FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Differential Power Analysis Countermeasures in Programmable Hardware

by

Stanislav Jeřábek

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.

Doctoral study programme: Informatics
Department of Digital Design

Prague, August 2023

Supervisor:
doc. Ing. Jan Schmidt, Ph.D.
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 Electrical Engineering
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract and contributions

This dissertation thesis deals with the threat of side-channel attacks to all implementations of cryptographic algorithms, which is an extensively researched area. The main aim of this thesis is non-provable secure countermeasures, which, despite that fact, make side-channel attacks much harder, sometimes practically infeasible.

We propose the usage of known countermeasures for more complex ciphers and our new countermeasure scheme for hardware implementations.

We have secured AES and Serpent by countermeasures proposed for PRESENT cipher. Our implementations have no leakage being evaluated by first-order Welch's t-test and have resisted second-order DPA/CPA attacks.

The new countermeasure proposed by us is called Dummy Runds, and it is straightforwardly applicable to any round-based cryptographic algorithm. Dummy Rounds are a hardware scheme for the implementation of shuffling when shuffling is a common countermeasure for software implementations.

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

1. Second-order DPA and CPA attacks resistance implementations of AES and Serpent: This work describes an implementation of AES and Serpent secured with previously proposed countermeasures implemented for PRESENT. This work evaluates the implementations as leakage-free using the non-specific univariate first-order Welch's t-test. In later work, these implementations resisted second-order DPA and CPA attacks.

2. Hardware Dummy Rounds countermeasure scheme: We present our proposed and continually evolved Dummy Rounds countermeasure. Its results are competitive with some other solo-used previously proposed countermeasures. The advantages of our Dummy Rounds are a straightforward approach to implementing the scheme and an implicit trade-off between security and area overhead.

3. Dummy Rounds linear overhead regardless of an implemented algorithm: Although Dummy Rounds has high area overhead in comparison with other countermeasures

considering light-weight ciphers (PRESENT is the case study), it has good area overhead results in comparison with the same countermeasures considering more complex ciphers.

4. Bringing attention to the control part of the circuit, as opposed to the usual focus on side channels of the datapath, optimization of the control algorithm and safe controller design.

Keywords:
Security, Side-channel analysis, Side-channel countermeasures, Dummy rounds, Field programmable gate arrays.

Abstrakt

Tato práce se zabývá útoky postranními kanály, které představují hrozbu pro všechny kryptografické implementace a jsou rozsáhle zkoumanou oblastí. Hlavní zkoumanou oblastí práce jsou taková protiopatření, jejichž bezpečnost nelze prokázat, ale navzdory tomu dělají útoky výrazně těžšími až prakticky neuskutečnitelnými.

Představujeme použití již známých protiopatření pro zabezpečení složitějších šifer, a také navrhujeme naše nové protiopatření pro hardwarové implementace.

Zabezpečili jsme šifry AES a Serpent protiopatřeními použitými pro šifru PRESENT. Z našich implementací podle testu Welchovým t-testem prvního řádu neuniká žádná informace a odolaly také útokům Rozdílovou a Korelační odběrovou analýzou druhého řádu.

Naše nové protiopatření se jmenuje Dummy Rounds a je snadno implementovatelné pro libovolnou šifru založenou na rundovém schématu. Dummy Rounds je schéma pro použití v hardwaru, které spadá do kategorie protiopatření Shuffling, jinak běžné pro softwarové implementace.

Klíčová slova:

Bezpečnost, Analýza postranních kanálů, Protiopatření proti útokům postranními kanály, Dummy rounds, Programovatelná hradlová pole.

Acknowledgements

First of all, I would like to express my gratitude to my dissertation thesis supervisor, doc. Ing. Jan Schmidt, Ph.D. and my co-supervisor, Dr.-Ing. Martin Novotný. They have been a bottomless well of knowledge and experience during my research and helped me with many problems.

I would like to thank all my colleagues at the department for keeping a great atmosphere. My special thanks go to Ing. Vojtěch Miškovský, Ph.D., for checking this thesis as an unexpected friendly gesture.

I would like to express special thanks to the department management and my supervisors for providing most of the funding for my research. My research has also 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.

Finally, my greatest thanks go to my family members, especially my parents, for their infinite patience and care.

Dedication

To my wife Anna.
You were my biggest motivation to finish this thesis, and you are the love of my life.

Contents

List of Figures

List of Tables

Abbreviations

Attacks

SCA	Side-Channel Analysis
SPA	Simple Power Analysis
DPA	Differential Power Analysis...
CPA	Correlation Power Analysis ...

Countermeasures

WDDL	Wave Dynamic Differential Logic
STTL	Secure Triple Track Logic

Ciphers

DES	Data Encryption Standard
AES	Advanced Encryption Standard
RSA	Rivest–Shamir–Adleman

Hardware Design

VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuit
FPGA	Field-Programmable Gate Array
LUT	Look-Up Table
CFGLUT	ConFiGurable Look-Up Table
SPN	Substitution-Permutation Network

Miscellaneous

IoT	Internet of Things
HD	Advanced Encryption Standard
HW	Rivest–Shamir–Adleman
LSB	Advanced Encryption Standard

Chapter 1

Introduction

Technology, especially communication technologies, has experienced rapid development in past decades. Personal computers and various smart devices have become a part of our lives on a daily basis, both professionally and personally. We often do not even realise all these electronic smart devices containing chips, such as payment cards, biometric passports, etc. Many other smart devices are connected to the Internet for various reasons, which exhibit our privacy.

