, and the best key candidate based on the max/min correlation coefficient.

Since the naïve one-pass algorithm is well parallelizable, our implementation uses OpenMP to achieve better time performance.

Tables 3.2 and 3.3 present the running time for various numbers of power traces, samples per trace and numbers of threads.

As can be seen, the naïve one-pass algorithm has good scalability. Also, the computation time grows linearly with the number of traces.

The scalability of the algorithm gets better with a higher number of samples per trace. This is due to the fine-grain parallelism, implemented over the computation of the matrix sized $s \times c$, where s is the number of samples per trace.

Figure 3.1 demonstrates the scalability of the naïve one-pass algorithm. Compared to our original two-pass algorithm, our implementation of the naïve one-pass algorithm gets up to 35× faster when processing 100,000 power traces with 1,000 samples per trace.

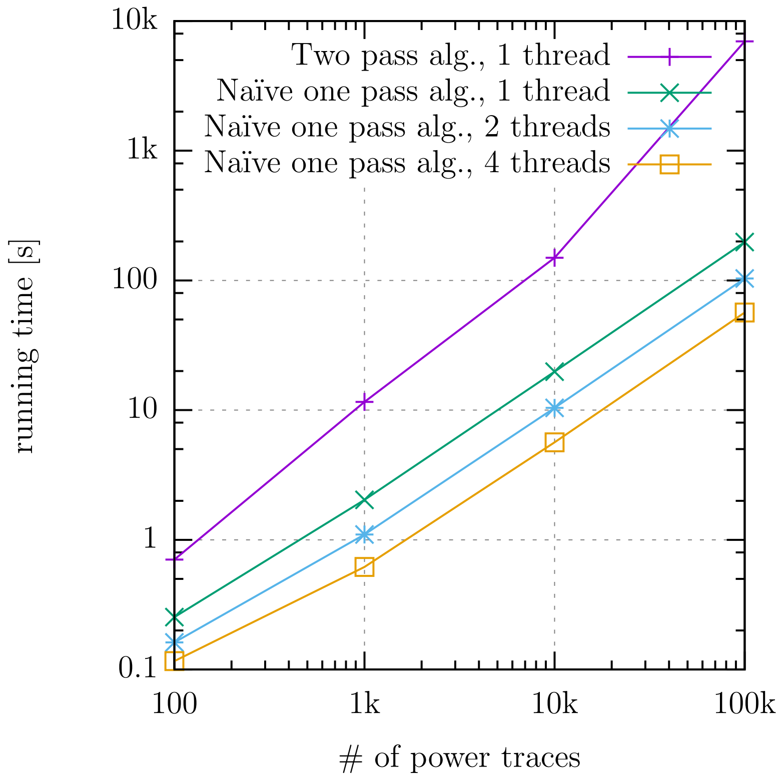


Figure 3.1: Comparison of the time performance of the two-pass and the naïve one-pass algorithm for various numbers of power traces, 1,000 samples per trace, and in a case of the naïve one-pass algorithm, various numbers of threads (CPU1)

Table 3.3: Running time of the naïve one-pass algorithm for various numbers of samples per trace and 100,000 power traces, in seconds (CPU1)

# of samples per trace / # of threads	10	100	1,000
1	2.616	19.53	198.7
2	2.165	10.93	103.8
4	2.271	6.795	56.47

Table 3.4: Running time of the incremental one-pass algorithm for various numbers of power traces and 1,000 samples per trace, in seconds (CPU1)

# of traces / # of threads	100	1k	10k	100k	1M
1	0.292	2.435	23.75	236.7	2383
2	0.184	1.320	12.69	124.9	1267
4	0.132	0.790	7.134	69.52	719.1

Table 3.5: Running time of the incremental one-pass algorithm for various numbers of samples per trace and 100,000 power traces, in seconds (CPU1)

# of samples per trace / # of threads	10	100	1,000
1	4.074	24.69	236.7
2	3.720	14.43	124.9
4	3.707	10.99	69.52

Table 3.6: Running time of the incremental one-pass algorithm for various numbers of power traces and 2,000 samples per trace, in seconds (CPU2)

# of power traces	2,000	4,000	8,000	16,000	32,000
1 CPU thread	18	41	72	143	289
2 CPU threads	10	20	41	85	169
4 CPU threads	6	12	25	51	105
GPU	2	5	10	19	40

Incremental One-Pass Approach Our implementation of the incremental one-pass algorithm, also written in C language, has very similar characteristics, as the naïve one-pass algorithm.

Tables 3.4 and 3.5 present the running time of the incremental one-pass algorithm for various amounts of power traces, samples per trace and threads.

As can be seen in Figure 3.2, the incremental algorithm is approximately 1.2× slower, than the naïve one-pass algorithm.

Table 3.6 presents a comparison of running times of the OpenMP implementation (using various numbers of threads) and the OpenCL GPU-accelerated implementation of incremental one-pass algorithm for various amounts of power traces. Running times of GPU-accelerated version includes data transfers between the main memory and GPU memory.

Separate Power Model Computation Our original two-pass implementation computes both the power leakage model and the correlation coefficients. For the one-pass implementations, we have separated the power model computation from the correlation computation.

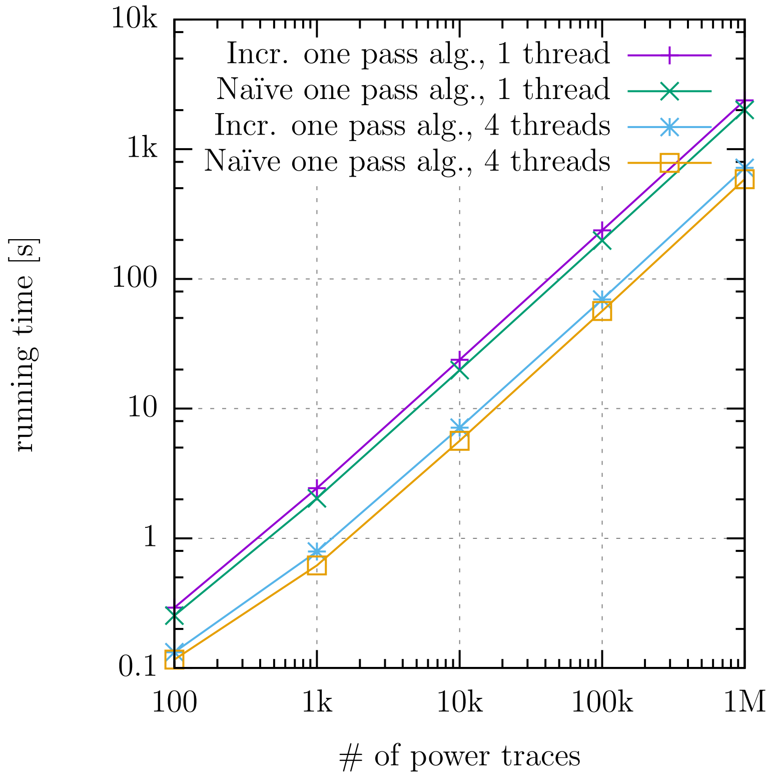


Figure 3.2: Comparison of the time performance of the naïve and the incremental algorithm for various numbers of power traces and 1,000 samples per trace, using 1 or 4 threads (CPU1)

Table 3.7: Running time of the power model precomputation, in seconds (CPU1)

# of traces / # of threads	100	1k	10k	100k	1M
1	0.005	0.052	0.509	5.097	49.80
2	0.002	0.027	0.266	2.665	25.22
4	0.003	0.015	0.141	1.355	13.13

The power model computation implementation takes the plaintext/ciphertext, used while measuring, and results with the pre-computed power model organized for the best correlation computation performance. Its running time depends linearly on the number of power traces and the scalability is quite satisfactory.

In comparison with correlation coefficients computation, one can neglect the power leakage model computation time as insignificant, as can be seen in Table 3.7.

Table 3.8: Running time of the preprocessing of power traces for 2,000 samples per trace, in order to perform arbitrary-order analysis, in seconds (CPU2)

# of power traces	2,000	4,000	8,000	16,000	32,000
preprocessing 1 CPU thread	<1	1	2	4	8
preprocessing + CPA 1 CPU thread	18	42	74	147	297
preprocessing + CPA 4 CPU threads	6	13	27	55	113
preprocessing + CPA GPU	2	6	12	23	48

3.1.4.2 Higher-Order Analysis

Both presented approaches to the higher-order analysis will be discussed and examined in this section: using preprocessing and using one-pass higher-order formulas.

Preprocessing The preprocessing approach to the higher-order analysis consists of preprocessing of the power traces according to the order of the attack, as described in Section 3.1.2.1, and running of the first-order analysis, as is evaluated in Section 3.1.4.1. The order of the attack does not affect the complexity of the preprocessing. Running time for 2,000 samples and various numbers of power traces is presented in Table 3.8.

As mentioned before, obtaining results for a different order, as well as adding new power traces to the dataset, requires running the preprocessing and analysis again from the beginning. This drawback is solved by higher-order one-pass formulas.

One-Pass Formulas Using formulas in [17], side-channel analysis up to arbitrary order may be performed in a way that allows for simple addition of new power traces, and for merging results based on processing different power traces sets together. Unlike the first-order attack and the higher-order preprocessing approach, this algorithm requires $2d$ central moments to be computed for each sampling point. Furthermore, in the case of CPA attack, d “adjusted central sums”, which serve to compute the covariance between higher-order power traces and power predictions, must be computed as well. These facts represent a significant performance penalty, as can be seen in Table 3.9 and Figure 3.3, where higher-order CPA running time is presented. The running time grows linearly with the number of power traces. Running time for various numbers of samples per trace is in Table 3.10.

3.1.4.3 Welch’s t-test

Welch’s t-test, unlike CPA, does not require covariance, i.e. d “adjusted central sums” matrices, to be computed. Since this is the bottleneck of the CPA computation, the t-test computation is much faster for the same number of power traces. Running time of first-order and higher-order t-test computation on 1 CPU thread is presented in Table 3.11.

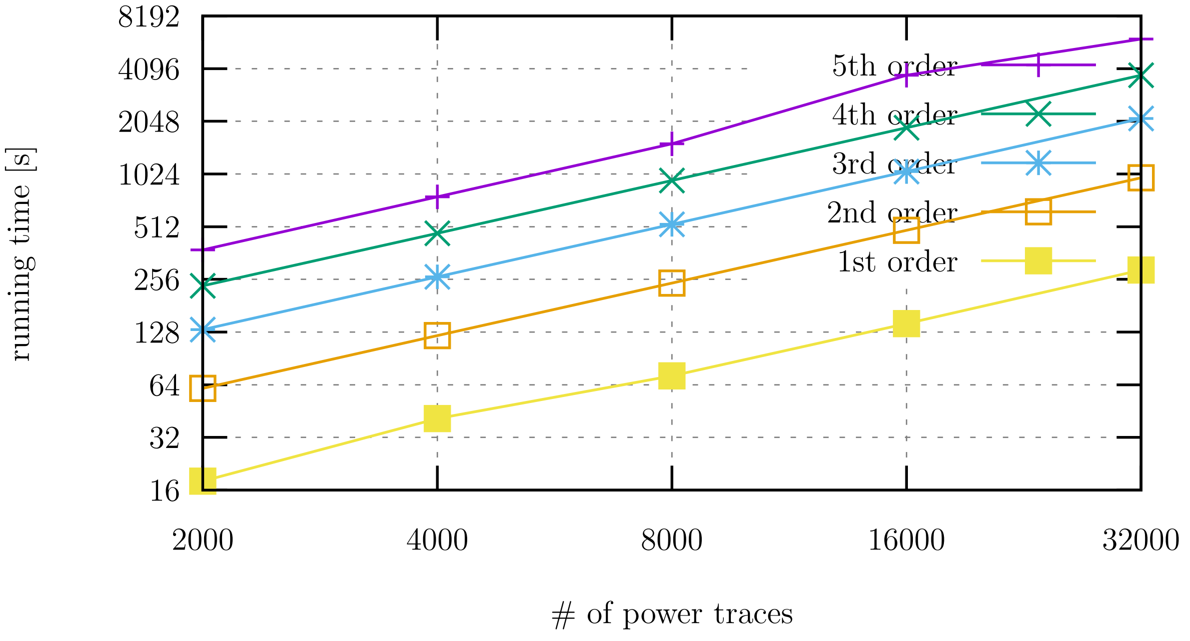


Figure 3.3: Comparison of the time performance of the CPA for various orders of the attack, using one-pass formulas (CPU2)

Table 3.9: Running time of the higher-order CPA one-pass algorithm for 2,000 samples per trace, in seconds (CPU2)

(a) 1 CPU thread

# of power traces	2,000	4,000	8,000	16,000	32,000
1st order	18	41	72	143	289
2nd order	61	122	244	489	975
3rd order	132	265	529	1056	2124
4th order	235	469	941	1881	3758
5th order	377	759	1523	2995	6043

(b) 4 CPU threads

# of power traces	2,000	4,000	8,000	16,000	32,000
1st order	6	12	25	51	105
2nd order	19	37	77	152	304
3rd order	43	79	158	310	618
4th order	69	138	277	547	1101
5th order	103	204	418	846	1633

Table 3.10: Running time of the higher-order CPA one-pass algorithm for 32,000 power traces and various numbers of samples per trace, 1 CPU thread, in seconds (CPU2)

# of samples per trace	500	1,000	2,000
1st order	79	152	289
2nd order	243	485	975
3rd order	517	1038	2123
4th order	908	1803	3758
5th order	1411	3797	6043

Table 3.11: Running time of first-order and higher-order Welch's t-test one-pass algorithm for 2,000 samples per trace and various numbers of power traces, 1 CPU thread, in seconds (CPU2)

# of power traces	32,000	64,000	128,000	256,000	512,000
1st order	<1	1	3	6	12
2nd order	1	3	6	12	25
3rd order	2	5	11	23	46
4th order	4	9	19	41	80
5th order	7	14	28	57	117

3.1.5 Conclusions

We have compared three algorithms for calculating Pearson correlation coefficients of two large matrices. This calculation is necessary for performing the CPA. The comparison was based on both mathematical and performance properties.

A simple two-pass algorithm proved to be slow, high memory demanding and not well parallelizable. Compared to that, a naïve one-pass algorithm is much faster, low memory demanding and well parallelizable, but not numerically stable. Finally, we presented an incremental version of the one-pass algorithm, which is, unlike the naïve one-pass algorithm, numerically stable, while being only about 20% slower. This incremental version of the one-pass algorithm also preserves other advantages, including low memory demands, satisfactory scalability and an ability to resume the stopped calculation with more added power traces.

While many researchers may still use a straightforward, two-pass algorithm or numerically unstable naïve one-pass algorithm, our comparison proves the incremental one-pass algorithm to be the best choice for researchers who are mounting CPA attack on any cryptographic device. We have not found any such comparison, providing tangible figures, in the open literature. The incremental one-pass implementation helps us to speed up CPA significantly. It also allows for an analysis of an unlimited amount of power traces thanks to its constant memory demands.

Additionally, we presented an algorithm combining two-pass and incremental one-pass algorithm, which is suitable for a GPU-accelerated implementation. This implementation provides additional speedup of the CPA computations even with a low-end GPU.

We have also compared two different approaches to higher-order side-channel analysis regarding their mathematical, performance and usage properties.

Naïve approach to the higher-order analysis, i.e. preprocessing, performs nearly as fast as in the case of the first-order analysis and the computation time does not depend on the selected order. However, for very large datasets, this approach may become troublesome because of its high memory demands and usage properties. Addition of new power traces requires running the whole computation from scratch.

Using one-pass formulas for the same purpose allows for simple addition of new power traces to the dataset. Unfortunately, both memory demands and time complexity of this approach grow with the order of the analysis. On the other hand, all orders up to the specified order can be easily derived from the algorithm working variables. These usage properties make the one-pass approach a better choice for very large power trace datasets and experimental work.

Higher-order analysis is necessary for attacking cryptographic implementations secured against side-channel attacks, e.g. when the masking is employed. It is also necessary for evaluating the leakage of cipher implementations and the efficiency of countermeasures. Our optimized implementations allow us to choose the right approach for a specific task and significantly speed up these computations.

Additionally, we evaluated the running time of Welch's t-test using the incremental one-pass approach for both first-order and higher-order analysis. Since Welch's t-test does not require computation of covariance, which is the bottleneck of CPA computations, its running times are dramatically lower for any order.

3.2 Aggregation of Traces

In this section, we present our work about speeding up the correlation computations by decreasing the number of samples per power trace. The main idea of this research is to somehow aggregate the traces. As the power model usually represents some intermediate value exhibited in the device during a single clock cycle, the most straightforward way is to integrate the traces by time for each clock cycle. Therefore, the number of samples per trace will be close to the number of clock cycles of one encryption (usually tens) instead of the usual hundreds or thousands. We show how this simple trace aggregation affects the efficiency of the attack. Specifically, we show how it affects the number of traces needed to reveal the correct key and how it affects the overall calculation time of the attack on three different FPGA platforms.

We introduce the FPGA platforms, AES implementation and the experiment setup in Section 3.2.1. Detailed results are presented in Section 3.2.2. We summarize and conclude the experiment in Section 3.2.3.

3.2.1 Experiment Setup

In this section, we comprehensively describe the process of trace aggregation. We also introduce the FPGA platforms and the AES implementation we used.

3.2.1.1 Hardware Platforms

We performed the experiment on three different FPGA platforms:

- Evariste III with Altera Cyclone III module — an open modular cryptographic platform developed by Fischer et al. [61]
- Sakura-G — standard evaluation board for side-channel attacks based on Xilinx Spartan-6 FPGA [62]
- DPAboard — open board for differential power analysis with Xilinx Artix-7 FPGA developed by Bartík et al. [1]

The whole experiment is the same for all the platforms, only the clock frequencies differ. Evariste III and Sakura-G run at 5 MHz while DPAboard runs at 6.25 MHz.

PicoScope 6404D was used for the measurement. The sampling rate was set to 625 MS/s. We collected 2000 samples per one power trace.

3.2.1.2 AES

AES cipher [9], specifically its 128-bit variant, is used for the experiment as it is one of the most commonly used block ciphers. The 128-bit variant of AES consists of one initial round and ten regular rounds. The initial round performs only an initial key addition. Each regular round uses a different round key derived from the initial one. The regular rounds

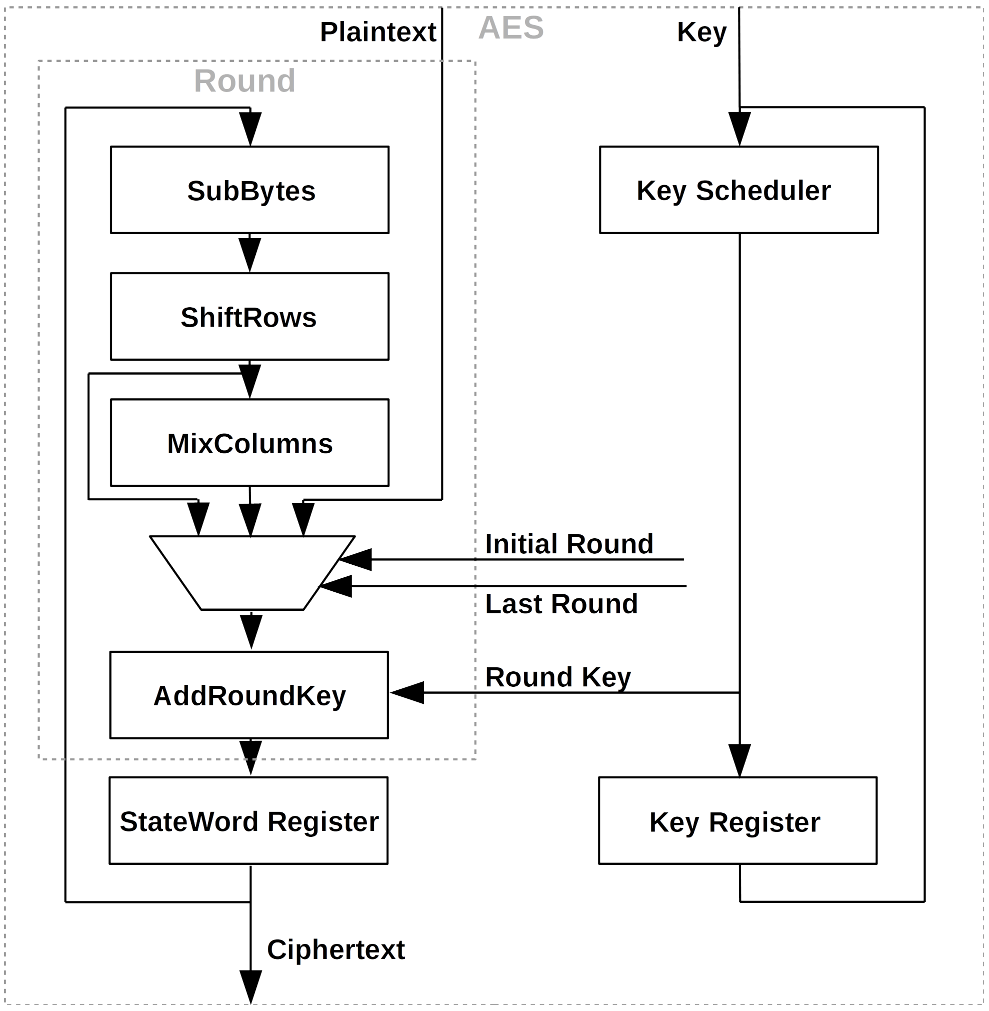


Figure 3.4: Diagram of AES implementation

consist of a key addition (AddRoundKey function), a non-linear transformation (SubBytes function), and two linear transformations (ShiftRows and MixColumns functions) with an exception of the last round, where the MixColumns function is omitted.

In our implementation, each round is processed by a combinational logic during one clock cycle. Therefore, the whole encryption takes 11 clock cycles. A diagram of our AES implementation can be seen in Figure 3.4.

3.2.1.3 Aggregation

The main idea of our trace aggregation approach is based on fact, that the intermediate value used by the power model and the real power consumption are not correlated in one specific power sample, but the correlating information is spread through the whole clock cycle the intermediate value is processed within. As the number of samples is the same during each clock cycle, we do not need to take it into account and the integration process

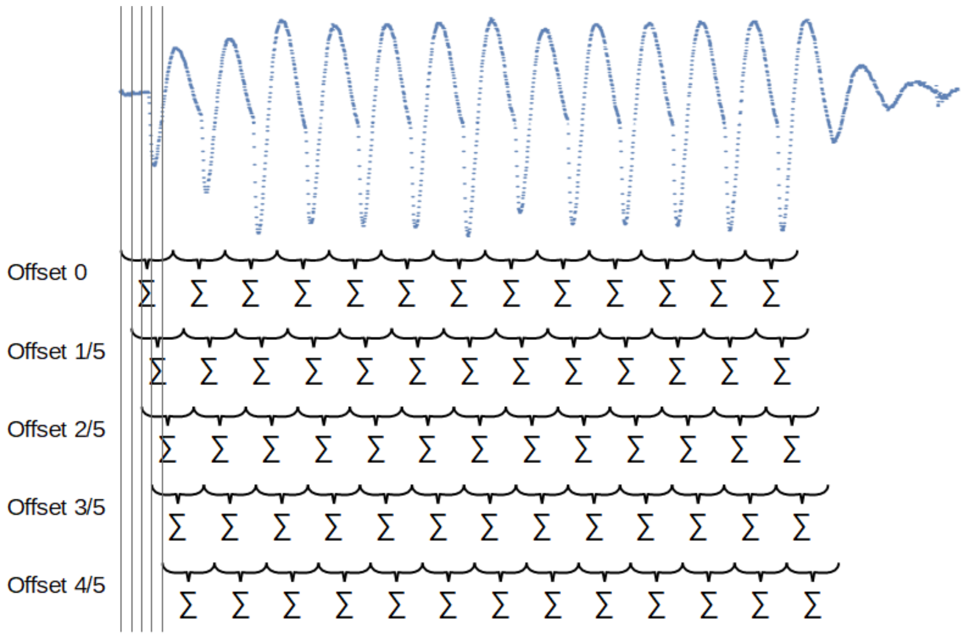


Figure 3.5: Visualization of trace aggregation using various offsets

can be simplified to summation. Therefore, we preprocess each power trace by summarizing the samples corresponding to each one clock cycle of the encryption. In our case, each clock cycle corresponds to 125 samples (Evariste III and Sakura-G) or 100 samples (DPAboard).

We added up samples of each clock cycle of the encryption (11 cycles) and one before and one after the encryption. We obtained 13 samples per trace instead of the original 2000 samples this way.

We also aggregated the traces with various starting samples that we call *offsets*. For example, with offset 0, the first sample of the integration is the first sample of the clock cycle; with offset 25, the first sample of the integration is the 26th sample of the clock cycle, therefore the last sample is the 25th sample of the next clock cycle. We used five offsets: 0, 25, 50, 75, and 100 in the case of Evariste III and Sakura-G, and 0, 20, 40, 60, 80 in the case of DPAboard. For simplicity, we will use relative designations of these offsets: 0, 1/5, 2/5, 3/5, and 4/5. Visualization of the offsets can be seen in Figure 3.5.

To summarize, we aggregated each power trace using 5 different offsets. Therefore, we gained 5 traces of 13 samples for each of the original trace.

3.2.1.4 Evaluation

For each of the hardware platforms, we measured 30 sets of power traces. Then each collected trace was aggregated using the method described above. Therefore, we obtained 5 aggregated power traces and the original one for each measured trace.

Having all the data collected, we performed the computational part of the attack. For each of the $3 \times 30 \times (5 + 1)$ sets of power traces (3 platforms; 30 sets; 5 offsets + 1 non-

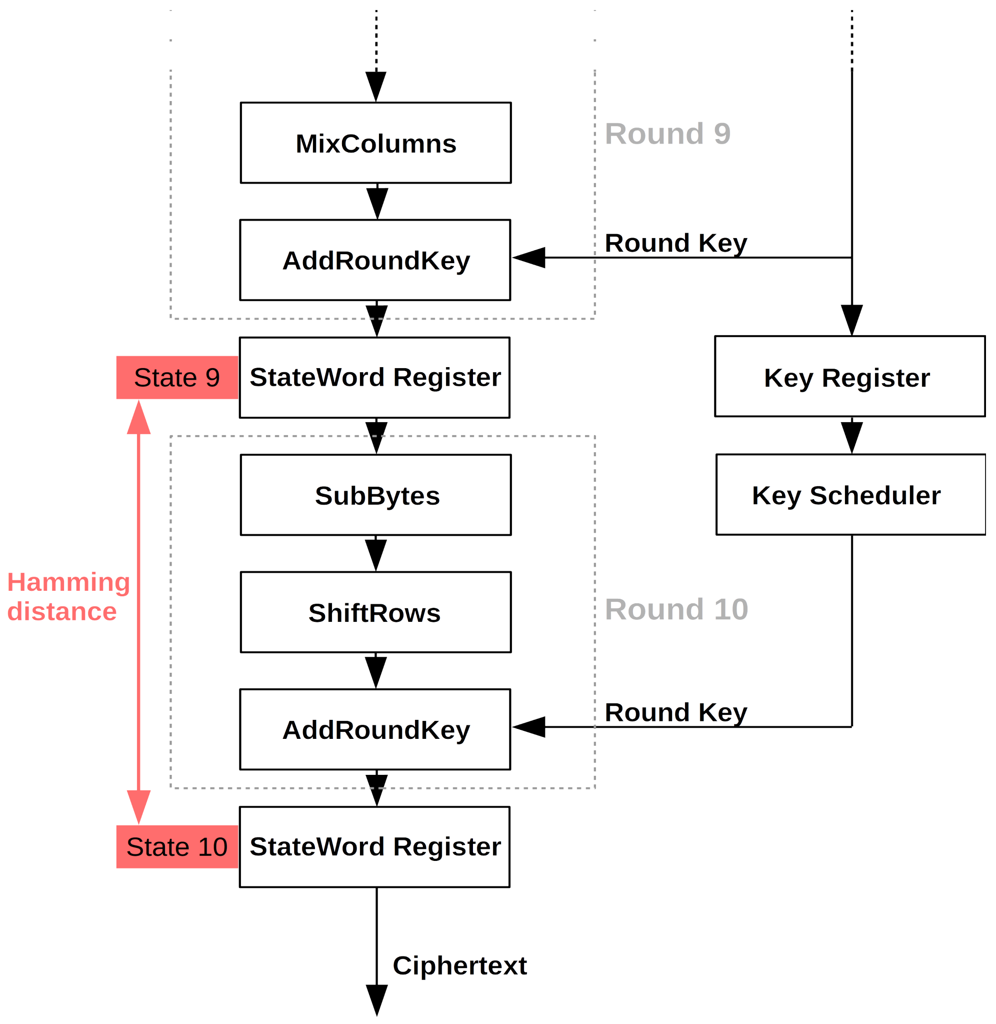


Figure 3.6: Diagram of the used power model

aggregated) we found the minimal number of power traces needed to reveal the correct cipher key using Hamming distance between output of the 9th round and output the 10th round (ciphertext), as is depicted in Figure 3.6. These collected results are presented in Section 3.2.2.

We also compared the calculation time for various numbers of traces to demonstrate the efficiency of our approach. All time measurements were done on a computer equipped with Intel i5 6440HQ CPU using the first-order incremental one-pass algorithm proposed in Section 3.1.1.3.

3.2.2 Results

In this section, we present the results obtained from the experiment proposed in Section 3.2.1. First, we show how our method affects the number of power traces needed to reveal

Table 3.12: Medians of numbers of power traces necessary to obtain the correct cipher key for various FPGAs using the original traces and aggregated traces with various offsets

Traces \ Platform FPGA		Evariste III Cyclone III	Sakura-G Spartan-6	DPAboard Artix-7
Original		779	1048	1967
Integrated	Offset 0	666	2202.5	1347.5
	Offset 1/5	714.5	885	1359
	Offset 2/5	737	993.5	1353
	Offset 3/5	692.5	1722.5	1470.5
	Offset 4/5	683.5	9718	1501.5

the correct key. Afterwards, we present a time comparison of common methods and our new approach.

3.2.2.1 Power Traces Needed

We successfully performed the attack using the proposed approach with all the offsets and on all the platforms. We compared the minimal number of traces needed to reveal the key using the original traces and the preprocessed ones with various offsets. In Table 3.12, we can see medians of minimal needed traces calculated from the results of all trace sets.

In the case of Evariste III and DPAboard, the number of traces needed is even lower than with the original traces for all offsets.

In the case of Sakura-G, we can see significant differences between the offsets. Using offsets 1/5 or 2/5 the results are slightly better than with the original traces, but in the case of offset 4/5, the number of traces needed is more than 9 times higher. On the other hand, the time saved by our method overcomes this increase, as is shown below.

Detailed results can be seen in Figure 3.7 (for Evariste III), Figure 3.8 (Sakura-G), and Figure 3.9 (DPAboard).

3.2.2.2 Time Comparison

Considering the algorithm being linearly dependent on the number of samples, we expected our approach to be significantly more time-efficient than the original method. As we can see in Table 3.13, the correlation calculation using 13 samples per trace is approximately 50 times faster than the one using 2000 samples per trace. As the duration of the integration is also much lower, the summary time consumption using our approach is roughly 40 times lower than with the original approach. In Figure 3.10, we can see that this improvement is constant with the number of power traces. To prevent the effect of badly chosen offset, one can repeat the attack for multiple offsets and still perform the attack many times faster than with the original approach.

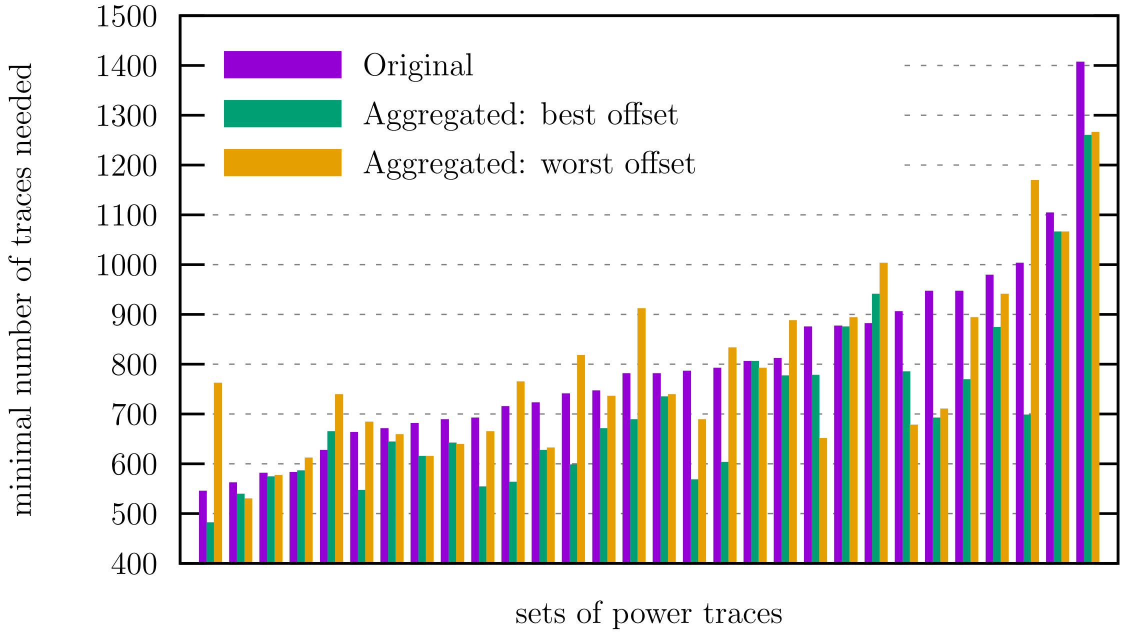


Figure 3.7: Plot of minimal power traces necessary to reveal the cipher key for the original power traces and the aggregated power traces using the best and the worst offset, Altera Cyclone III FPGA on Evariste III, 30 sets of traces (sorted by values for the original traces)

Table 3.13: Running time of correlation calculation using 2000 or 13 samples and running time of integration preprocessing for various numbers of power traces in seconds

Procedure \ # traces		1K	10K	100K	1M
Correlation	2000 samples	1.8	17.6	177	1850
	13 samples	0.06	0.36	3.7	36
Integration		0.03	0.12	1.11	11

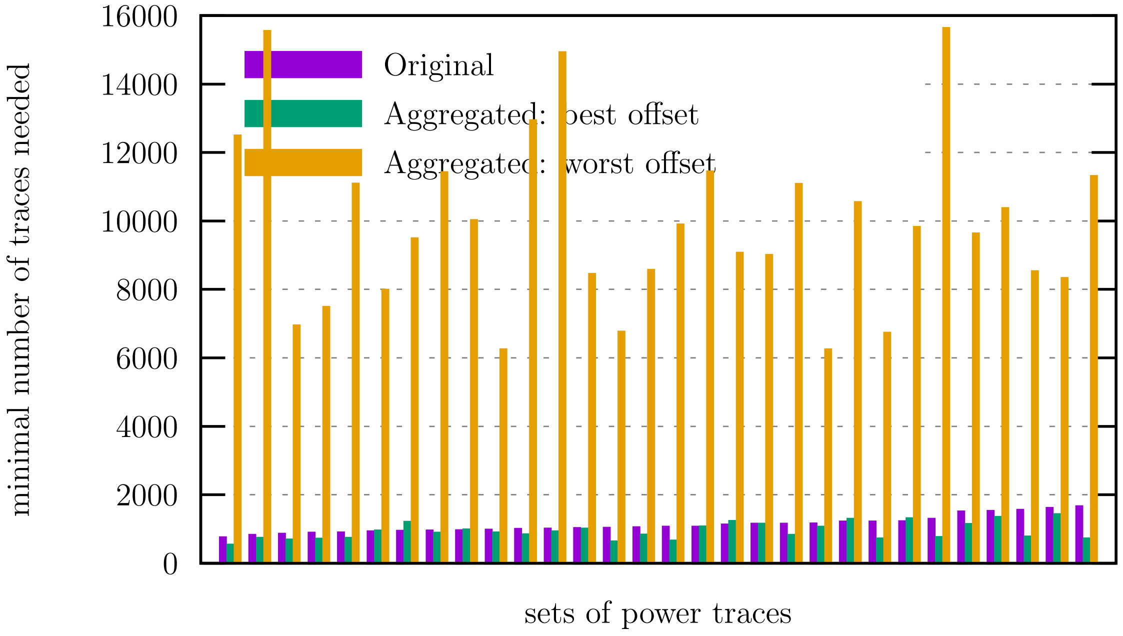


Figure 3.8: Plot of minimal power traces necessary to reveal the cipher key for the original power traces and the aggregated power traces using the best and the worst offset, Xilinx Spartan-6 FPGA on Sakura-G, 30 sets of traces (sorted by values for the original traces)

3.2.3 Conclusion

We propose a new approach to fast evaluation of correlation coefficients when performing correlation power analysis. It is based on power trace preprocessing. In each trace, we integrate samples corresponding to each clock cycle of the encryption. This preprocessing is easy to implement and marginally time demanding in comparison with the correlation calculations. This approach makes the correlation calculations much faster as it drastically decreases the number of samples per trace. We successfully confirmed these assumptions in Section 3.2.2. We show, that the overall time of the calculation part of the attack is approximately 40 times lower with our approach in comparison with the original method.

We also show, how the integration affects the number of power traces needed to obtain the correct key. Three different platforms were used. On two platforms, the number of traces was even lower than without the preprocessing. Also, on the third platform, our approach is more time-efficient even though more power traces are demanded when using some particular offsets.