1.1 Motivation

As implied above, we should give special attention to ensuring the security of computer systems and their users. We know various methods to ensure confidentiality, integrity, availability, and non-repudiation of data with efficiency, ease of use, and cost in mind. Nowadays, widely used cryptographic algorithms are considered secure from the crypto-analytic point of view. However, implementations of these and other algorithms may leak sensitive information through the side channels of the physical device.

Side-channel attacks exploit side channels, e.g., device power consumption and data dependency. Thanks to information leakage, secret information such as cipher keys can be extracted, compromising the whole system. The threat is even bigger considering IoT devices, where the attacker can easily get physical access to a device. Various countermeasures were proposed as a reaction to the existence of side-channel attacks. They can be implemented in both hardware and software and have various levels of security, performance, and cost overheads. Ideal countermeasures should have low overhead and cost and offer straightforward implementation for various algorithms, possibly in both software and hardware.

1.2 Problem Statement

Side-channel attacks are a threat to all cryptographic implementations. Many countermeasures have been proposed, offering various trade-offs considering cost, efficiency, security, effort required for implementation and other aspects of those countermeasures. Also, provably secured countermeasures at arbitrary order are known. However, their other specifications, such as overhead, are often infeasible. Therefore, an approach that does not offer proven security but makes things harder enough for an attacker can be an acceptable solution. These solutions can apply to relatively simple devices, especially in the industry. Considering the hardness of performing the attack, even if not proveable impossible, the attack can be, e.g., practically infeasible or at least too expensive, that the effort is higher than the returns from a successful attack.

1.3 Goals of the Dissertation Thesis

1. Implementation of complex cryptographic algorithms secured by known countermeasures used primarily for lightweight algorithms.
2. Proposal of new countermeasure implemented in hardware offering better or at least competitive combination of security, overhead and implementation complexity.
3. Examine countermeasures used in software implementations and the transferability of their principles to hardware implementations.

1.4 Structure of the Dissertation Thesis

The thesis is organized into ... chapters as follows:

1. *Introduction*: Describes the motivation behind our efforts together with our goals. There is also a list of contributions of this dissertation thesis.
2. *Background and State-of-the-Art*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art.
3. *Dynamic Logic Reconfiguration Based Side-Channel Protection of AES and Serpent*: Describes our implementation of known hardware countermeasures used in lightweight cryptographic algorithms for more complex algorithms published in [A.3].
4. *Dummy Rounds Scheme*: Presents Dummy Rounds scheme – our side-channel analysis hardware countermeasure published iteratively in [A.1, A.2, A.4, A.5].
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, we present an overview of side-channel attacks with a bigger focus on non-profiled attacks. Section 2.2 presents an overview of countermeasures against side-channel attacks. Countermeasures used in programmable hardware are described more deeply. Section 2.3 presents a methodology for leakage assessment of hardware implementations focused on non-specific Welch's t-test as a method used in our research.

Side-channel analysis is a group of attacks that uses digital systems' physical properties to compromise them. In this way, even algorithms such as Rijndael/AES [25, 26] or RSA [79], which are considered secure from the cryptoanalytic point of view and, therefore, widely used nowadays, can be compromised.

To do so, many side channels can be exploited. Possibly exploitable side channels (extra information that can be gathered because of the fundamental way of implementaion) include power consumption [43, 28, 13, 19], electromagnetic radiation [75], temperature [42], combinational logic delay [83, 109], timing [44], and more. Some side channels are not independent, e.g. combinational logic delay is proportionally inversion to the voltage drop (the difference of the voltage at the start and at the and of combinational logic) [69], and the relationship between a magnetic field and current intensity is described with electromagnetic induction [90].

We can classify side-channel attacks in many ways, such as active/passive, invasive/non-invasive or vertical/horizontal [90]. Active attacks manipulate the proper functionality of the attacked device (e.g., by inserting faults); meanwhile, passive attacks only observe the device while running without any disturbance. Invasive attacks require the depackaging of the chip before the attack itself, where the attacker accesses internal parts such as data buses. Non-invasive attacks do not require anything like that, and all the data an attacker needs can be collected just by observing and exploiting external access. Vertical attacks use information from multiple measurements at the same time point, so we need multiple runs of the device under attack to collect sufficient data (e.g. power traces), when performing vertical attacks. Then, the measured data is used to resolve the attack, typically with a statistical method. Horizontal attacks are rare and work rather as the examples of attacks

against naïve implementations. Considering naïve hardware RSA implementation using square and multiply exponentiation, we can easily distinguish either square and multiply (bit value one) or only square operations (bit value zero) are performed for every bit of exponent. The usage of square and multiply or square only influences the execution time of the encryption. Also, in some cases, we can directly read the secret key from a single measured power trace as the square and square and multiply operations form different patterns [44].

2.1 Side-Channel Attacks

It began in 1999 when Paul Kocher et al. introduced the first two side-channel attacks, which became a foundation of side-channel analysis research – Simple Power Analysis and Differential Power Analysis [43]. Till now, power consumption has been the most commonly used side channel, which is also the primary focus of our research. This thesis then focuses on dynamic power consumption side-channel attacks and primarily countermeasures against them. However, the majority of attacks described in this thesis can be used with other side channels than power consumption.