Considering these facts, the presented method was confirmed as an excellent way to speed up the correlation power analysis.

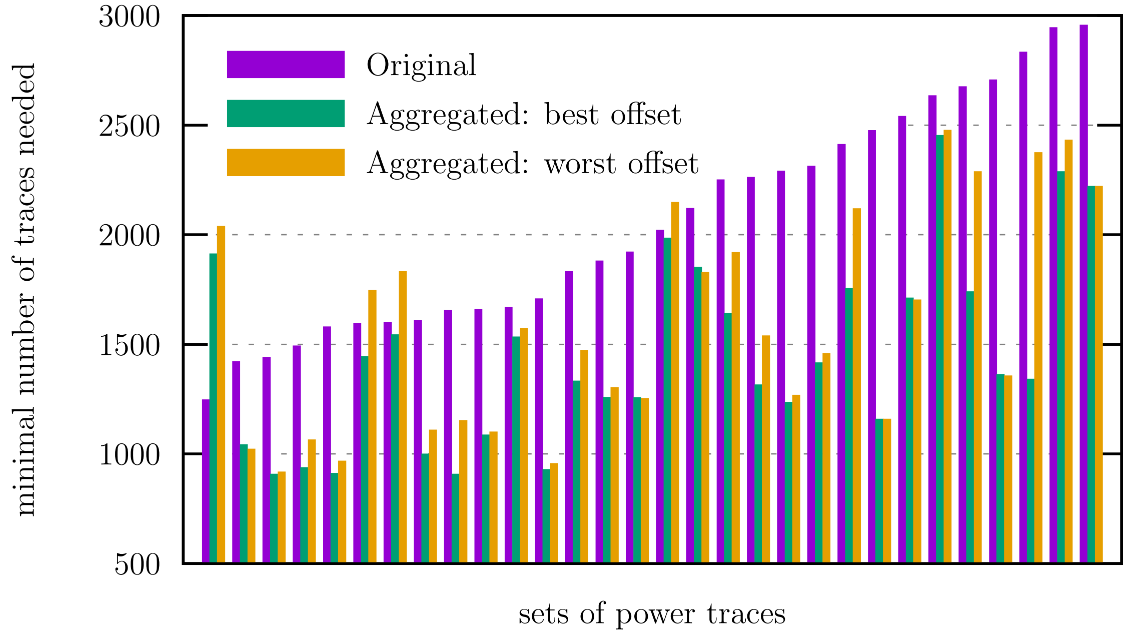


Figure 3.9: Plot of minimal power traces necessary to reveal the cipher key for the original power traces and the aggregated power traces using the best and the worst offset, Xilinx Artix-7 FPGA on DPAboard, 30 sets of traces (sorted by values for the original traces)

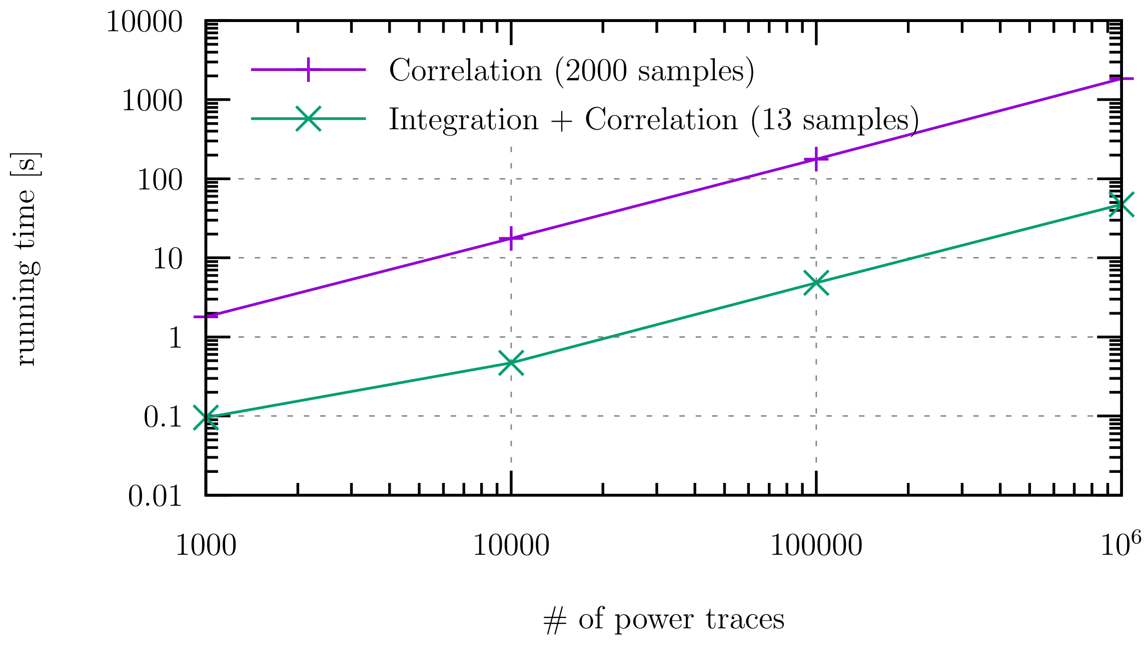


Figure 3.10: Plot of running times of the original method and our approach for various numbers of power traces

3.3 Key Candidate Selection

The examination and comparison of correlation traces, in order to obtain the correct key candidate, can be done visually by the attacker. However, an algorithmization of this final step of the CPA attack is highly relevant for batch attacking and the automatic evaluation of the side-channel attack. The naïve way to automate this process is simply selecting the key candidate which maximizes the Pearson correlation coefficient. In this work, we propose an algorithmic way of extracting the key candidate based on a significant change in the correlation trace rather than on the correlation coefficient magnitude.

We describe our approach and present solution for noise filtering and computational aspects of edge detection in Section 3.3.1. Results of our experimental work is presented in Section 3.3.2. We summarize and conclude our results in Section 3.3.3.

3.3.1 Our Approach

Our approach extends correlation power analysis, i.e. non-profiled side-channel attack, and it is based on detecting a sudden change (edge) in a correlation trace (a time series of a correlation coefficient). With this approach, we are able to

- significantly *reduce* the number of acquired power traces necessary to successfully mount an attack in a noisy environment, and
- in some cases make the attack even *feasible*.

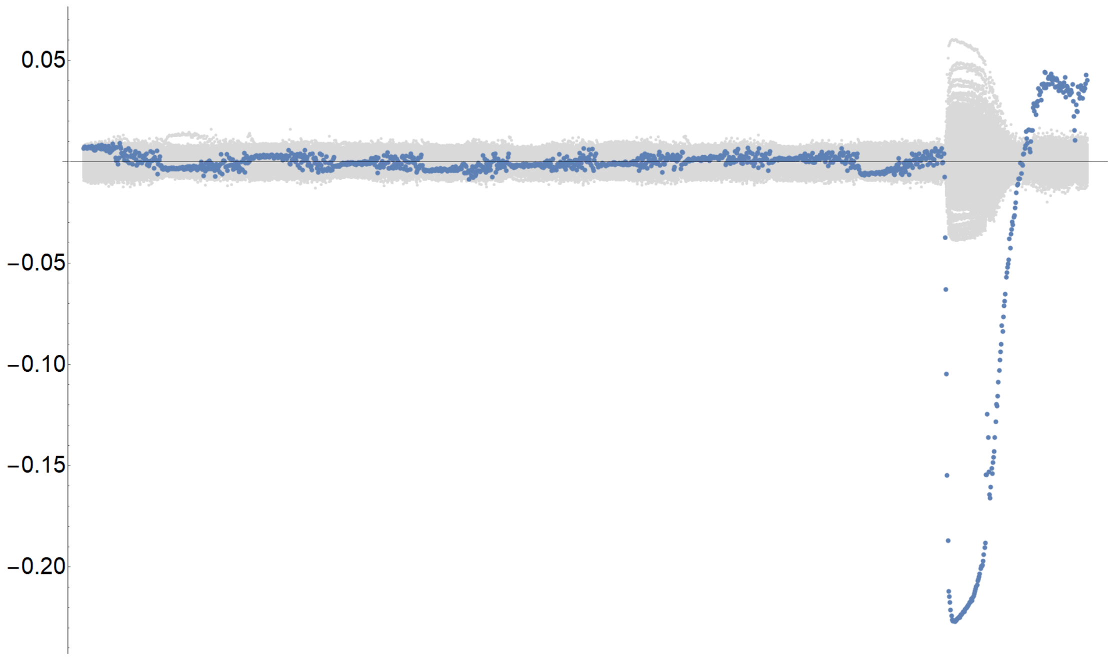
The reduction of the number of power traces reduces both the acquisition (measurement) time and the time of the analysis.

3.3.1.1 Motivation

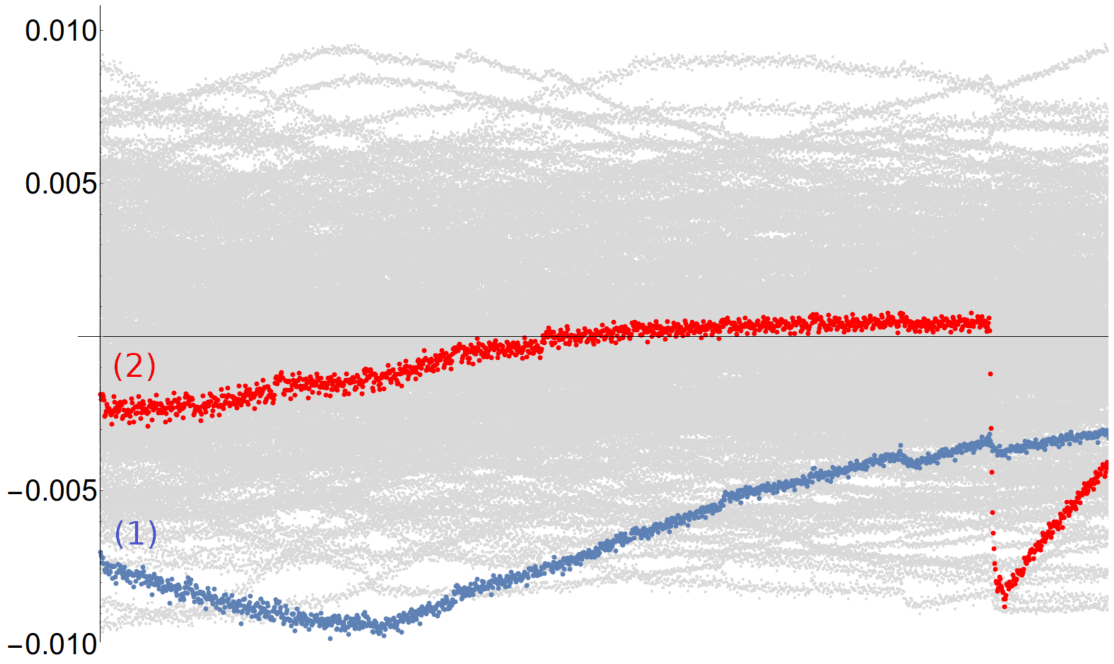
To evaluate the correlation analysis results, the attacker would visually examine the correlation traces, looking for specific anomalies that might give him a hint about the right cipher key. The naïve way to automate this process might be searching for the correlation curve with the strongest (positive or negative) correlation and thus relevant key candidate.

As we described in Section 2.1.4, correlating our predictions with each sample point in the power trace gives us a correlation matrix with dimensions $s \times c$, with s being the number of samples per trace and c being the number of all possible key candidates. Looking for the maximum Pearson correlation coefficient in this matrix gives us a hint for selecting the correct key candidate. In situation depicted in Figure 3.11(a), this approach works just fine.

However, the shape of the correlation curve in time is still more informative, than the magnitude of the correlation coefficient itself. In Figure 3.11(b), one can easily identify the correct key candidate by the naked eye (red curve (2)), while looking for the Pearson correlation coefficient with the highest absolute value fails, as in such a case the blue curve (1) would be selected. Note that with more measurements and power traces available, the



(a) Correlation traces based on a sufficient amount of power traces. The correct key candidate is colored blue.



(b) Correlation traces based on an insufficient amount of power traces. Searching for a (negative) maximum correlation coefficient leads us to the wrong key candidate, which is colored blue (1). The correct key candidate is colored red (2).

Figure 3.11: Correlation traces (a time series of a Pearson correlation coefficient during the encryption), for all 256 key candidates.

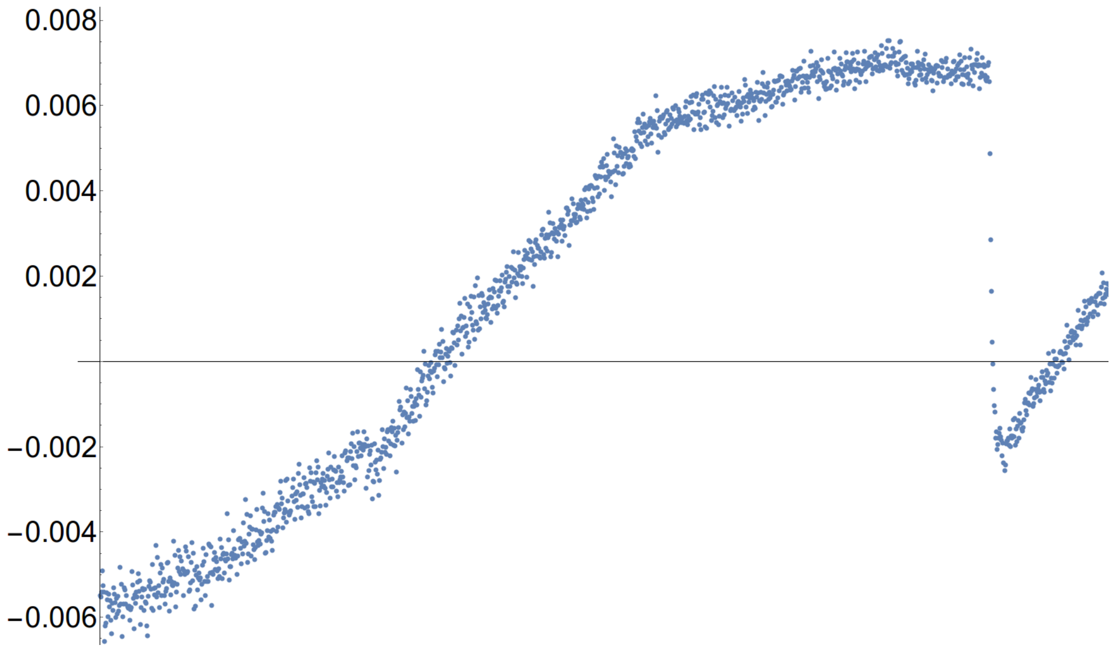


Figure 3.12: Correlation trace obtained while attacking AES on DPAboard [1] (Artix 7 FPGA board with a switching power supply).

spike on the red curve (2) would grow bigger, while correlation at other samples would converge to zero.

The algorithmic evaluation of the correlation coefficients using the naïve maximum likelihood-based method may become even more problematic when there is a significant noise present. Correlation trace for the correct key candidate can be seen in Figure 3.12. This trace was obtained from a board featuring a switching power supply, which represents a significant source of a background noise. Identifying the key candidate by the naked eye is possible, but time demanding, and searching for the maximum/minimum of correlation coefficients does not work well. Our observations led us to an idea of examining the *progression* of the correlation coefficient in time to algorithmically give the attacker a hint about the correct key candidate, rather than examining its *instantaneous magnitude*. According to our research, when correlated working variable causes a change in the power consumption of the device, an edge typically appears in the correlation trace. This problem is very similar to image edge detection problem described in [63, 64].

Since edge detecting operators are very sensitive to noise, appropriate filtering/smoothing of the correlation traces must be done first. In Section 3.3.1.2, we discuss filters that can be used for smoothing the correlation traces. Section 3.3.1.3 discusses edge detectors explored in this work. In Section 3.3.1.4, we describe how to combine filters and edge detecting operators into one operation in order to reduce the computational complexity.

3.3.1.2 Noise Filtering/Smoothing

In image processing, the typical choice is a Gaussian filtering [64, 63]. For our further experimental purposes, we have chosen two filters: the moving average filter and the Gaussian filter.

Moving average filter is defined as follows: Assume that $f(t)$ is a discrete variable, then convolution

$$(f * ma(d))(t) = \frac{1}{d} \sum_{i=t-\lfloor \frac{d}{2} \rfloor}^{t+\lceil \frac{d}{2} \rceil -1} f(i) \tag{3.9}$$

is the result of filtering the variable $f(t)$ using moving average filter with diameter d.

Gaussian filter is defined as follows: Assume that $f(t)$ is a discrete variable, then convolution

$$(f * g(d,\sigma))(t) = \sum_{i=t-\lfloor \frac{d}{2} \rfloor}^{t+\lceil \frac{d}{2} \rceil -1} f(i) \cdot \frac{\exp(-\frac{(i-t)^2}{\sigma^2})}{\mathrm{norm}(d,\sigma)} \tag{3.10}$$

is the result of filtering the variable $f(t)$ using Gaussian filter with diameter d and deviation σ, where

$$\mathrm{norm}(d,\sigma) = \sum_{j=-\lfloor \frac{d}{2} \rfloor}^{\lceil \frac{d}{2} \rceil -1} \exp(-\frac{j^2}{\sigma^2}) \tag{3.11}$$

is the normalization, making sure that the sum of used Gaussian filter equals to 1.

3.3.1.3 Edge Detection

After the noise filtering, the edge detection takes place. There are two approaches to this: a first derivative-based operator, and a second derivative-based (Laplace) operator [63].

When the first derivative approach is used, the filtered correlation traces are processed with the first derivative operator, and then the search for the largest absolute value of the derivative is done. When using the second derivative approach, the algorithm searches for significant zero-crossings of the Laplacian of the correlation trace. Both approaches are compared in Section 3.3.2.

3.3.1.4 Computational Approach

As suggested in [64], both derivative operators and filtering are performed using a discrete convolution. The moving average filter with diameter d can be easily implemented as a convolutional kernel:

$$ma(d) = \frac{1}{d}[\underbrace{1, 1, \ldots, 1}_{d\times}]. \tag{3.12}$$

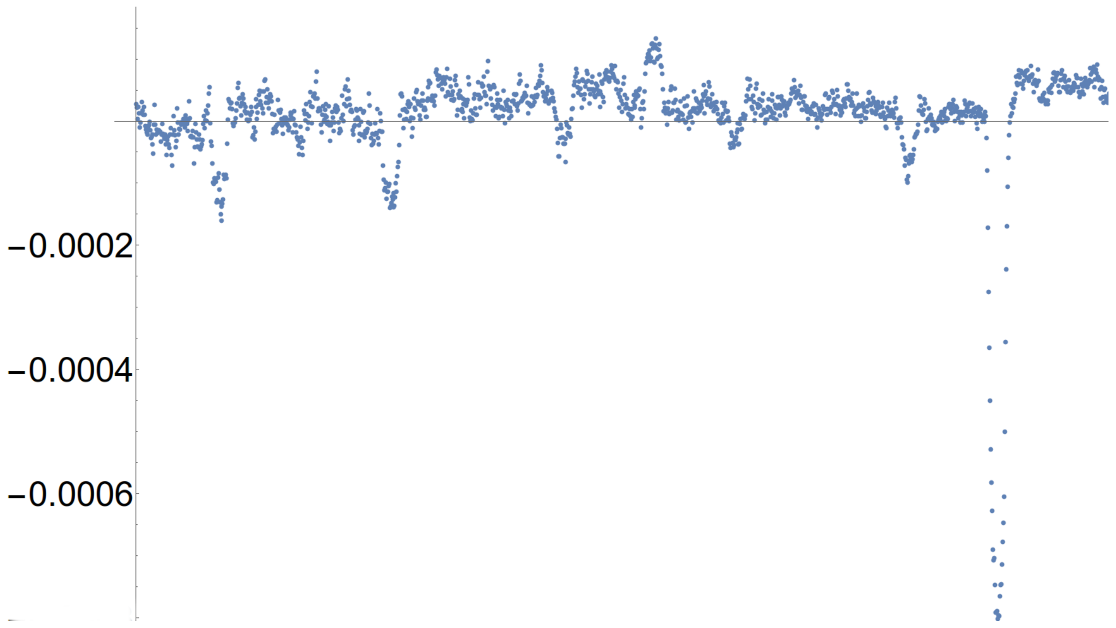


Figure 3.13: Correlation trace from Figure 3.12 processed by first derivative operator with Gaussian filtering.

In the case of the Gaussian filter with deviation σ, an appropriate convolutional kernel of width d can be obtained using a formula

$$G(x,\sigma) \propto \exp(-\frac{x^2}{\sigma^2}), \tag{3.13}$$

and making sure, that the sum of all the terms in the kernel is equal to 1. This can be done by dividing every term of the kernel by the sum of all the kernel terms. For example, Gaussian kernel $g(d=5,\sigma=1)$ looks like

$$g(5,1) = [0.06135, 0.2448, 0.3877, 0.2448, 0.06135]. \tag{3.14}$$

For an approximation of the first derivative, the following convolutional kernel is used:

$$d1 = [-1, 0, 1], \tag{3.15}$$

while for an approximation of the second derivative, the discrete Laplace kernel is used:

$$d2 = [1, -2, 1]. \tag{3.16}$$

Thanks to the associativity of convolution, the smoothing and derivative operator can be precomputed beforehand, resulting in one kernel performing both operations at once. First derivative Gaussian convolution kernel can be obtained using a formula

$$G(x,\sigma)' \propto \frac{x}{\sigma^2} \cdot \exp(-\frac{x^2}{\sigma^2}), \tag{3.17}$$

and Laplacian of Gaussian convolution kernel can be obtained using a formula

$$\Delta G(x,\sigma) \propto \frac{x^2-\sigma^2}{\sigma^4} \cdot \exp(-\frac{x^2}{\sigma^2}). \tag{3.18}$$

The edge detection on a correlation trace can now be done as a convolution, with time complexity $\mathcal{O}(s \times d)$, where s is the number of samples in the correlation trace, and d is the diameter of the filter.

Searching for the correct key candidate in the correlation matrix consists of applying this convolution on each row of the matrix and looking for the largest value (in the case of the first derivative) or zero-crossings (in the case of Laplacian) in the resulting matrix.

3.3.2 Experimental Results

We have executed two classes of experiments:

1. We have evaluated all proposed distinguishers regarding the amount of correctly revealed bytes of the AES-128 cipher key, for various fixed numbers of power traces available. Results of this class of experiments are presented in Section 3.3.2.1.

2. We have evaluated the first derivative + Gaussian distinguisher regarding the filter parameters (filter diameter and deviation). Results of this class of experiments are presented in Section 3.3.2.2.

The platforms we used to evaluate presented methods were following:

- **DPAboard** [1] (open experimental board) with Xilinx Artix 7 FPGA in two revisions: with a switching power supply, and with a low-noise power supply,
- **Sakura-G** board [62] with Xilinx Spartan 6 FPGA,
- **Evariste III** board [61] with Altera Cyclone III FPGA module.

While working with the DPAboard [1] with a switching power supply, we have experienced a lot of unwanted noise in the measured power traces. A correlation trace based on these power traces is shown in Figure 3.12. The correlation trace processed with first derivative operator is shown in Figure 3.13.

3.3.2.1 Evaluation of Proposed Distinguishers

We evaluated the following four distinguishers:

1. Standard CPA, maximizing the Pearson correlation coefficient,

2. Maximizing the first derivative of correlation traces smoothed using moving average,

3. Maximizing the first derivative of correlation traces smoothed using Gaussian filter,

Table 3.14: The number of correctly guessed bytes of the key, Xilinx Artix 7 with a switching power supply.

# of power traces available / **Evaluation method**	100	175	250	500	1k	2.5k	5k	10k	50k	100k
Maximum Pearson correlation coefficient	0	0	0	0	0	0	0	0	2	4
First derivative + moving average (d=25)	0	0	0	0	0	11	16	16	16	16
First derivative + Gaussian (d=25, σ=12.0)	0	0	1	1	7	16	16	16	16	16
Laplacian of Gaussian (d=25, σ=12.0)	0	0	0	0	0	0	1	6	9	9

Table 3.15: The number of correctly guessed bytes of the key, Xilinx Artix 7 with a low-noise power supply.

# of power traces available / **Evaluation method**	100	175	250	500	1k	2.5k	5k	10k	50k	100k
Maximum Pearson correlation coefficient	0	0	2	3	14	16	16	16	16	16
First derivative + moving average (d=25)	0	3	6	13	16	16	16	16	16	16
First derivative + Gaussian (d=25, σ=10.0)	0	1	5	15	16	16	16	16	16	16
Laplacian of Gaussian (d=25, σ=12.0)	0	1	2	5	5	6	7	7	7	7

Table 3.16: The number of correctly guessed bytes of the key, Sakura-G (Xilinx Spartan 6 with a low-noise power supply).

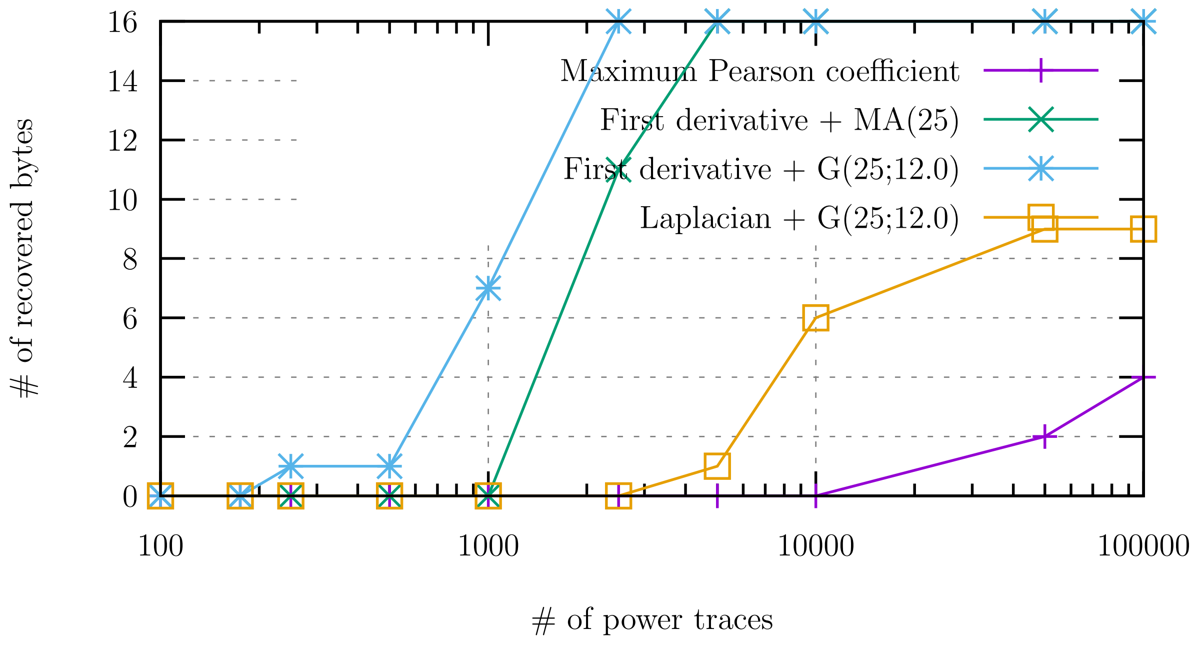
# of power traces available / **Evaluation method**	100	175	250	500	1k	2.5k	5k	10k	50k	100k
Maximum Pearson correlation coefficient	2	2	5	12	16	16	16	16	16	16
First derivative + moving average (d=30)	1	4	6	13	16	16	16	16	16	16
First derivative + Gaussian (d=25, σ=12.0)	2	3	6	12	16	16	16	16	16	16
Laplacian of Gaussian (d=25, σ=12.0)	1	2	5	11	16	16	16	16	16	16

4. Searching for zero-crossings of the Laplacian of correlation traces smoothed using Gaussian filter.

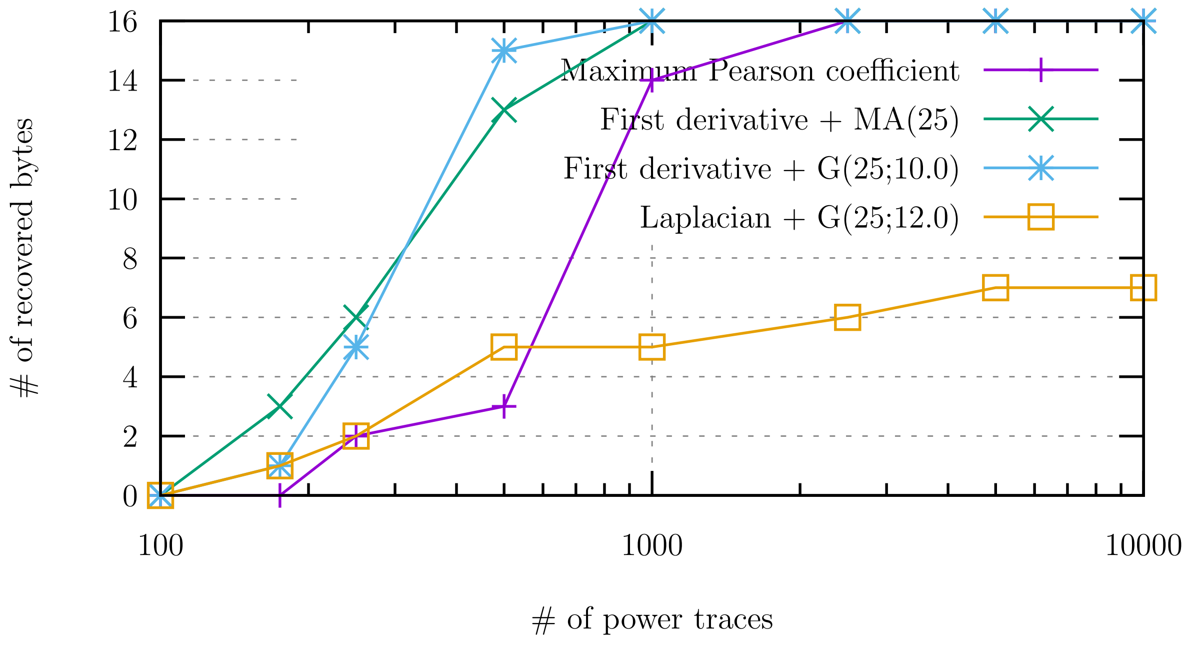
The distinguishers were tested on an AES cipher with 128-bit key, run on three platforms mentioned above. Results are summarized in Tables 3.14, 3.15, 3.16, and 3.17. Each table contains the number of successfully recovered bytes of key for numbers of power traces varying between 100 and 100,000.

Tables 3.14 and 3.15 contain the results based on the correlation traces obtained from an open DPA evaluation board DPAboard [1] with Xilinx Artix 7. We have evaluated these distinguishers using two different revisions of the board: Table 3.14 and Figure 3.14(a) present the results when using the DPAboard with a switching power supply. Table 3.15 and Figure 3.14(b) present the results when using the DPAboard with a low-noise power supply.

As can be seen in Figure 3.14(a), in the case of noisy power traces obtained from the board with a switching power supply, the performance of both first derivative-based distinguishers is much better than the performance of the maximum Pearson correlation



(a) DPAboard (Xilinx Artix 7 FPGA), powered by a switching power supply.



(b) DPAboard (Xilinx Artix 7 FPGA), powered by a low-noise power supply.

Figure 3.14: The number of successfully recovered bytes of AES-128 cipher key using different distinguishers, for various numbers of power traces.

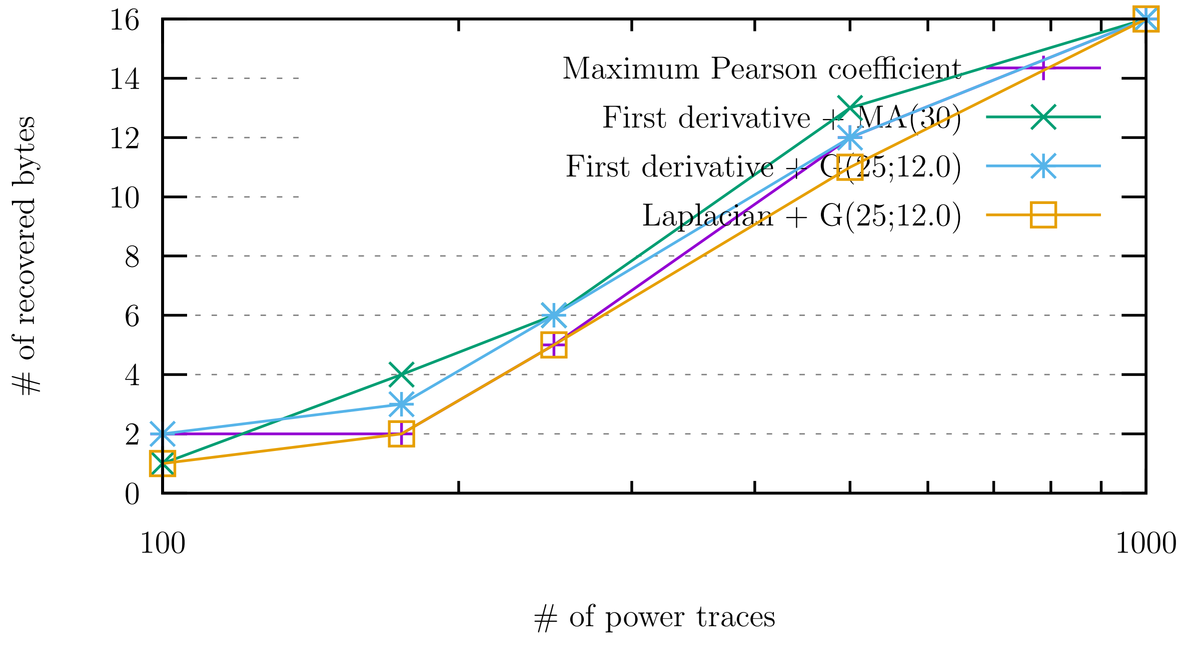


Figure 3.15: The number of successfully recovered bytes of AES-128 cipher key using different distinguishers, for various numbers of power traces, using Sakura-G.

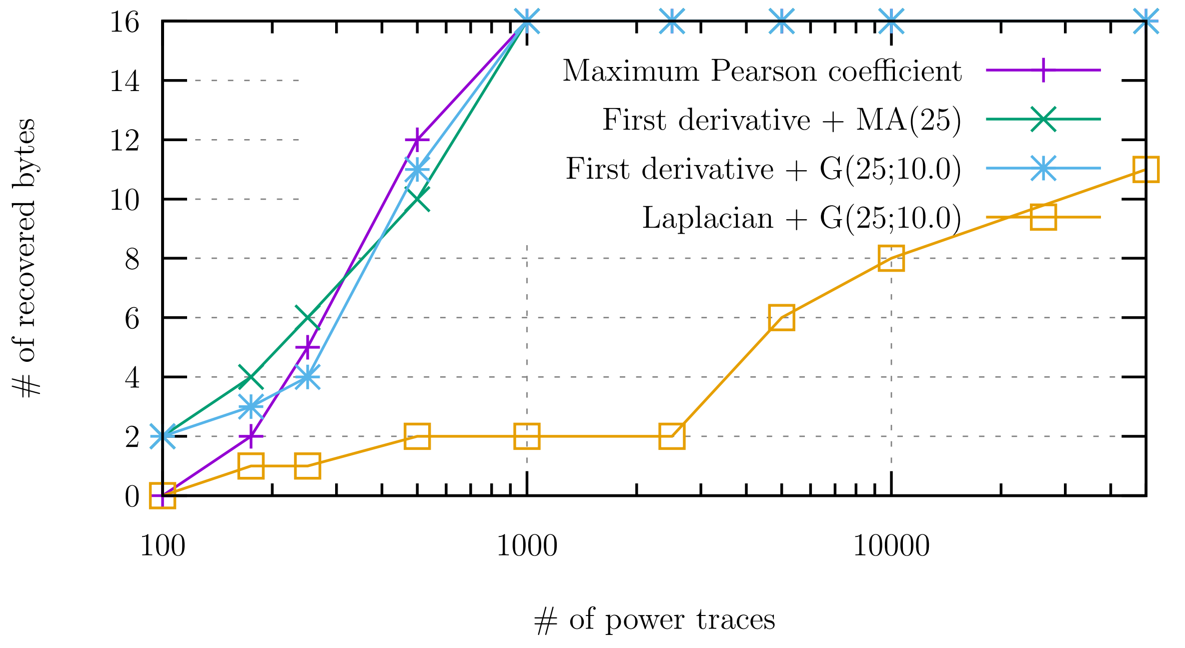


Figure 3.16: The number of successfully recovered bytes of AES-128 cipher key using different distinguishers, for various numbers of power traces, using Evariste III + Altera Cyclone III.

Table 3.17: The number of correctly guessed bytes of the key, Evariste III + Altera Cyclone III with a low-noise power supply.

# of power traces available / **Evaluation method**	100	175	250	500	1k	2.5k	5k	10k	50k	100k
Maximum Pearson correlation coefficient	0	2	5	12	16	16	16	16	16	16
First derivative + moving average (d=25)	2	4	6	10	16	16	16	16	16	16
First derivative + Gaussian (d=25, σ=10.0)	2	3	4	11	16	16	16	16	16	16
Laplacian of Gaussian (d=25, σ=10.0)	0	1	1	2	2	2	6	8	11	11

coefficient method, which actually fails. While in the case of the first derivative approach we needed just 2,500 power traces to successfully reveal all 16 bytes of the key, the maximum Pearson correlation coefficient method did not reveal any byte of the key with the same amount of power traces, and only 4 bytes with all of 100,000 available power traces.