2.1.1 Non-Profiled Attacks

Simple Power Analysis is a way to identify particular encryption operations from just a few or even a single power trace and reveal the secret information. This attack is especially efficient against naïve implementations of asymmetric ciphers based on modular exponentiation like RSA [44], where we can visually identify square and square and multiply operations. Later, this attack evolved to other types of ciphers, e.g. to exploit key expansion procedures in AES [51, 26].

Differential Power Analysis – the second attack presented by Kocher et al. [43] – exploits the fact that the power consumption of the device under attack depends on currently processed data. This attack presumes that an attacker is able to send arbitrary input data (e.g. plaintext) to the device and then is able to measure the power consumption during runs with this data inserted.

The Differential Power Analysis attack starts with measuring the device's power consumption under attack with random plaintexts at the input. The amount of traces required for a successful attack depends on the device and the amount of leaked information. The key is then guessed one part after another in this way:

We use a function of a key and plaintext or ciphertext to compute a hypothetical intermediate internal value of the implemented encryption algorithm. We compute those predicted intermediate values for each part of the key (e.g. byte of the key), each key candidate and each measured encryption. Then, for each combination of key candidate and part of the key, we split the traces into two sets according to the value of the chosen

bit (e.g. LSB). We compute the means of these sets and then the difference of the means[1]. When a wrong key candidate is assumed, then the difference will be nearly constant (with some small noise peaks) and very close to zero (the traces are theoretically uniformly distributed in both sets). Otherwise, with the correct key candidate, major peaks will appear at time points where the predicted intermediate value is being processed due to the bias caused by the fixed bit. In this way, we can extract the whole secret cipher key part by part.

Differential Power Analysis described by Kocher et al. [43] became the foundation of a highly researched topic. Since then, many improvements and related topics have been published. For example, Quisquater and Samyde [75] proposed an attack using electromagnetic radiation based on DPA. Messerges et al. [59] proposed a *Multi-bit Differential Power Analysis*, where the measured power traces are grouped into two sets by Hamming weight of predicted hypothetical value. Another approach deals with power consumption estimates based on its model (e.g. Hamming weight of hypothetical value or Hamming distance between that value and the consequent value in the previous/next stage of encryption). Then, the correlation between these estimations and measured power traces is done; thus, this attack is called *Correlation Power Analysis* [13]. The result of the attack is the key with the highest correlation (absolute value) between the estimation and traces. Correlation power analysis assumes a linear relationship between the estimation from a model and the physical observation [24]. Using the Spearman coefficient instead, this requirement is relaxed to the monotonicity requirement [5].

Gierlichs et al. proposed *Mutual Information Analysis*, which ranks the key candidates using mutual information between the hypothetical leakage and the measured power consumption [31]. Mutual information can be interpreted as the amount of information about leakage from the device under attack obtained by observing its power consumption. With this attack, the problematic task is to estimate the probability densities of leakage model functions, whereas with the trivial ones, the attack will always fail [103, 91, 106]. Another attack is *Kolmogorov–Smirnov Analysis* [103, 105], which, unlike Mutual Information Analysis, can be used with a trivial identity leakage model and, therefore, without precise knowledge about leakage of implementation [105]. In recent years, *Differential Deep Learning Analysis* using neural networks was proposed [17]. For the attack itself, various deep-learning architectures may be used.

2.1.1.1 Higher-Order Analysis

The Higher-Order moment based attack means that a sample is analyzed in a higher statistical moment [43]. There are two variants, univariate and multivariete. Multivariete higher-order analysis, combines samples in more time points of a trace, which is helpful when the predicted value is processed at different times, typically for sequentional software implementations or when some hardware timing (hiding in time) countermeasure (described later in Section 2.2.2) is implemented. Instructions on how to find timing points

[1] In more recent research, the statistically more correct t-test is used to test distribution averages instead of the difference of the means.

for multivariete higher order analysis were proposed in [84], highlighting possible problems with this quite complex task. When the intermediate values are (masked and) processed in parallel, so they influence the power traces simultaneously, we use univariate fashion of higher-order analysis. Higher-Order moment based attack are typically realized in a similar fashion like first-order attacks after the traces are preprocessed as described in [68].

2.1.2 Profiled Attacks

To have a fully controlled identical copy of the device under attack allows profiled attacks to be performed. The attacker can observe the copy (its side channels) during any required operation, primarily the identical cryptographic implementation with arbitrary inputs and keys. These attacks are tailored for a specific device and implementation; thus, the number of required traces to measure and analyze compared to non-profiled attacks is much lower. Profiled attacks consist of two phases, where first, thanks to observation, the empirical leakage model is created, and then the attack with this accurate leakage model is made. The most known profiled attack is the *Template Attack* [19, 76].