Figure 3.14(b) presents the number of successfully recovered bytes of the key at Xilinx Artix 7 platform with a low-noise power supply. As can be seen, even in a much less noisy environment, our method provides better results. While in the case of the first derivative approach we needed just 1,000 power traces to successfully reveal all 16 bytes of the key, in the case of the maximum Pearson correlation coefficient method we needed 2,500 traces to fully recover the whole key. The Laplacian of Gaussian distinguisher did not prove to be any more effective than the standard CPA. This may be due to the higher noise sensitivity of the second derivative approach.

Table 3.16 and Figure 3.15 present the results obtained while using Sakura-G board [62], equipped with Xilinx Spartan 6 FPGA and a low-noise (linear) power supply. In this case, all methods perform similar, although the first derivative-based distinguishers provide a slightly better results when there is an insufficient amount of power traces available.

Table 3.17 presents the results for the Evariste III [61] with Altera Cyclone III FPGA and a low-noise (linear) power supply. In this case, first derivative distinguishers and standard CPA are comparable again. The first derivative approach may perform a little better for a low amount of power traces, nevertheless, at least 1,000 power traces were necessary for a recovery of the full key. The Laplacian of Gaussian operator did not prove to be any useful in this case either, as can be seen in Figure 3.16.

3.3.2.2 Gaussian Parameters Evaluation

Previous results, summarized in Tables 3.14, 3.15, 3.16, and 3.17, indicate the first derivative distinguisher with Gaussian filtering to be the most promising method. It is particularly successful in noisy environment, as demonstrated in Table 3.14 and Figure 3.14(a).

The performance of the edge detecting operator depends on the selection of its smoothing filter parameters:

- diameter of the filter d, and
- deviation of the Gaussian σ.

Table 3.18: The number of power traces necessary to obtain a full AES encryption key (16 bytes), running on DPAboard (Xilinx Artix 7 with a switching power supply).

Max. Pearson corr. coef.	>100,000								
First derivative + Gaussian (various parameter settings)									
diameter (d) $\rightarrow$ / $\downarrow$ **deviation** (σ)	5	11	17	23	29	35	41	47	53
1.0	13,400	13,400	13,400	13,400	13,400	13,400	13,400	13,400	13,400
2.0	9,400	7,500	7,500	7,500	7,500	7,500	7,500	7,500	7,500
4.0	9,400	4,600	4,500	4,400	4,400	4,400	4,400	4,400	4,400
8.0	9,400	4,000	3,700	2,400	2,400	2,400	2,400	2,400	2,400
12.0	9,500	3,900	3,100	2,400	2,400	2,400	2,400	2,400	2,400
16.0	9,500	3,900	3,100	2,400	2,400	2,400	2,400	2,400	2,400
20.0	9,500	3,900	3,100	2,400	2,400	2,400	2,400	2,400	2,800
24.0	9,500	3,900	3,100	2,400	2,400	2,500	2,400	2,400	2,800
30.0	9,500	3,900	3,100	2,400	2,400	2,500	2,400	2,800	2,800

Table 3.19: The number of power traces necessary to obtain a full AES encryption key (16 bytes), running on DPAboard (Xilinx Artix 7 with a low-noise power supply).

Max. Pearson corr. coef.	1,200								
First derivative + Gaussian (various parameter settings)									
diameter (d) $\rightarrow$ / $\downarrow$ **deviation** (σ)	5	11	17	23	29	35	41	47	53
1.0	>20,000	>20,000	>20,000	>20,000	>20,000	>20,000	>20,000	>20,000	>20,000
2.0	11,300	3,800	3,800	3,800	3,800	3,800	3,800	3,800	3,800
4.0	10,500	600	600	600	600	600	600	600	600
8.0	9,900	700	600	500	500	500	500	500	500
12.0	9,900	900	600	500	500	500	500	500	500
16.0	9,900	900	600	500	500	600	600	600	600
20.0	9,900	900	600	500	500	500	600	600	600
24.0	9,900	900	600	500	500	500	600	600	600
30.0	9,900	900	600	500	500	500	600	600	600

In this section, we evaluate the first derivative distinguisher with Gaussian filtering, using all the evaluation platforms listed in Section 3.3.2.1, and compare the results with the maximum Pearson correlation coefficient method.

Table 3.18 presents the number of power traces necessary for obtaining the whole 16-byte long AES key, using the open FPGA evaluation platform DPAboard [1] (Xilinx Artix 7) with a switching power supply. While for the maximum Pearson distinguisher, we could not retrieve the whole key even with 100,000 power traces available, with optimal first derivative distinguisher only 2,400 power traces are necessary to obtain the whole key.

Table 3.19 presents results obtained using the same evaluation platform, but equipped with the low-noise power supply. In this case, less than a half of the power traces are necessary for a successful attack when using first derivative distinguisher compared to the standard CPA maximum Pearson correlation coefficient (500 vs 1,200).

Table 3.20 presents results obtained when using the Sakura-G [62] evaluation platform (Xilinx Spartan 6), where the performance of both distinguishers is similar.

Table 3.20: The number of power traces necessary to obtain a full AES encryption key (16 bytes), running on Sakura-G (Xilinx Spartan 6 with a low-noise power supply).

Max. Pearson corr. coef.	800								
First derivative + Gaussian (various parameter settings)									
diameter (d) $\rightarrow$ / $\downarrow$ **deviation** (σ)	5	11	17	23	29	35	41	47	53
1.0	>20,000	>20,000	>20,000	>20,000	>20,000	>20,000	>20,000	>20,000	>20,000
2.0	6,200	2,200	2,200	2,200	2,200	2,200	2,200	2,200	2,200
4.0	5,400	1,200	1,200	1,200	1,200	1,200	1,200	1,200	1,200
8.0	5,400	1,200	1,000	900	900	900	900	900	900
12.0	5,400	1,200	900	900	900	800	800	800	800
16.0	5,400	1,200	900	900	900	800	800	700	700
20.0	5,400	1,200	900	900	800	800	700	700	700
24.0	5,400	1,200	900	900	800	800	700	700	700
30.0	5,400	1,200	900	900	800	800	700	700	700

Table 3.21: The number of power traces necessary to obtain a full AES encryption key (16 bytes), running on Evariste III + Altera Cyclone III with a low-noise power supply.

Max. Pearson corr. coef.	700								
First derivative + Gaussian (various parameter settings)									
diameter (d) $\rightarrow$ / $\downarrow$ **deviation** (σ)	5	11	17	23	29	35	41	47	53
1.0	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800	6,800
2.0	2,100	1,400	1,400	1,400	1,400	1,400	1,400	1,400	1,400
4.0	1,900	900	800	800	800	800	800	800	800
8.0	9,900	900	900	700	700	700	700	700	700
12.0	9,900	900	900	900	700	700	700	700	700
16.0	9,900	900	900	900	700	700	700	600	600
20.0	9,900	900	900	900	800	700	700	700	700
24.0	9,900	900	900	900	800	800	900	700	700
30.0	9,900	900	900	900	800	900	900	700	700

Table 3.21 presents the results obtained using the Evariste III platform [61] + Altera Cyclone III module. In this case, the optimal first derivative distinguisher performs slightly better than the maximum Pearson method (600 vs 700).

As can be seen from results summarized in Tables 3.18, 3.19, 3.20, and 3.21, the selection of the filter parameters (d, σ) is crucial for a satisfactory performance of a distinguisher with Gaussian filtering. In our case, the deviation σ should be $\sigma \geq 8.0$ to minimize the number of power traces necessary for successful identification of the correct key. The diameter d also influences the success of the method, although its impact is not that strong as in the case of deviation. Nevertheless, the diameter d should be large enough to fit the Gaussian with the selected deviation, in our case $d \geq 23$.

3.3.3 Conclusion

We have presented a new algorithmic approach to the final step of the CPA attack, which is the selection of the correct key candidate from the correlation traces.

Selecting the key candidate which maximizes the correlation coefficient, according to the maximum likelihood principle, is quite sufficient if the cryptographic device runs in an environment well suitable for power trace measurements. However, this method may fail in the presence of noise or interference in the production environment, caused e.g. by a switching power supply.

We show that our distinguisher based on first derivative edge detection is more successful when evaluating the correlation traces obtained in a noisy environment, such as that made by the switching power supplies. Using our method, approximately 2,400 power traces were necessary for a recovery of the whole key, while the maximization of the Pearson correlation coefficient failed to do so even with 100,000 power traces.

While working with low-noise/linear power supplies and having a sufficient amount of power traces available, both approaches work equally good. When the amount of power traces is insufficient, our first derivative method may provide slightly better results. The Laplacian of Gaussian based distinguisher did not prove to be much useful.

The extra time complexity of the proposed methods is insignificant compared to the rest of the CPA attack. The resulting reduction of the power traces necessary to reveal the cipher key is even more beneficial considering that the measuring of the power traces is usually by far the most time-consuming part of the attack. Although the time complexity of distinguishers with Gaussian filtering increases with increasing diameter d, the (very slight) increase of time is more than compensated by reducing the time necessary for both acquisition of power traces (i.e. measurement by an oscilloscope) and calculation of correlation traces.

Chapter **4**

Dependability vs. Security

Sometimes, a digital device needs to be designed with regard to both dependability and security. For example, when a cryptographic module is designed as a part of a mission-critical system. In this chapter, we discuss the problematics of digital design being dependable and secure at the same time. We focus on basic fault-tolerant schemes based on multiple kinds of redundancy on the dependability side of the problem, while we focus on the security issue of protection against side-channel analysis. In Section 4.1, the influence of basic fault-tolerant architectures on resistance against side-channel analysis (specifically correlation power analysis) is evaluated. We propose novel architectures based on simple redundancy techniques and Boolean masking providing both fault-tolerance and resistance to side-channel analysis in Section 4.2.

The experimental evaluation of results presented in Section 4.1 was done in cooperation with a bachelor student Jan Říha. The main idea of the work discussed in the section was presented at an international conference MECO 2016 [A.1] and PhD workshops PESW 2016 [A.9] and PAD 2016 [A.10]. The paper presented at MECO 2016 was invited for an extension to Microprocessors and Microsystems journal, where an evaluation of the influence of space redundancy on SCA resistance was published [A.3], which we also published at a workshop FCTRU 2016 [A.11]. The rest of this work was presented at an international conference DSD 2017 [A.4], security workshops CryptArchi 2017 [A.14] and Trudevice 2018 [A.16], and a PhD workshop PAD 2017 [A.15].

The work introduced in Section 4.2 was presented at a workshop CryptArchi 2019 [A.18] and at a PhD workshop PAD 2019 [A.19].

4.1 Influence of Dependable Architectures on CPA-Resistance

The most straightforward way of designing a device fault-tolerant are simple fault-tolerant architectures [47, 48, 65] described in Section 2.2. While using fault-tolerant design methods in cryptographic circuits, one important question arises: how do these methods influence the attack-resistance of the design?

In this section, we focus on the common fault-tolerant architectures, specifically space redundancy, time redundancy, and information redundancy. We compare these architectures with a simple (fault unprotected) implementation of the cryptographic algorithm. As a reference design, we have chosen AES-128 without any SCA countermeasures, so we can easily compare the influence of the fault-tolerant architectures. We experimentally evaluate the resistance against side-channel analysis using correlation power analysis.

Similar work is presented by Regazzoni et al. [49, 50] and Dofe et al. [51], as described in Section 2.3, but only fault-tolerance as a countermeasure against fault attacks is regarded in these publications. **Unlike the above mentioned studies, we focus on fault-tolerance in context of dependability.** While the referred works focus on protection of S-boxes using error detection/correction codes like parity or Hamming code, we focus on protection of the whole design (or major parts of it) using different kinds of redundancy.

Detailed description of the fault-tolerant architectures we used for the evaluation and our expectations about their influence on CPA resistance are presented in Section 4.1.1. In Section 4.1.2, we describe the experiment setup. Results of our experimental evaluation are presented and discussed in Section 4.1.3 and our findings are concluded in Section 4.1.4.

4.1.1 Architectures

To evaluate and compare the influence of fault-tolerant design techniques on resistance against CPA, we used a reference AES design (Section 4.1.1.1) and five AES designs employing various fault-tolerant design techniques:

- Information redundancy (Section 4.1.1.2)
- Space redundancy at round level (Section 4.1.1.3)
- Space redundancy at algorithm level (Section 4.1.1.4)
- Time redundancy at round level (Section 4.1.1.5)
- Time redundancy at algorithm level (Section 4.1.1.6)

4.1.1.1 Reference AES

Round based implementation of AES-128 described in Section 3.2.1.2 and depicted in Figure 3.4 is used as a reference design.

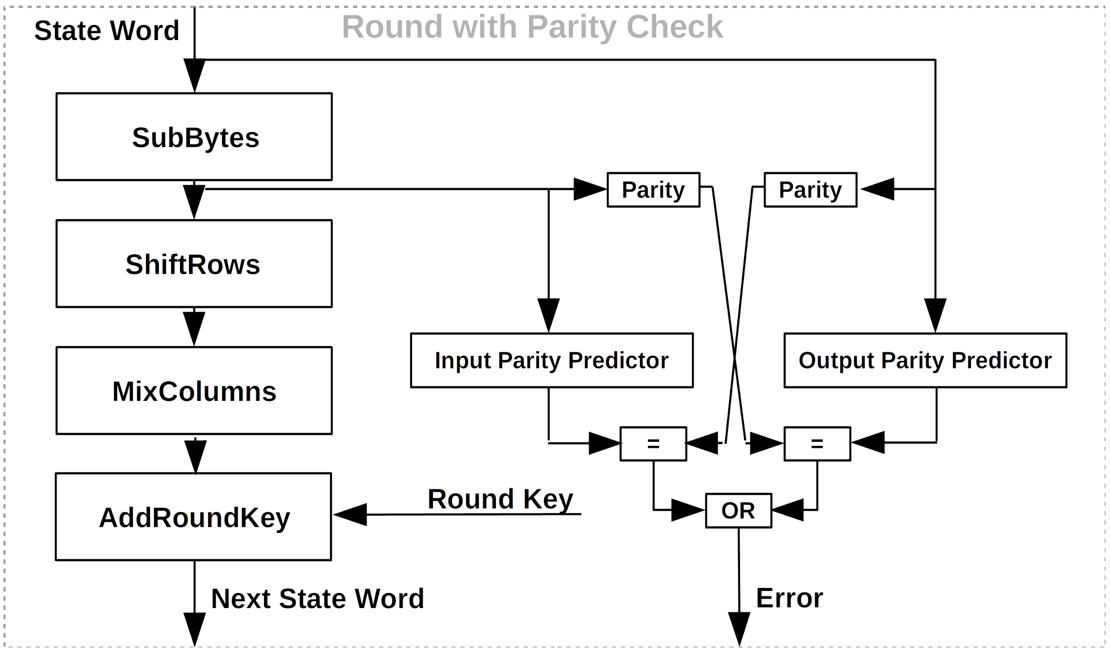


Figure 4.1: Diagram of a round with parity checked SubBytes

4.1.1.2 SubBytes Parity Check

This fault-tolerant architecture is designed to detect a fault a by parity check in the modules implementing SubBytes function (S-boxes) [66], which covers a large part of the area of AES implementation. Since SubBytes, being a nonlinear function, does not preserve parity, we needed to implement a parity predictor. We use two parity predictors implemented as look-up tables. The first one is predicting the parity of the output using the input and the second one is predicting the parity of the input using the output. Both predicted values are compared with the real values and a fault is indicated when they differ. There are both input and output parity checkers for each of 16 S-boxes. Diagram of a round with parity checked SubBytes function is shown in Figure 4.1.

As this architecture introduces quite a low area overhead, we expect the power consumption to be similar to the power consumption of the reference design and we also expect this design not to affect the CPA resistance. On the other hand, this architecture does not provide any fault correction. It provides just the fault detection; moreover, it is limited to S-boxes only.

4.1.1.3 Space Redundancy — Round Level

Space redundancy is represented by the most common fault-tolerant architecture — triple-modular redundancy [47]. It is based on three copies of a module and a majority voter. When one of the modules is faulty, the majority voter ensures that the result is still correct. A diagram of this architecture is shown in Figure 4.2.

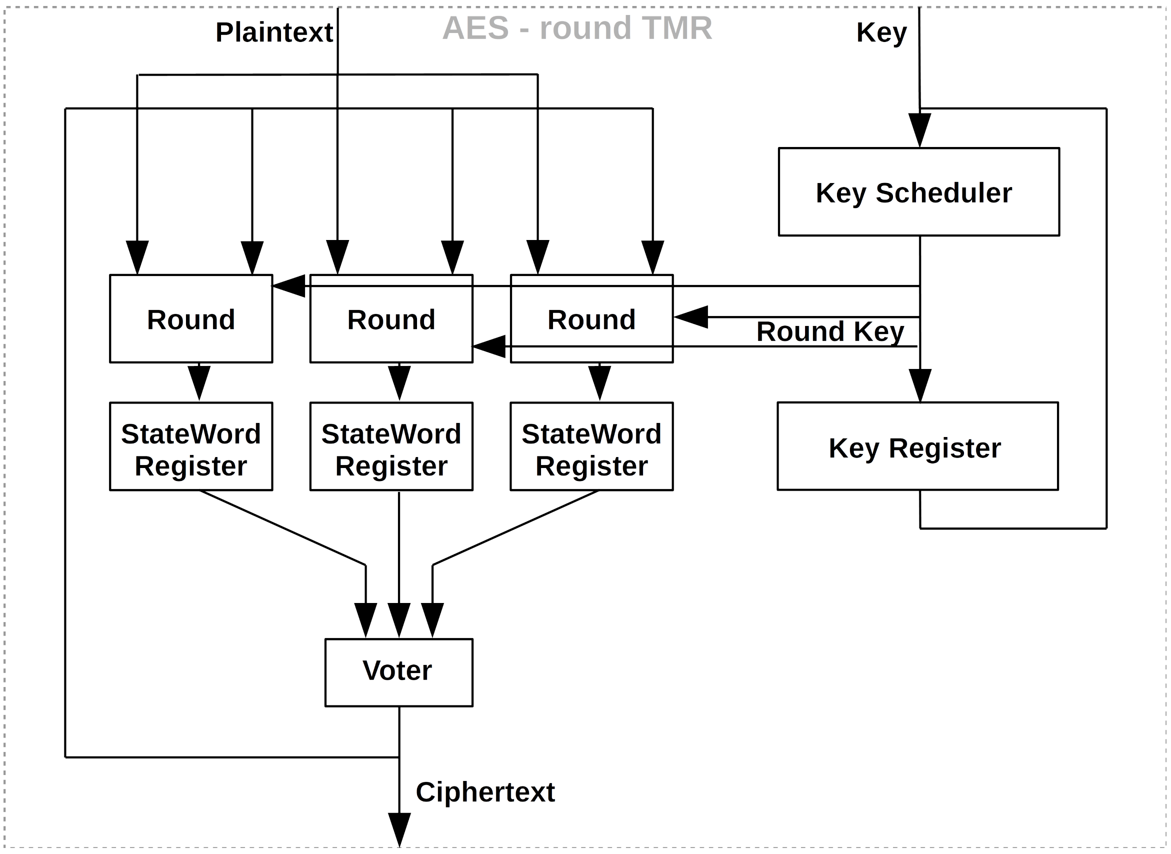


Figure 4.2: Diagram of AES secured by space redundancy at round level

In this case, we use TMR at round level, so the data path implementing a single round of the design is triplicated.

This architecture causes high area overhead (the round logic is triplicated), which leads to increased power consumption. This increase should triplicate the information available in the power consumption, thus it should increase the signal to noise ratio, which would make the device more vulnerable to CPA. On the other hand, the increased power consumption may cause an increase in the noise (caused e.g. by power supply) and also the power consumption of the majority voter can introduce some additional noise.

4.1.1.4 Space Redundancy — Algorithm Level

This architecture is another one based on TMR. In this case, the whole AES module is triplicated and outputs are evaluated using three majority voters, therefore this architecture masks a fault in any part of the whole design. A diagram of this architecture is shown in Figure 4.3.

In this case, we also expect increased power consumption. In comparison with the round level, the overall power consumption increase should be higher, but the additional information in this increase should be the same. Therefore we expect this architecture

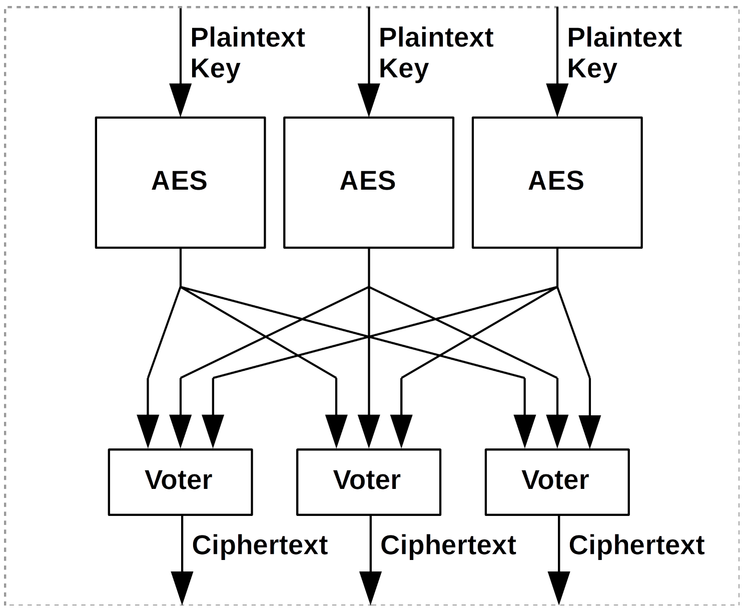


Figure 4.3: Diagram of AES secured by space redundancy at algorithm level

to have the same or lower influence on CPA resistance than the architecture described in Section 4.1.1.3.

4.1.1.5 Time Redundancy — Round Level

Time redundancy is based on repeating the same calculations on the same data multiple times. In this architecture each round is run three times, each result is stored in a register and then the results are compared by a majority voter similarly to the one in Section 4.1.1.3. Using this approach, the design is tolerant to transient faults only. A diagram of this architecture is shown in Figure 4.4.

This architecture introduces minimal area overhead (only a voter and registers for a result of each iteration are added). Compared to the reference circuit described in Section 4.1.1.1, the information available in the power consumption is exposed three times in the circuit with time redundancy. We do not expect this scheme to affect the CPA resistance significantly.

4.1.1.6 Time Redundancy — Algorithm Level

This architecture is very similar to the one in Section 4.1.1.5, instead of repeating each round separately, the whole encryption is repeated. The results are also stored in registers and then compared by a majority voter. A diagram of this architecture is shown in Figure 4.5.

We expect similarly small influence on CPA resistance as in the case of round level time redundancy presented in Section 4.1.1.5.

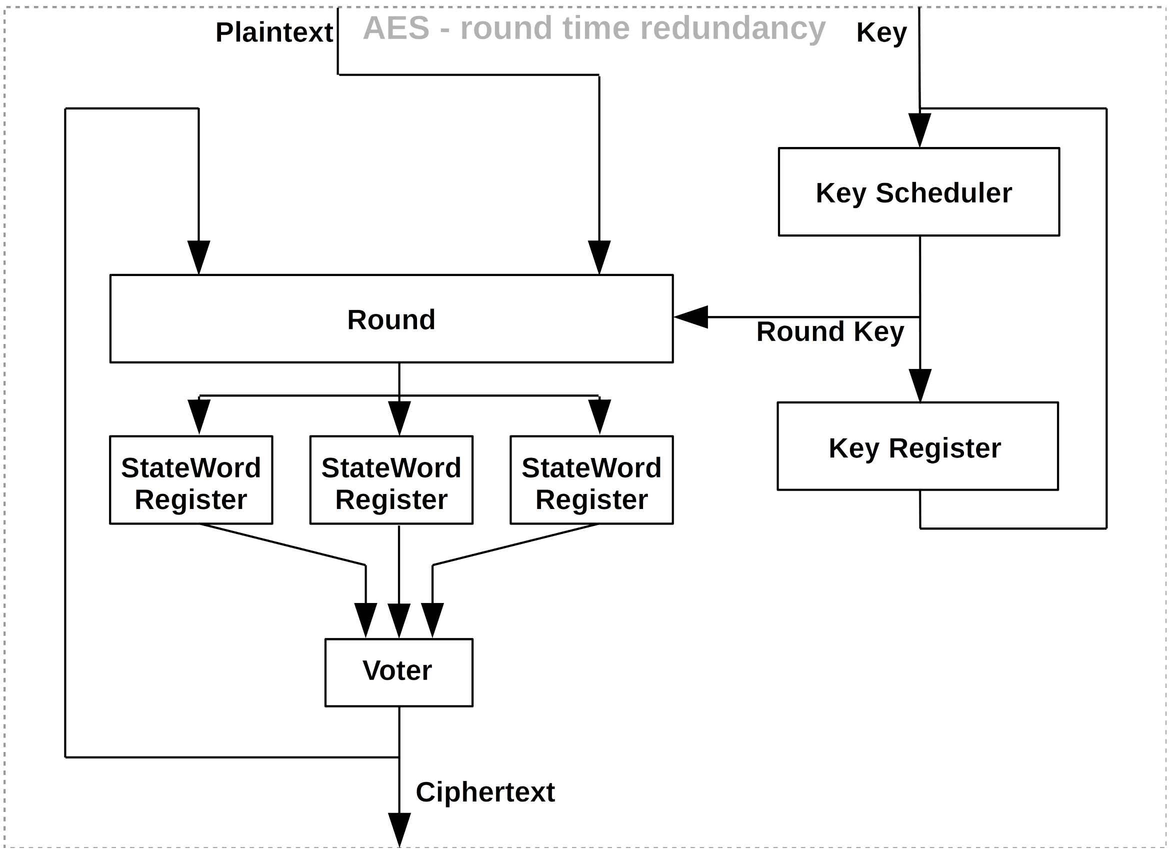


Figure 4.4: Diagram of AES secured by time redundancy at round level

4.1.2 Measurement

The hardware platform used for the measurement is described and the evaluation process is discussed in this section.

4.1.2.1 FPGA Platform

Evariste III board with Altera Cyclone III FPGA module [61] has been chosen as an evaluation platform. All AES variants were synthesized by Altera Quartus II 13.1. The frequency of the FPGA was set to 1 MHz.

4.1.2.2 Power Analysis

The power consumption was measured by Agilent DSO 7104A oscilloscope. For each encryption, a power trace of 1000 samples was measured.

We analyzed the data using the DPA with correlation distinguisher (CPA). Concerning the power model, we use Hamming distance between the value at the beginning of the 10th round (state 9) and the end of the 10th round (ciphertext) (state 10), as it is depicted

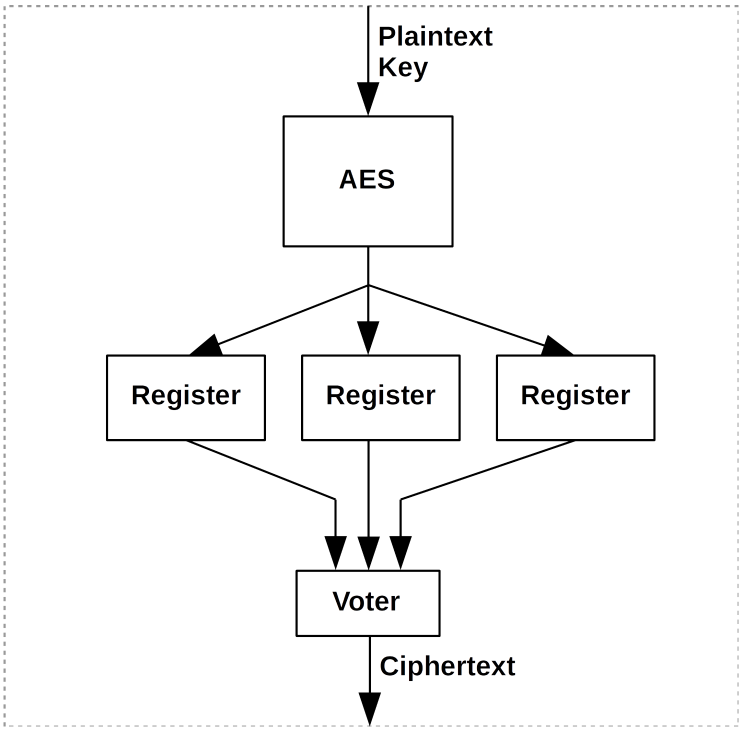


Figure 4.5: Diagram of AES secured by time redundancy at algorithm level

in Figure 3.6. The actual attack is described in Section 2.1.4. The key candidate with the highest absolute value in a correlation trace was chosen as the correct key candidate.

4.1.2.3 Evaluation

The minimal amount of power traces sufficient to reveal the correct key candidate for all bytes of the key is used as a quantification of the CPA resistance.

For each variant of AES, we collected 50 different sets of power traces, thus we obtained 50 quantifications for each variant.

4.1.3 Results

The evaluated architectures are compared by medians of the minimal numbers of power traces sufficient to reveal the correct key and the interquartile ranges because the standard deviation was up to 20% of the mean and quantiles are less susceptible to long-tailed distributions and outliers than means [67].

The comparison of medians and interquartile ranges is shown in Table 4.1 with the following meaning of the abbreviations:

- **AES:** Standard AES module (reference design, Section 4.1.1.1)
- **AES-SPC:** SubBytes parity check (Section 4.1.1.2)

Table 4.1: Comparison of AES variants based on median and interquartile range of minimal power traces necessary to obtain the correct key

Architecture	Median	Interquartile range	Difference from AES
AES	850	175	0%
AES-SPC	950	250	+12%
AES-SR-R	900	275	+6%
AES-SR-A	812	150	-4%
AES-TR-R	1025	250	+21%
AES-TR-A	1037	275	+22%

- **AES-SR-R:** Space redundancy at round level (Section 4.1.1.3)
- **AES-SR-A:** Space redundancy at algorithm level (Section 4.1.1.4)
- **AES-TR-R:** Time redundancy at round level (Section 4.1.1.5)
- **AES-TR-A:** Time redundancy at algorithm level (Section 4.1.1.6)

As we can see in Table 4.1 and a box plot in Figure 4.6, the differences between studied AES architectures lie within the interquartile ranges, therefore these differences are not statistically significant. Almost all architectures show slightly positive (if any) influence on CPA resistance. In the following subsections, we discuss obtained results and we confront them with our assumptions stated in Section 4.2.

4.1.3.1 SubBytes Parity Check (AES-SPC)

This architecture made the design a little less vulnerable to CPA. This is probably caused by the parity checkers, which act as a noise generator. Nevertheless, the influence is low.

4.1.3.2 Space Redundancy (AES-SR-R, AES-SR-A)

In this case, the difference from the reference AES is very low; nevertheless, the algorithm level space redundancy is the only architecture reaching less resistance to CPA. In the case of the round level space redundancy, the resistance is a bit increased. This result is contrary to our assumptions. On the other hand, the results are both very similar to the original AES implementation, so it is hard to make any conclusion.

4.1.3.3 Time Redundancy (AES-TR-R, AES-TR-A)

Despite our assumptions, the results of the time redundancy shown the highest influence on the CPA resistance of the measured architectures. The medians are on the edge of the interquartile range of the reference AES implementation. However, this result may have another explanation: as long as we obtained 1000 samples per a power trace in all studied architectures and the encryption time of time redundancy architectures is three times

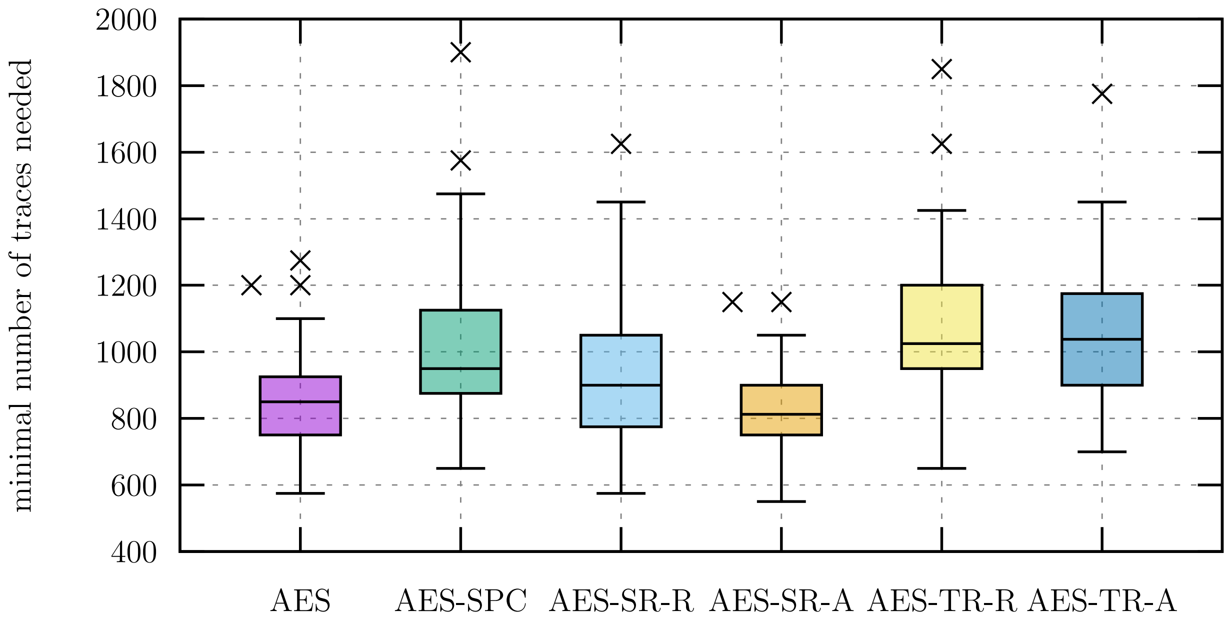


Figure 4.6: Box plot of the minimal number of power traces necessary to reveal the correct key for all measured AES architectures

higher, the sample resolution is three times lower in comparison with the rest of the architectures. This may be the reason for the increased resistance to CPA.

4.1.4 Conclusion

The influence of multiple fault-tolerant architectures using various kinds of redundancy on resistance against side-channel analysis is experimentally evaluated and analyzed in this work.

As we show in Section 4.1.3, the measured fault-tolerant architectures had minimal influence on resistance against CPA. Most of the architectures even increased the resistance.

The parity check of SubBytes introduces noise and it slightly increases the resistance. The space redundancy seems to balance the noise and information increase. The time redundancy shows the highest influence but it is probably based on the longer duration of encryption and it may be evaded by e.g. attacking a third of the power trace.

These results indicate that the studied fault-tolerant architectures can be used for designing a fault-tolerant cryptographic device without making it more vulnerable to side-channel analysis. The difference in SCA resistance is statistically insignificant and above that, it is usually positive.

These results are in contradiction with the results of similar studies mentioned in Section 2.3. There are multiple explanations to this difference: the fact that the other studies focused mostly on securing the S-boxes only, different platforms used or others. Disregarding the reasons, in all studies, including ours, the influence proved to be quite small and the differences in the results only strengthen the insignificance of the influence of evaluated architectures on side-channel analysis resistance.

4.2 Area-Efficient Dependable Architectures Exploiting Masking Scheme Randomness

One of the possible ways to protect a device against side-channel analysis is masking. An overview of masking countermeasures is presented in Section 2.1.6.1, where multiple schemes provably secure against SCA of arbitrary order, e.g. threshold implementation or domain-oriented masking, are presented. Masking randomizes all the intermediate values of a cryptographic algorithm and therefore it eliminates the side-channel leakage. Randomization of inputs (and therefore outputs) for each encryption is a common property of these schemes.

An easy way of increasing dependability is hardware (space) modular redundancy, which is described in Section 2.2.1. These schemes are based on simple replication of logic modules. Common examples are DMR, TMR or NMR. These architectures detect and/or correct both permanent or transient faults of common types (stuck-at-0, stuck-at-1, bit-flipping). The biggest advantage is the simplicity making these architectures very universal and popular. On the other hand, these architectures introduce very high overhead of area, power, etc.

This work aims to propose architectures based on existing masking schemes and modular redundancy architectures. In comparison with existing methods, these architectures decrease the overhead while they keep both the attack-resistance of masking schemes and the simplicity and dependability properties of the existing modular redundancy architectures. Specifically, we exploit the fact that a fault in the masked circuit causes different (unmasked) outputs when different random masks are used. This property permits us to build less redundant design utilizing the redundancy introduced by the masking scheme itself.

The principles of the proposed architectures using various amounts of redundant modules are described in detail in Section 4.2.1. Case study of TMR-equivalent architecture using TI of PRESENT cipher [68] is presented in Section 4.2.2, where also the reduction of area overhead and leakage evaluation of the proposed architecture is discussed. Results of our work are concluded in Section 4.2.3.

4.2.1 Methodology

To demonstrate our approach, we describe fault-tolerant architectures providing similar dependability properties as TMR and NMR, but using a lower number of redundant modules. The architectures rely on an assumption that a fault in the encryption leads to different faulty output for the same but differently masked inputs. Note that in this work, we deal only with fault-tolerance of the data paths. The control unit should be handled independently (as it also usually is in the case of the traditional modular redundancy architectures).