Also, a lot of *Machine Learning-Based Attacks* have been proposed, which can solve a problem without being explicitly programmed to solve that problem. Considering the side-channel analysis, the learning phase is used to build an empirical model by observing the device, and then during the solving phase, the real data is evaluated with the usage of the model. So, profiled side-channel attacks can be reduced to a general classifying task in the context of machine learning [45]. The attack itself is performed similarly to a Template attack, and they are often classified as one. The difference is in the model, where a machine-learning-based classifier is used. Many different machine-based learning classifiers can be used, such as support vector machines [41, 40, 4], decision trees or random forests [46, 47]. Neural network-based deep learning classifiers are a popular choice considering side-channel security [8, 39], where both multilayer perceptron [55, 54] and convolutional neural network [16, 50] can be used to perform a profiling attack.

2.2 SCA Countermeasures

Over the years of side-channel analysis research, many countermeasures against the attacks were proposed. In this section, we describe them with a focus on the countermeasures implemented in hardware, as it is the field of this thesis. We also briefly describe some countermeasures implemented in software, which ideas can be implemented in hardware, and which inspired our Dummy Rounds described in Chapter 4.

Still, even the correct implementation of proposed countermeasures does not provide absolute security. The goal of the countermeasures is to make an attack infeasible. It is made typically by increasing the number of measurements necessary for a successful attack, making the attack practically unavailable or at least unfavourable in comparison with the cost of the attack. With increasing computational capacities worldwide, the proposed

real-world attack example shows that resilience against many measurements would be needed [49].

The side-channel attack countermeasures can be divided into two basic groups: Masking and Hiding [52].

2.2.1 Masking

The most common masking method is Boolean masking [18], also multiplicative [34], or affine masking [30] can be used. With each of these masking techniques used, implementation of masking countermeasures requires good knowledge of the cryptographic algorithm because masking randomizes the internal intermediate values of encryption while the result is still correct. In the ideal case, the intermediate values inside the device appear absolutely random and are independent of the intermediate values of an unmodified encryption algorithm.

In Boolean masking, which is assumed further in this section, sensitive intermediate value x is split into $d+1$ shares x_i, where

$$x = \bigoplus_{i=0}^{d} x_i. \tag{2.1}$$

For the splitting, d uniform random masks $x_1, \ldots, x_d$, where $x_0 = x \oplus x_1 \oplus \cdots \oplus x_d$ are used. According to the number of masks d, we call the final implementation d-order masked, which should be ideally secured against attacks up to d-th order [18, 74] as described in Section 2.1.1.1. In practice, achieved security is often lower because of unpredicted weaknesses [53, 61].

When masking a typical cryptographic algorithm, which consists of several linear and nonlinear operations, some of them have to be changed to preserve the correct result. With linear operations, it is trivial because the operations with shares can be made independently:

$$f(x) = f(x_0 \oplus \cdots \oplus x_d) = f(x_0) \oplus \cdots \oplus f(x_d). \tag{2.2}$$

The typical non-linear operation, considering substitution-permutation network-based ciphers focused on in this thesis, is substitution boxes (S-boxes). The most common approach is to pre-compute masked S-boxes, which were proposed first for software implementations [60] and later adapted for hardware implementations [36, 82]. One of the vulnerabilities of these implementations is data-dependent glitches observable during S-box computations [53]. This problem is solved by glitch-resistant masking schemes, which were proposed, like Threshold implementation [66, 67, 10], Domain-oriented masking [35] or Consolidated masking schemes [77]. Threshold implementation is provably secured to an arbitrary d-th order. However, its required properties for non-linear transformations (correctness, non-completeness and uniformity) cause huge overhead for implementations of more complex ciphers with the function of higher algebraic degree [10, 11].

2.2.2 Hiding

The main objective of hiding is to remove the dependency of processed data and side-channel output from an attacker's point of view. However, the information is still present, therefore *hiding*. In other words, remove the dependency of processed data and side-channel output from an attacker's point of view. There are, in general, two options: how to hide the value, hiding in amplitude and hiding in time.

Hiding in amplitude is naturally used, especially with hardware implementations, which are much closer to the physics of the device. Hiding in amplitude generally focuses on the reduction of signal-to-noise ratio with, e.g. noise generation [48], switching capacitors [85, 73] or pipelining [92]. Another hardware approach helpful in hiding in amplitude is Dual-Rail Logic [27, 94, 98] or even Quadruple-Rail Logic [99], which can be combined with precharge logic. Designs using this countermeasure are built with special logic gates with the same switching activity for any logic transition. The value is given by a combination of signals on two/four semi-wires, which have the opposite value (their sum is constant), so the power consumption is constant and independent of internal values. Many schemes using this principle exist, e.g. WDDL [98, 97] or STTL [87]. An overview of Dual-Precharge Logic techniques is presented in [27].

Hiding in time is typically achieved by the usage of a specific clock signal or by modifications of the cryptographic algorithm. The clock signal can be randomized [36] or isolated into the clock network and so unavailable to an attacker for synchronization [72]. There can be some random delays added both in hardware way by D flip-flop with a randomized propagation [15] and software by placing dummy cycles [22] or random delays [23, 100].

2.2.2.1 Shuffling

Although its effect is similar to that of hiding, Shuffling [102] is sometimes, just as in this thesis, considered a separate category. As it is implemented on the algorithm level, it is naturally implemented in software (or in hardware as processor extensions). Shuffling randomizes the algorithm flow and then breaks the dependency of data and leakage, or at least hides the sensitive value to another time point than an attacker assumes.