We assume round-based implementation of a symmetric cipher as depicted in Figure 4.7. Such a design can be divided into three parts: input logic (usually just input signals), encryption logic (round logic and round register), and output logic (output signals and e.g.

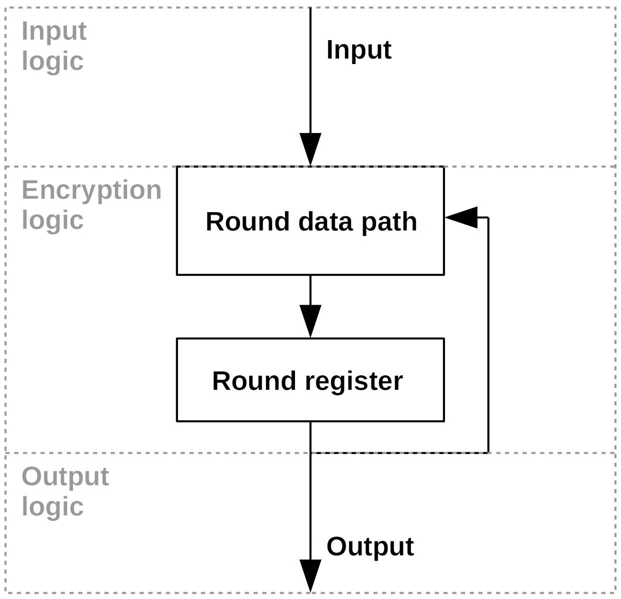


Figure 4.7: Diagram of round based cipher implementation

output multiplexers). Any single fault in encryption logic leads to different outputs for different random masks with high probability, as such a fault is propagated through the whole encryption process. On the other hand, a fault in the input or output logic can lead to the same (unshared) outputs for different masks.

4.2.1.1 TMR-Equivalent

TMR architecture serves to correct a fault in one of three modules. We propose an architecture with similar properties using two modules only. A diagram of the proposed architecture can be seen in Figure 4.8. The two modules run the encryption in parallel with the same masks. The outputs of the encryption are compared and, in the case they differ, the encryption is repeated with new masks (same for both modules). For both modules, (unmasked) outputs of the consecutive encryptions are compared. There are three possible results:

1. The consecutive outputs are equal for both modules. This result may be caused by two reasons:

   a) A transient fault occurred during the previous encryption.

   b) A permanent fault occurred, but it is not detectable with the current mask.

   Regardless of the reason, we assume that both of the current outputs are correct.

2. The consecutive outputs are equal for one of the modules. We assume that the current output of the module with equal outputs is correct.

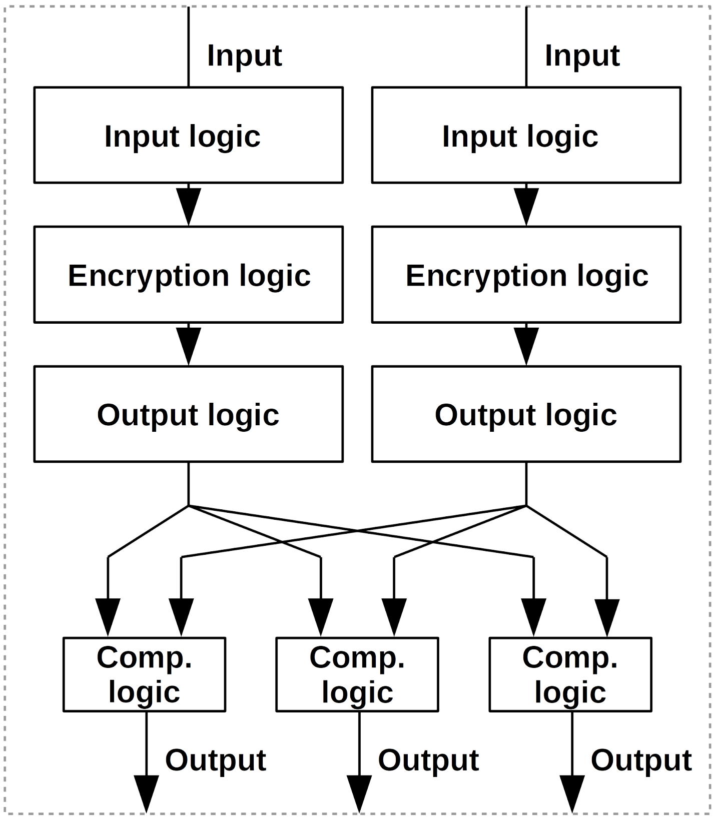


Figure 4.8: Diagram of proposed TMR-equivalent architecture

3. The consecutive outputs are not equal for either of the modules. None of the outputs is correct, fault in both modules is detected.

As mentioned above, the proposed architecture enables masking of a single fault, similarly to TMR. In comparison with TMR, we spared one of three modules. The encryption takes the same time as TMR unless a fault is introduced. Additionally, we can detect (different) faults in both modules at the same time with no additional logic.

To protect the design also against a fault in the comparison logic, this needs to be triplicated similarly as in the case of TMR. A diagram of an example of comparison logic is in Figure 4.9. This logic is more complex than the majority voters in TMR. On the other hand, the proposed comparison logic compares unmasked outputs allowing to use a shorter comparator. The comparator is also multiplexed and it is used for comparison of outputs of both modules as well as for comparison of outputs of two subsequent encryptions.

4.2.1.2 NMR-Equivalent

Since NMR is a generalization of TMR, we can analogically generalize our TMR-equivalent architecture. Using the architecture proposed in the previous section, we only need one faultless module at a time, since if not all outputs are the same, the correctly working

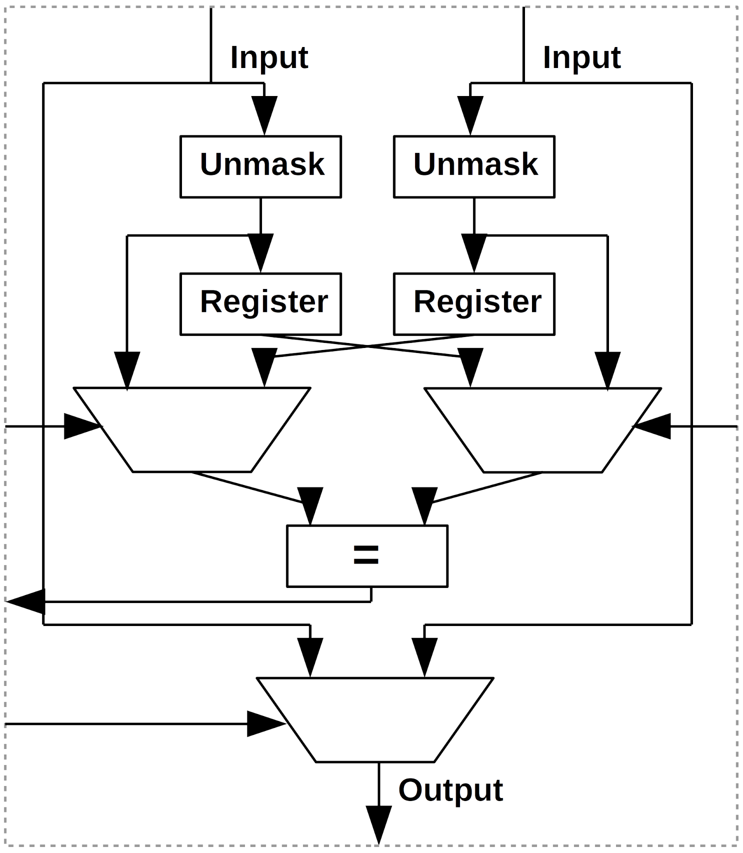


Figure 4.9: Diagram of proposed comparison logic for TMR-equivalent architecture

module(s) can be detected by repeating the encryption. Therefore, we only need $n + 1$ modules to tolerate n faults (instead of $2 \times n + 1$ in the case of NMR). Nevertheless, we still need $2 \times n + 1$ very complex comparator circuits.

4.2.2 Case Study

In this section, we evaluate the TMR-equivalent architecture proposed in Section 4.2.1.1 using three-share threshold implementation of PRESENT cipher [68] as is described in [69]. The cipher is implemented in Spartan-6 FPGA on Sakura-G evaluation board [62]. Three implementations are compared regarding the area of the design: single module, TMR, and our TMR-equivalent architecture. All three designs were synthesized using Xilinx XST (Xilinx ISE 14.7).

4.2.2.1 Results

A comparison of slice utilization for each of the implementations can be seen in Table 4.2. Our architecture is about 20% smaller than standard TMR. This difference can seem smaller than expected considering the fact that two modules instead of three were used.

Table 4.2: Comparison of area demands of evaluated implementations

Design	**Slice utilization**	**Overhead**
Single module	2199	0%
TMR	7180	227%
TMR-equivalent	5764	162%

This small difference is caused by the fact that PRESENT is a lightweight cipher, and therefore the encryption modules are relatively small, while the comparison logic occupies significant area. In the case of more robust ciphers like AES [9], the advantage of our approach would be more significant.

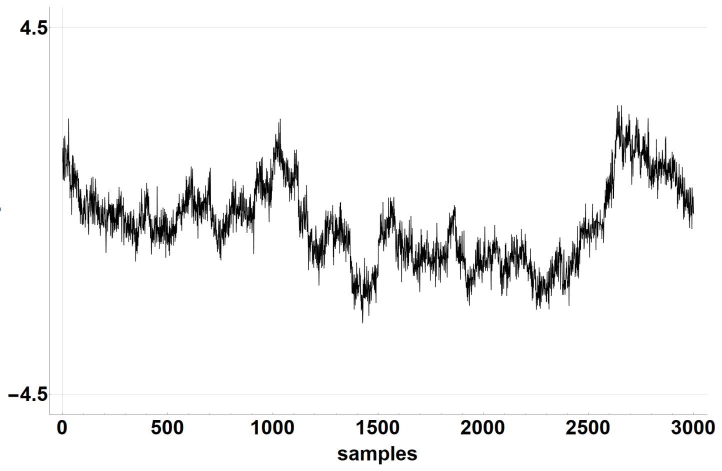
4.2.2.2 Test Vector Leakage Assessment

Test vector leakage assessment was performed for each implementation. The leakage was evaluated for 1,000,000 traces using non-specific, fixed vs. random, first-order Welch's t-test, as it is described in Section 2.1.7.1. PicoScope 6404D was used for the measurements. Plots of t-values can be seen in Figure 4.10. As we can see, there is no leakage in the case of a single module (4.10(a)) and TMR (4.10(b)). In the case of our TMR-equivalent (4.10(c)), there is leakage in the end of the encryption. This leakage is caused by the comparison of unmasked outputs in the comparison logic. Even though this leakage is not a major security threat, as it only leaks information about the ciphertext, which is considered a public information, we propose an alternative comparison logic.

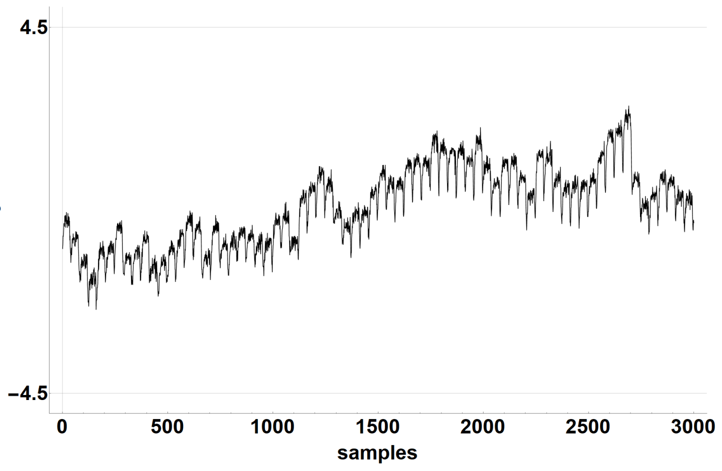
Alternative Comparison Logic The problem of the side-channel leakage of the comparison logic can be solved by comparing masked outputs of the modules. This approach only demands a larger comparator. Nevertheless, when a fault is detected and outputs of two consecutive encryptions need to be compared, the outputs must be unmasked (as different input masks are used). In this case, completely random input data (plaintext and cipher key) are used for encryption and the encryption is repeated with different masks until one of the modules is detected to be faulty (or until some threshold number of repetitions is reached when the fault is considered to be transient). No useful data get unmasked by a comparison logic designed in this way.

4.2.3 Conclusion

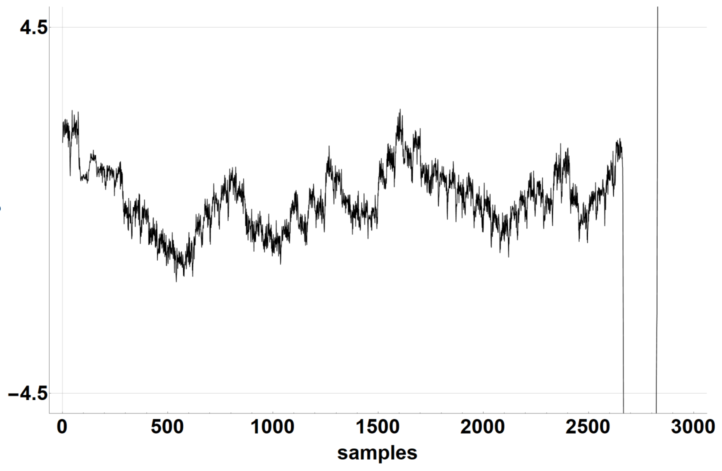
This work deals with an issue of having a device both secure and dependable. Glitch-resistant masking schemes offer provable protection against side-channel analysis and additionally against fault sensitivity analysis. Nevertheless, they introduce quite high area overhead. When such a masked design needs to also fulfill dependability requirements, the additional overhead introduced by e.g. simple modular redundancy schemes like TMR can be unbearable.



(a) Single module



(b) TMR



(c) TMR-equivalent

Figure 4.10: Plots of t-values

We proposed architectures based on masking schemes exploiting their involved randomness. These architectures keep the simplicity of modular redundancy, while they decrease the number of redundant modules and therefore they significantly decrease the area overhead. In the case of TI of PRESENT cipher, the area overhead of standard TMR architecture is 227%, while the area overhead of our TMR-equivalent architecture is only 162%. The saving would be even more significant for some more complex encryption scheme (e.g. AES). Implemented architecture also passed the Welch's t-test with an exception of output unmasking in comparison logic. Nevertheless, an alternative comparison logic dealing with this issue was also presented.

Considering these facts, we can conclude that we proposed a novel approach of secure and dependable design, exploiting the redundancy introduced by a masking scheme, with similar qualities and lower overhead in comparison with the original methods.

Chapter 5

Conclusions

Since side-channel analysis poses a serious threat to cryptographic devices, it became an extensively researched area. Two topics related to side-channel analysis are the main aim of this dissertation thesis: efficient attack implementations and fault-tolerance of SCA countermeasures.

An introduction to the topic and goals of this thesis are presented in Chapter 1.

In Chapter 2, the theoretical background and state of the art of the topic are presented. An overview of side-channel analysis attacks and countermeasures is introduced, focusing on attacks evolved from differential power analysis and countermeasures based on masking. An overview of common dependability architectures is also presented. Since, to the best of our knowledge, there is no research about dependability issues in the context of side-channel analysis resistance and countermeasures, an overview of a similar topic — the influence of fault attack countermeasures on resistance against SCA — is presented including a discussion about differences between this topic and ours.

Methods to speed up correlation power analysis are introduced in Chapter 3. A comparison of different approaches for correlation computations is presented together with a proposal of efficient implementations of first-order and higher-order analysis. A method of decreasing the complexity of the correlation computations by aggregation of power traces is also presented. The last method proposes an algorithmic key candidate selection based on edge detection.

An interplay between dependability and security is a topic of Chapter 4. Evaluation of the influence of multiple dependable architectures on resistance to SCA is presented and area-efficient architectures for dependable and SCA-resistant digital design are proposed.

In Chapter 5, the thesis is concluded. Summary and contributions are presented and some future work is proposed.

5.1 Summary

Three approaches for time-efficient enhancements of correlation power analysis were proposed. All three approaches proved to be functional and useful by experimental evaluations.

How different dependability architectures based on different kinds of redundancy affect resistance to side-channel analysis is evaluated. The results show, that the influence is not statistically significant. In accordance with this result, architectures for dependable and secure design are proposed. These architectures are based on hardware modular redundancy and masking schemes. The case study shows that the proposed architectures keep the simplicity of the modular redundancy architectures and SCA resistance of masking schemes, while it significantly saves resources.

All results were presented and discussed in the scientific community. In the first place, it was published in proceedings of 5 international conferences and 2 journals, while some of these works reached a considerable amount of citations.

5.2 Contributions of the Dissertation Thesis

1. Evaluation of multiple approaches to first-order and higher-order correlation computations and a proposal of time-efficient and numerically stable implementations: This work is based on existing algorithmic solutions, nevertheless a comprehensive performance evaluation was presented and advantages and disadvantages of different approaches were thoroughly discussed.

2. An efficient method for speeding up the power analysis based on an aggregation of power traces: The method was experimentally evaluated on multiple platforms and it drastically (tens of times) speeds up the computational part of correlation power analysis.

3. Improved correlation distinguisher based on edge detection: An improvement of key candidate selection from correlation traces based on edge detection using various functions and various parameters is evaluated. The first derivative showed better results than the original maximization of the Pearson correlation coefficient, especially in a noisy environment.

4. Comprehensive evaluation of the influence of dependable architectures on resistance against side-channel analysis: An experimental evaluation of the influence of various dependable architectures based on various kinds of redundancy on resistance against side-channel analysis provides possibly surprising results, showing that the influence is minimal for all evaluated architectures. Therefore, fault-tolerant side-channel countermeasures can be designed using these architectures.

5. Area-efficient architectures for dependability and resistance to side-channel analysis: Architectures for both side-channel-resistant and fault-tolerant digital design are proposed. These architectures keep the simplicity of hardware modular redundancy architectures and SCA-resistance of masking schemes, while they significantly save resource in comparison with a simple combination of the original approaches.

5.3 Future Work

The author of the dissertation thesis suggests to explore the following:

- It would be interesting to evaluate the trace-aggregation technique using another SCA distinguisher.
- The evaluation of the influence of dependable architectures on SCA resistance could be extended by other platforms (e.g. ASIC) and/or more dependable architectures. Also, a comparison using TVLA could be interesting.
- Continuation of this work on correlation distinguisher and a proposal for some new improvements or whole new SCA distinguisher is one of the aims for our future research.

Bibliography

[1] Bartík, M.; Buček, J. A low-cost multi-purpose experimental FPGA board for cryptography applications. In *Advances in Information, Electronic and Electrical Engineering (AIEEE), 2016 IEEE 4th Workshop on*, IEEE, 2016, pp. 1–4.

[2] Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Annual International Cryptology Conference*, Springer, 1999, pp. 388–397.