Shuffling randomizes the algorithm flow and then breaks the dependency of data and leakage, or at least hides the sensitive value to another time point than an attacker assumes. This approach is usually used in software. It can be implemented, e.g. as random order execution [96, 78], or achieved by processor extensions, making them non-deterministic [7, 56].

The technique called *Dummy Rounds* appeared before in software implementation [32]. The Dummy Rounds method is similar to some other software countermeasures, such as Dummy Cycles [22], which is hiding in time, or shuffling methods [96, 102]. Dummy Rounds were studied, in conjunction with other methods, as a countermeasure against fault and combined attacks [32, 70, 71]. Usage of Dummy Rounds has been also proposed as a DPA countermeasure [38]. As some of the previous applications were shown to be flawed [6, 101],

we limited our study to mere DPA. However, the principle is still the same as in the other software implementations – insertion of dummy round function instructions.

2.2.3 Countermeasures in FPGAs

Considering programmable hardware, especially FPGAs, its reconfigurability is an opportunity to implement specialized countermeasures. For example, Mesquita et al. in [58] proposed a coarse grain reconfigurable architecture based on leak-resistant arithmetic [2]. Specific hiding countermeasures can be implemented with temporal jitter [57] or FPGA-specific design units [36]. Dynamically adjustable decomposition of S-box using a configurable look-up table (CFGLUT) reconfiguration can be used as a specific masking technique [80].

Implementation of some countermeasures described above on FPFA was proposed in [82]. These countermeasures are

1. S-Box Random Decomposition,
2. Boolean Masking,
3. Register Precharge.

As a case study, the PRESENT [12] cipher was used. Its S-Boxes functions S are randomly decomposed into two functions R_1, R_2 in such a way that

$$\forall x, R_2(R_1(x)) = S(x). \tag{2.3}$$

The registers are there between functions R_1 and R_2, so the S-box output is never stored in a register. The stored values are also masked using Boolean masking described in Section 2.2.1. Register precharge is then hiding in amplitude method, where a random value is stored in the registers between storing two consecutive intermediate values. Otherwise, a leakage by storing the consecutive round inputs can be easily detected by a Hamming distance (HD) model as

$$HD(x \oplus m, y \oplus m) = HW(x \oplus y). \tag{2.4}$$

This implementation is used during our research to compare our designs and implementations.

2.3 Leakage Assessment

Considering the amount of side-channel attack countermeasures proposed in more than twenty years and described in the previous section, a method for evaluating and comparing the countermeasures is needed. A naïve approach to mount all the attacks would be time-consuming and expensive. Instead, the Leakage assessment methodology examines whether there is an information leakage from the tested device in a more general way, requiring less

less time and computational resources. The methods described in this section are based on dividing measurements into two or more sets and examining the difference between those sets.

Several tests for Leakage Assessment were proposed. First, statistical tests based on Mutual Information were proposed [20, 21]. These tests require an estimate of the probability distribution. Later tests based on Student's t-distribution were proposed [33]. Welch's t-test, one of them, will be described later as the test is used in our research. This test was also used for leakage assessment without a deeper examination of the procedure in some works, e.g. [3, 10]. There is an extensive evaluation of leakage assessment using Welch's t-test described in [84], and it became a widely used method. Other methods were recently published, e.g. chi-square test-based [64] or deep learning-based [62]. For all these tests, the evaluater's full control over the implementation is assumed.

The Leakage assessment tests can be categorized as specific or non-specific tests. The specific tests [33] typically evaluate measurements of random uniform plaintext with a fixed key. These measurements are divided into two or more groups according to the intermediate value – groups are distinguished by an expected leakage considering the chosen leakage function and known input plaintext. Hypothetical statistical distinguishability of the groups suggests an information leakage from the chosen sensitive bit(s), although not suggesting anything about its usability for an attack.

The non-specific tests [33, 84] then do not target any chosen specific sensitive intermediate value but evaluate the traces as a whole. Measurements of an encryption of either random uniform plaintext or of a pre-selected fixed plaintext are typically used. Such tests are called *random vs. fixed tests*. The other option is a fixed vs. fixed test, where two different pre-selected plaintexts are used. In both options, the measurements of both types (traces measured into both groups) have to be randomly interleaved to prevent false results caused by external effects such as environmental noise [84]. Once again, a distinguishability between the groups of traces suggests an information leakage. Non-specific tests are more sensitive and general, although, on the other hand, they provide only limited information about the leakage as they evaluate the traces as a whole.

2.3.1 Non-Specific Welch's t-Test

In this section, we focus on non-specific, fixed vs. random Welch's t-test, as it is proposed in [84] and also used for leakage assessment during our research in this thesis. With a two-tailed Welch's t-test, we examine a null hypothesis that two groups' means are equal; hence, we use it to test whether traces measured with fixed/random plaintext fit into one same population. Welch's t-test is a generalization of the Student's t-test for populations with different variances [104]. Welch's t-test is a univariate moment-based statistic, so the measurements must be aligned. We can compute the Welch's statistic t for the two groups in every sampling point independently as

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \tag{2.5}$$

where μ_1, μ_2 are sample means, s_1^2, s_2^2 are sample variances, and n_1, n_2 are cardinalities of the first, respectively the second group. The number of degrees of freedom v then can be estimated as

$$v \approx \frac{(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2})^2}{\frac{(\frac{s_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{s_2^2}{n_2})^2}{n_2 - 1}}. \tag{2.6}$$

No assumptions about the implementation are needed when assuming a non-specific test such as this. We measure traces and divide them into two groups according to the plaintext, which is fixed or random. To avoid statistically false results, we have to interleave both sets as mentioned before, e.g. with a coinflip before each measurement. Then, we compute the results. Under the null hypothesis, the statistic t follows Student's t-distribution with v degrees of freedom. At the same time, for simplicity, a threshold ± 4.5 or ± 5 is usually defined to reject the null hypothesis without considering the degrees of freedom v. The threshold roughly corresponds to a significance level $\alpha \leq 10^{-5}$. Therefore, having $|t| > 5$ means that with a significance level $\alpha \leq 10^{-5}$, the null hypothesis is rejected, and it suggests an information leakage (the traces are not from the same population). However, it says nothing about the leakage and its usability for an attack. Not rejecting the null hypothesis suggests nothing; most importantly, it does not suggest the implementation is leakage-free. Welch's t-test can also be used as a higher-order leakage assessment, which is realized in the same fashion (using preprocessing) as higher-order attacks described in Section 2.1.1.1.

Chapter 3

Dynamic Logic Reconfiguration Based Side-Channel Protection of AES and Serpent

In this chapter, we extend the work presented in [82] by using dynamic logic reconfiguration to secure two of the Advanced Encryption Standard (AES) competition finalists, Rijndael [26] (winner of the competition, nowadays therefore known as the AES) and Serpent [9]. We describe our implementations and the non-straightforward way in which we tailored the countermeasures in [82] to AES and Serpent. We evaluate the side-channel leakage and the effectiveness of different countermeasure combinations.

3.1 Theoretical Background

In this work, we intend to secure AES and Serpent using the approach described in [82]. In the following subsections, we first describe both AES/Rijndael and Serpent. Then we explain the concept of dynamic logic reconfiguration on FPGA, and finally we describe the implemented and evaluated countermeasures.

3.1.1 AES Finalists: Rijndael and Serpent

Both ciphers share common features [65]. They are iterated substitution-permutation networks (SPN) with a block size of 128 bits and possible key sizes of 128, 192 or 256 bits. The plaintext (i.e. the data to be encrypted) is transformed into a ciphertext by iteratively applying a number of operations. Each iteration is called a round. Both ciphers also describe a method for expanding the secret key into a number of subkeys which are used as an input to each round.

3.1.1.1 AES/Rijndael

Rijndael [26] consists of 10, 12 or 14 rounds (depending on the key length). First, the secret key is XORed with the plaintext. After that, a number of round transformations is performed. Each round consists of four layers: a non-linear substitution layer (SubBytes, i.e. 16 parallel applications of an 8-bit substitution box or S-box), two linear mixing layers (ShiftRows and MixColumns) and a XOR with the round subkey (AddRoundKey). In the last round, the MixColumns transformation is omitted.

3.1.1.2 Serpent

Serpent [9] consists of 32 rounds. First, an initial permutation is applied and then the round transformations take place. Each round consists of three layers: a XOR with the round subkey, a non-linear substitution layer (i.e. 32 parallel applications of one of the eight specified 4-bit S-boxes, which are different in the consecutive rounds), and a linear transformation. In the last round, a second XOR takes place instead of the linear transformation. In the end, the final permutation is applied.

3.1.2 Dynamic Logic Reconfiguration

In FPGAs, combinational circuits are typically implemented using Look-Up Tables (LUTs), i.e. configurable primitives which store truth tables of k-input Boolean functions $f : \mathbb{B}^k \to \mathbb{B}$. Dynamic logic reconfiguration allows for the run-time alteration of the circuit behaviour by modifying the content of specific look-up tables, while leaving the routing intact. The reconfiguration of LUTs is done from within the chip itself and can be achieved e.g. by using a shift register (allowing for serial programming) and a cascade of addressing multiplexers. In Xilinx FPGAs [107], this functionality is provided by k-input Configurable Look-Up Tables (CFGLUTs) with a serial configuration input and output (allowing to connect CFGLUTs in separately configurable chains). In Xilinx Spartan-6 FPGAs, 5-input CFGLUTs are available.

In order to implement dynamically reconfigurable Boolean functions $f : \mathbb{B}^n \to \mathbb{B}$, where $n > k$, multiple k-input CFGLUTs are required in combination with addressing multiplexers (using Boole's expansion, also referred to as the Shannon expansion [14]). Specifically, to implement an n-input function using k-input CFGLUTs and 2-to-1 multiplexers, we need 2^{n-k} CFGLUTs and $2^{n-k} - 1$ multiplexers.

Multiple-output Boolean functions $f : \mathbb{B}^n \to \mathbb{B}^m$ can be trivially implemented as m single-output Boolean functions $f_i : \mathbb{B}^n \to \mathbb{B}$.

3.1.3 Countermeasures

To protect AES and Serpent, we have implemented countermeasures that were proposed (and evaluated on PRESENT) by Sasdrich et al. in [82]. In this subsection, we briefly describe these countermeasures.