[3] Quisquater, J.-J.; Samyde, D. Electromagnetic analysis (ema): Measures and countermeasures for smart cards. *Smart Card Programming and Security*, 2001: pp. 200–210.

[4] Bernstein, D. J. Cache-timing attacks on AES. 2005.

[5] Hutter, M.; Schmidt, J.-M. The temperature side channel and heating fault attacks. In *International Conference on Smart Card Research and Advanced Applications*, Springer, 2013, pp. 219–235.

[6] Peeters, E.; Standaert, F.-X.; Quisquater, J.-J. Power and electromagnetic analysis: Improved model, consequences and comparisons. *Integration, the VLSI journal*, volume 40, no. 1, 2007: pp. 52–60.

[7] Rivest, R. L.; Shamir, A.; Adleman, L. On Digital Signatures and Public-Key Cryptosystems. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1977.

[8] Mangard, S. A simple power-analysis (SPA) attack on implementations of the AES key expansion. In *International Conference on Information Security and Cryptology*, Springer, 2002, pp. 343–358.

[9] Pub, N. F. 197: Advanced encryption standard (AES). *Federal Information Processing Standards Publication*, volume 197, no. 441, 2001: p. 0311.

[10] Messerges, T. S.; Dabbish, E. A.; Sloan, R. H. Examining smart-card security under the threat of power analysis attacks. *IEEE transactions on computers*, volume 51, no. 5, 2002: pp. 541–552.

[11] Brier, E.; Clavier, C.; Olivier, F. Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2004, pp. 16–29.

[12] Gierlichs, B.; Batina, L.; Tuyls, P.; et al. Mutual information analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2008, pp. 426–442.

[13] Mangard, S.; Oswald, E.; Standaert, F.-X. One for all–all for one: unifying standard differential power analysis attacks. *IET Information Security*, volume 5, no. 2, 2011: pp. 100–110.

[14] de Chérisey, E.; Guilley, S.; Rioul, O. Confused yet Successful:. In *Information Security and Cryptology*, edited by F. Guo; X. Huang; M. Yung, Cham: Springer International Publishing, 2019, ISBN 978-3-030-14234-6, pp. 533–553.

[15] Croxton, F. E.; Cowden, D. J. *Applied general statistics*. Prentice-Hall, 1940.

[16] Schneider, T.; Moradi, A. Leakage assessment methodology. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2015, pp. 495–513.

[17] Schneider, T.; Moradi, A.; Güneysu, T. Robust and one-pass parallel computation of correlation-based attacks at arbitrary order. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, Springer, 2016, pp. 199–217.

[18] Chari, S.; Jutla, C. S.; Rao, J. R.; et al. Towards sound approaches to counteract power-analysis attacks. In *Annual International Cryptology Conference*, Springer, 1999, pp. 398–412.

[19] Golić, J. D.; Tymen, C. Multiplicative masking and power analysis of AES. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2002, pp. 198–212.

[20] Fumaroli, G.; Martinelli, A.; Prouff, E.; et al. Affine masking against higher-order side channel analysis. In *International Workshop on Selected Areas in Cryptography*, Springer, 2010, pp. 262–280.

[21] Nikova, S.; Rechberger, C.; Rijmen, V. Threshold implementations against side-channel attacks and glitches. In *International Conference on Information and Communications Security*, Springer, 2006, pp. 529–545.

[22] Nikova, S.; Rijmen, V.; Schläffer, M. Secure hardware implementation of non-linear functions in the presence of glitches. In *International Conference on Information Security and Cryptology*, Springer, 2008, pp. 218–234.

[23] Bilgin, B.; Gierlichs, B.; Nikova, S.; et al. Higher-order threshold implementations. In *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2014, pp. 326–343.

[24] Groß, H.; Mangard, S.; Korak, T. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In *TIS@ CCS*, 2016, p. 3.

[25] Reparaz, O.; Bilgin, B.; Nikova, S.; et al. Consolidating masking schemes. In *Annual Cryptology Conference*, Springer, 2015, pp. 764–783.

[26] Li, Y.; Sakiyama, K.; Gomisawa, S.; et al. Fault sensitivity analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2010, pp. 320–334.

[27] Arribas, V.; De Cnudde, T.; Šijačić, D. Glitch-Resistant Masking Schemes as Countermeasure Against Fault Sensitivity Analysis. In *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, IEEE, 2018, pp. 27–34.

[28] Ors, S. B.; Preneel, B. Power Analysis of an FPGA Implementation of Rijndael: Is Pipelining a DPA Countermeasure. In *in Cryptographic Hardware and Embedded Systems*, Citeseer, 2004.

[29] Coron, J.-S.; Kizhvatov, I. An efficient method for random delay generation in embedded software. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2009, pp. 156–170.

[30] Jeřábek, S.; Schmidt, J. Analyzing and Optimizing the Dummy Rounds Scheme. In *2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, IEEE, 2019, pp. 1–4.

[31] Tiri, K.; Verbauwhede, I. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Proceedings of the conference on Design, automation and test in Europe-Volume 1*, IEEE Computer Society, 2004, p. 10246.

[32] Tiri, K.; Hwang, D.; Hodjat, A.; et al. Prototype IC with WDDL and differential routing–DPA resistance assessment. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2005, pp. 354–365.

[33] Soares, R.; Calazans, N.; Lomné, V.; et al. Evaluating the robustness of secure triple track logic through prototyping. In *SBCCI'08: Symposium on Integrated Circuits and Systems Design*, ACM, 2008, pp. 193–198.

[34] Danger, J.-L.; Guilley, S.; Bhasin, S.; et al. Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors. In *Signals, Circuits and Systems (SCS), 2009 3rd International Conference on*, IEEE, 2009, pp. 1–8.

[35] Bajard, J.-C.; Imbert, L.; Liardet, P.-Y.; et al. Leak resistant arithmetic. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2004, pp. 62–75.

[36] Mesquita, D.; Badrignans, B.; Torres, L.; et al. A cryptographic coarse grain reconfigurable architecture robust against dpa. In *2007 IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2007, pp. 1–8.

[37] Mentens, N.; Gierlichs, B.; Verbauwhede, I. Power and fault analysis resistance in hardware through dynamic reconfiguration. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2008, pp. 346–362.

[38] Güneysu, T.; Moradi, A. Generic side-channel countermeasures for reconfigurable devices. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2011, pp. 33–48.

[39] Sasdrich, P.; Mischke, O.; Moradi, A.; et al. Side-channel protection by randomizing look-up tables on reconfigurable hardware. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, Springer, 2015, pp. 95–107.

[40] Chatzikokolakis, K.; Chothia, T.; Guha, A. Statistical measurement of information leakage. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2010, pp. 390–404.

[41] Chothia, T.; Guha, A. A statistical test for information leaks using continuous mutual information. In *2011 IEEE 24th Computer Security Foundations Symposium*, IEEE, 2011, pp. 177–190.

[42] Gilbert Goodwill, B. J.; Jaffe, J.; Rohatgi, P.; et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, 2011, pp. 115–136.

[43] Balasch, J.; Gierlichs, B.; Grosso, V.; et al. On the cost of lazy engineering for masked software implementations. In *International Conference on Smart Card Research and Advanced Applications*, Springer, 2014, pp. 64–81.

[44] Moradi, A.; Richter, B.; Schneider, T.; et al. Leakage detection with the x2-test. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018: pp. 209–237.

[45] Wegener, F.; Moos, T.; Moradi, A. DL-LA: Deep Learning Leakage Assessment: A modern roadmap for SCA evaluations. Cryptology ePrint Archive, Report 2019/505, 2019, `https://eprint.iacr.org/2019/505`.

[46] Welch, B. L. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, volume 34, no. 1/2, 1947: pp. 28–35.

[47] Pradhan, D. *Fault-tolerant computer system design.* Upper Saddle River, N.J: Prentice Hall PTR, 1996, ISBN 0-13-057887-8.

[48] Koren, I. *Fault-tolerant systems.* Amsterdam Boston: Elsevier/Morgan Kaufmann, 2007, ISBN 9780120885251.

[49] Regazzoni, F.; Eisenbarth, T.; Breveglieri, L.; et al. Can knowledge regarding the presence of countermeasures against fault attacks simplify power attacks on cryptographic devices? In *Defect and Fault Tolerance of VLSI Systems, 2008. DFTVS'08. IEEE International Symposium on*, IEEE, 2008, pp. 202–210.

[50] Regazzoni, F.; Breveglieri, L.; Ienne, P.; et al. Interaction between fault attack countermeasures and the resistance against power analysis attacks. In *Fault Analysis in Cryptography*, Springer, 2012, pp. 257–272.

[51] Dofe, J.; Pahlevanzadeh, H.; Yu, Q. A Comprehensive FPGA-Based Assessment on Fault-Resistant AES against Correlation Power Analysis Attack. *Journal of Electronic Testing*, volume 32, no. 5, 2016: pp. 611–624.

[52] Messerges, T. S. Using second-order power analysis to attack DPA resistant software. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2000, pp. 238–251.

[53] Waddle, J.; Wagner, D. Towards efficient second-order power analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2004, pp. 1–15.

[54] Loğoğlu, K. B.; Ateş, T. K. Speeding-up Pearson correlation coefficient calculation on graphical processing units. In *Signal Processing and Communications Applications Conference (SIU), 2010 IEEE 18th*, IEEE, 2010, pp. 840–843.

[55] Gamaarachchi, H.; Ragel, R.; Jayasinghe, D. Accelerating correlation power analysis using graphics processing units (gpus). In *Information and Automation for Sustainability (ICIAfS), 2014 7th International Conference on*, IEEE, 2014, pp. 1–6.

[56] Higham, N. J. *Accuracy and stability of numerical algorithms.* Siam, 2002.

[57] Bottinelli, P.; Bos, J. W. Computational aspects of correlation power analysis. *Journal of Cryptographic Engineering*, 2015: pp. 1–15.

[58] Chan, T. F.; Golub, G. H.; LeVeque, R. J. Updating formulae and a pairwise algorithm for computing sample variances. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, Springer, 1982, pp. 30–41.

[59] Pébay, P. Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments. *Sandia Report SAND2008-6212, Sandia National Laboratories*, volume 94, 2008.

[60] Socha, P. SICAK: SIde-Channel Analysis toolKit. Available from: `https://petrsocha.github.io/sicak/`

[61] Fischer, V.; Bernard, F.; Haddad, P. An open-source multi-FPGA modular system for fair benchmarking of true random number generators. In *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, IEEE, 2013, pp. 1–4.

[62] Guntur, H.; Ishii, J.; Satoh, A. Side-channel attack user reference architecture board SAKURA-G. In *Consumer Electronics (GCCE), 2014 IEEE 3rd Global Conference on*, IEEE, 2014, pp. 271–274.

[63] Marr, D.; Hildreth, E. Theory of edge detection. *Proceedings of the Royal Society of London B: Biological Sciences*, volume 207, no. 1167, 1980: pp. 187–217.

[64] Canny, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, , no. 6, 1986: pp. 679–698.

[65] Anghel, L.; Alexandrescu, D.; Nicolaidis, M. Evaluation of a soft error tolerance technique based on time and/or space redundancy. In *Integrated Circuits and Systems Design, 2000. Proceedings. 13th Symposium on*, IEEE, 2000, pp. 237–242.

[66] Di Natale, G.; Flottes, M.-L.; Rouzeyre, B. A novel parity bit scheme for SBox in AES circuits. In *Design and Diagnostics of Electronic Circuits and Systems, 2007. DDECS'07. IEEE*, IEEE, 2007, pp. 1–5.

[67] Serfling, R. J. *Approximation theorems of mathematical statistics*, volume 162. John Wiley & Sons, 2009.

[68] Bogdanov, A.; Knudsen, L. R.; Leander, G.; et al. PRESENT: An ultra-lightweight block cipher. In *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 2007, pp. 450–466.

[69] Poschmann, A.; Moradi, A.; Khoo, K.; et al. Side-channel resistant crypto for less than 2,300 GE. *Journal of Cryptology*, volume 24, no. 2, 2011: pp. 322–345.

Reviewed Publications of the Author Relevant to the Thesis

[A.1] Miškovský, V.; Kubátová, H.; Novotný, M. Influence of fault-tolerant design methods on differential power analysis resistance of AES cipher: Methodics and Challenges. In *5th Mediterranean Conference on Embedded Computing (MECO 2016)*, Bar, Montenegro, 2016.

The paper has been cited in:

- Socha, P.; Brejník, J.; Bartik, M. Attacking AES implementations using correlation power analysis on ZYBO Zynq-7000 SoC board. In *7th Mediterranean Conference on Embedded Computing MECO 2018*, Budva, Montenegro, 2018.

[A.2] Socha, P.; Miškovský, V.; Kubátová, H.; Novotný, M. Optimization of Pearson correlation coefficient calculation for DPA and comparison of different approaches. In *20th International Symposium on Design and Diagnostics of Electronic Circuit & Systems (DDECS)*, Dresden, Germany, 2017.

The paper has been cited in:

- Liu, Y.; Zhao, Y.; He, J.; Liu, A.; Xin, R. SCCA: Side-channel correlation analysis for detecting hardware Trojan. In *11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 2017.
- Nascimento, J.F.O. Estudo de preços de energia no mercado spot e futuros no MIBEL. *PhD thesis*, Instituto Superior de Engenharia do Porto, 2017.
- Wei, Y. Form Factors of Modeling Design Language with Improved Entropy Weight Based on Kaisen Engineering. *Revista de la Facultad de Ingeniería*, 32.10, 2017.

[A.3] Miškovský, V.; Kubátová, H.; Novotný, M. Influence of Passive Hardware Redundancy on Differential Power Analysis Resistance of AES Cipher implemented in FPGA. *Microprocessors and Microsystems – Elsevier B.V 2017*, Volume 51, pp. 220-226, ISSN 0141-9331, 2017.

[A.4] Říha, J.; Miškovský, V.; Kubátová, H.; Novotný, M. Influence of Fault-Tolerance Techniques on Power-Analysis Resistance of Cryptographic Design. In *20th Euromicro Conference on Digital System Design*, Vienna, Austria, 2017.

[A.5] Miškovský, V.; Kubátová, H.; Novotný, M. Speeding up differential power analysis using integrated power traces. In *7th Mediterranean Conference on Embedded Computing MECO 2018*, Budva, Montenegro, 2018.

[A.6] Socha, P.; Miškovský, V.; Kubátová, H.; Novotný, M. Correlation Power Analysis Distinguisher Based on the Correlation Trace Derivative. In *21st Euromicro Conference on Digital System Design*, Prague, Czech Republic, 2018.

The paper has been cited in:

- Wang, M.; Huang, K.; Wang, Y.; Wu, Z.; Du, Z. A novel side-channel analysis for physical-domain security in cyber-physical systems. *International Journal of Distributed Sensor Networks*, 2019.

[A.7] Socha, P.; Miškovský, V.; Novotný, M. First-Order and Higher-Order Power Analysis: Computational Approaches and Aspects. In *8th Mediterranean Conference on Embedded Computing MECO 2019*, Budva, Montenegro, 2019.

[A.8] Socha, P.; Miškovský, V.; Kubátová, H.; Novotný, M. Efficient Algorithmic Evaluation of Correlation Power Analysis: Key Distinguisher Based on the Correlation Trace Derivative. *Microprocessors and Microsystems* – Elsevier B.V, 2019. *(ACCEPTED)*

Remaining Publications of the Author Relevant to the Thesis

[A.9] Miškovský, V.; Kubátová, H.; Novotný, M. Influence of Fault-tolerant Design Methods on Resistance against Differential Power Analysis. In *The 4th Prague Embedded Systems Workshop*, Roztoky u Prahy, Czech Republic, 2016.

[A.10] Miškovský, V.; Kubátová, H.; Novotný, M. Číslicový návrh spojující odolnost proti útokům a odolnost proti poruchám. In *Počítačové architektury a diagnostika*, Bořetice, Česká republika, 2016.

[A.11] Miškovský, V.; Kubátová, H.; Novotný, M. Influence of Fault-tolerant Design Methods on Resistance against Differential Power Analysis in FPGA. In *Conference on Trustworthy Manufacturing and Utilization of Secure Devices*, Barcelona, Spain, 2016.

[A.12] Miškovský, V. Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices. *Ph.D. Minimum Thesis*, Faculty of Information Technology, Prague, Czech Republic, 2017.

[A.13] Miškovský, V. Fault tolerance and resistance against side channel attacks in FPGA. *Technical Report TR-FIT-17-03*, Faculty of Information Technology, Prague, Czech Republic, 2017.

[A.14] Miškovský, V. Influence of Fault-Tolerant Design Techniques on Resistance against Differential Power Analysis. In *CryptArchi*, Smolenice, Slovakia, 2017.

[A.15] Miškovský, V.; Kubátová, H.; Novotný, M. Číslicový návrh spojující odolnost proti poruchám a odolnost proti útokům. In *Počítačové architektury a diagnostika*, Smolenice, Slovensko, 2017.

[A.16] Miškovský, V.; Kubátová, H.; Novotný, M. Influence of Fault-Tolerance Techniques on Power-Analysis Resistance of AES implemented in FPGA. In *TRUDEVICE 2018*, Dresden, Germany, 2018.

[A.17] Socha, P.; Miškovský, V.; Novotný, M. SICAK: An open-source SIde-Channel Analysis toolKit. In *TRUDEVICE 2019*, Baden-Baden, Germany, 2019.

[A.18] Miškovský, V. Area-efficient fault-tolerant architectures exploiting masking scheme randomness. In *CryptArchi*, Průhonice, Czech Republic, 2019.

[A.19] Miškovský, V.; Kubátová, H.; Novotný, M. Útoky postranními kanály: efektivní implementace a ochrany odolné proti poruchám. In *Počítačové architektury a diagnostika*, Doksy, Česká republika, 2019.

Remaining Publications of the Author

[A.20] Jeřábek, S.; Schmidt, J.; Novotný, M.; Miškovský, V. Dummy Rounds as a DPA countermeasure in hardware. In *21st Euromicro Conference on Digital System Design*, Prague, Czech Republic, 2018.

Projects of the Author

[A.21] SGS15/119/OHK3/1T/18, *Attack-Resistant and Fault-Tolerant Architectures Based on Reconfigurable Devices*, Czech Technical University in Prague, member of a research team, 2015

[A.22] SGS16/121/OHK3/1T/18, *Dependable architectures suitable for FPGAs*, Czech Technical University in Prague, member of a research team, 2016

[A.23] SGS17/017/OHK3/1T/18, *Dependable and attack-resistant architectures for programmable devices*, Czech Technical University in Prague, member of a research team, 2017–2019

[A.24] GA16-05179S, *Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features*, Czech Grant Agency, member of a research team, 2016–2018

[A.25] CELSA/17/033, *DRASTIC: Dynamically Reconfigurable Architectures for Side-channel analysis protection of Cryptographic implementations*, Central Europe Leuven Strategic Alliance, member of a research team, 2017–2019