3.1.3.1 S-box Decomposition

Since information leakage often occurs based on changing values in registers, and since the output of the non-linear substitution layer is a frequent target of side-channel attacks, the S-box decomposition countermeasure is based on avoiding the storage of the S-box outputs into such registers. This is done by decomposing the S-box into two bijections R_1, R_2, where

$$\text{S-box}(x) = R_2(R_1(x)), \tag{3.1}$$

and placing the register in between the two bijections. The number of possible n-bit bijections for R_1 is equal to $(2^n)!$. For each option, a bijection R_2 can be found such that Eq. (3.1) holds.

Thanks to dynamic logic reconfiguration, different bijections R_1, R_2 can easily be used for every encryption. Starting with R_1 being an identity and R_2 being the actual S-box, the bijections for the next encryption are computed by randomly selecting two pairs of elements in the R_1 mapping, swapping them, and recomputing R_2 accordingly.

3.1.3.2 Boolean Masking

In order to randomize intermediate values, a random mask is added (XORed) to the data prior to encryption, and subtracted (i.e. once again XORed) after the encryption. For the cipher to produce valid results working with masked data, various alterations must be done.

Boolean masking can be combined with the previously mentioned bijective S-box decomposition and can once again take advantage of dynamic logic reconfiguration. Two different random masks m_1, m_2 are used for every encryption: mask m_1 is used outside the decomposed S-box, and mask m_2 is used inside of it. If the substitution layer would be the only layer in the round, the previously mentioned bijections R_1, R_2 would get adjusted as follows:

$$R'_1(x) = R_1(x \oplus m_1) \oplus m_2, \tag{3.2}$$
$$R'_2(x) = R_2(x \oplus m_2) \oplus m_1. \tag{3.3}$$

The function R'_1 first subtracts/removes mask m_1, then performs the R_1 bijection mapping, and finally masks this value using m_2. The output of this function is stored in the register. Analogically, the function R'_2 subtracts the mask m_2, does the R_2 mapping and masks the result using m_1. This way, the same CFGLUTs can be used for both the S-box decomposition and the masking, saving both area and reconfiguration time.

However, to deal with the linear transformation layers, further alterations to the R'_1, R'_2 bijections need to be done. We can exploit one of these two facts:

$$f(x) = f(x \oplus f^{-1}(m)) \oplus m, \tag{3.4}$$
$$f(x) = f(x \oplus m) \oplus f(m), \tag{3.5}$$

which both hold when $f(x)$ is a linear mapping. These give us two different and fairly straightforward approaches to take linear transformations $f(\cdot)$ into account.

One option is to alter R'_2 function in terms of Eq. (3.4) so that m_1 processed by the inverse transformation is used to mask the data, allowing to subtract m_1 in R'_1:

$$R'_1(x) = R_1(x \oplus m_1) \oplus m_2, \tag{3.6}$$
$$R'_2(x) = R_2(x \oplus m_2) \oplus f^{-1}(m_1). \tag{3.7}$$

The second option is to use m_1 for masking in R'_2, and to alter R'_1 according to Eq. (3.5), so that m_1 processed by the linear transformation gets subtracted:

$$R'_1(x) = R_1(x \oplus f(m_1)) \oplus m_2, \tag{3.8}$$
$$R'_2(x) = R_2(x \oplus m_2) \oplus m_1. \tag{3.9}$$

Notice that further alterations may be required for the first and the last round, depending on the selected approach.

The last obstacle is the subkey XOR layer, which can be considered an affine transformation. Suppose we have a vector x, which gets XORed with the subkey: $x \oplus k$. Suppose we process masked data the same way: $(x \oplus m) \oplus k$, then by subtracting the mask m with no alterations we have:

$$((x \oplus m) \oplus k) \oplus m) = x \oplus k. \tag{3.10}$$

Therefore, no further alterations need to be done to take the XOR layer into account.

3.1.3.3 Register Precharge

Because the same masks are used for the whole encryption (i.e. for every round), the leakage occurs in the register, since

$$\mathrm{HD}(x \oplus m, y \oplus m) = \mathrm{HD}(x, y), \tag{3.11}$$

where $\mathrm{HD}(x, y)$ denotes the Hamming distance between x and y. To avoid this leakage, the register is duplicated and the processed data are interleaved with random data. This technique avoids leakage, however, it reduces the throughput of the circuit when it is implemented using an architecture that is not fully unrolled.

3.2 Secure Cipher Design

In this section, we examine the specifics of both AES/Rijndael and Serpent and we propose a manner in which these ciphers can be secured against side-channel attacks using the countermeasures explained in Section 3.1.3.

In order for our implementations to fit into a Xilinx Spartan-6 FPGA device, we take into account that CFGLUTs with at most 5 input bits are available. When a platform with smaller CFGLUTs is available, the dynamic logic reconfiguration method can be implemented using the approach described in Section 3.1.2.

3.2.1 AES/Rijndael

Rijndael employs an 8×8 S-box, which can be considered as a function $\text{S-box}_{\text{Rijndael}} : \mathbb{B}^8 \rightarrow \mathbb{B}^8$. Therefore, to implement the Rijndael S-box using reconfigurable logic, $8 \cdot 2^{8-5} = 64$ (5-input) CFGLUTs and $8 \cdot (2^{8-5} - 1) = 56$ (2-to-1) multiplexers are necessary. Moreover, the S-box decomposition countermeasure suggests the S-box to be split into two bijections $R_1, R_2 : \mathbb{B}^8 \rightarrow \mathbb{B}^8$, which doubles the amount of CFGLUTs and multiplexers in the secured version. Since the Rijndael algorithm applies 16 S-boxes in parallel, this brings the total count up to 2048 (5-input) CFGLUTs and 1792 (2-to-1) multiplexers.

The decomposition into two bijections is done in a similar fashion as described in Section 3.1.3, with the round register being placed in between the two bijections. For the AES algorithm, we have decided to swap 8 pairs of elements in the R_1 bijection after every encryption (in contrast to the PRESENT 4-bit S-box decomposition in [82], where only two pairs get swapped).

To implement the Boolean masking countermeasure as described in Section 3.1.3, bijections R_1', R_2' (i.e. the decomposed S-box combined with masking) must be altered. We choose the option where R_2' adds the mask m_1 and R_1' subtracts m_1 processed by the linear transformations (see Eq. (3.8)):

$$R_1'(x) = R_1(x \oplus \text{MixColumns}(\text{ShiftRows}(m_1))) \oplus m_2, \tag{3.12}$$

$$R_2'(x) = R_2(x \oplus m_2) \oplus m_1. \tag{3.13}$$

Note that the data are masked by m_1 in the second bijection R_2 and that this mask is subtracted in the following round. Therefore prior to the first round, the input data must be masked properly. Also, the last round of Rijndael omits the MixColumns operation. Therefore, in the last round, only $\text{ShiftRows}(m_1)$ must be subtracted in R_1', or additional unmasking of the output must be done (which is our choice).

The implementation of the register precharge requires the register to be duplicated and the controller to be adjusted appropriately, such that the processed data are interleaved with random data.

3.2.2 Serpent

Unlike Rijndael or PRESENT, Serpent defines eight different 4×4 S-boxes. Each S-box is used in a different round. One way to implement the S-box decomposition is to decompose each of these S-boxes into two bijections, resulting in 16 bijections in total. We have decided for an approach where the first bijection R_1 is shared among all S-boxes, while the other 8 bijections R_2^i, $i \in \{0, .., 7\}$, implement the eight S-boxes, with the correct output being selected by a multiplexer. The eight decomposed Serpent S-boxes are depicted in Figure 3.1. Notice the **demultiplexer**, which selects the right R_2^i bijection, while the other bijections are fed with zeroes. This demultiplexer is necessary to prevent glitches that lead to information leakage. Since the Serpent S-boxes realize the functions $\text{S-box}^{\text{i}}_{\text{Serpent}} : \mathbb{B}^4 \rightarrow \mathbb{B}^4$, only four CFGLUTs are necessary to implement the bijection. Given the selected architecture, $4 + 8 \cdot 4 = 36$ CFGLUTs are required to decompose all

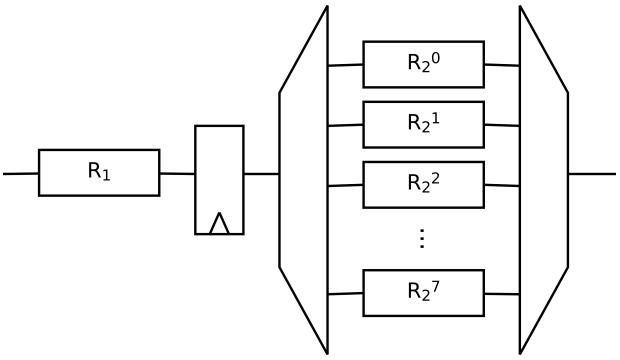


Figure 3.1: Serpent S-box Decomposition

eight S-boxes. Since the S-box is applied 32 times in parallel, this results in 1152 CFGLUTs in total.

Boolean masking is implemented similarly to the Rijndael algorithm, with m_1, processed by the linear transformation, being subtracted in the R'_1 bijection (see Eq. (3.8)). Suppose the Serpent linear transformation is $L_{Serpent}$, then:

$$R'_1(x) = R_1(x \oplus \mathrm{L_{Serpent}}(m_1)) \oplus m_2, \tag{3.14}$$

$$R'_2(x) = R_2(x \oplus m_2) \oplus m_1. \tag{3.15}$$

Regarding the first round, similarly to the Rijndael approach, appropriate initial masking of the input data must be performed first. Also, there is no linear transformation in the last round, therefore, either the unprocessed mask m_1 gets subtracted in R'_1, or final unmasking must be performed.

Register precharge is once again implemented simply by duplicating the round register and altering the controller appropriately to interleave the processed data with random data.

3.2.3 Reconfiguration Controller

For every encryption, new bijections are generated (as described in Section 3.1.3), as well as new masks m_1, m_2. This requires the CFGLUTs configurations to be computed and loaded prior to every encryption. The reconfiguration of all CFGLUTs can be done using different levels of parallelism (the CFGLUTs "programming" I/O can be variously chained, given its shift register nature).

3.3 Side-Channel Leakage Evaluation

In this section, we present our experimental set-up and a leakage methodology used to evaluate all combinations of previously described countermeasures.

3.3.1 Set-up and Methodology

We choose the Sakura-G board [37] with a Xilinx Spartan-6 FPGA as our evaluation platform. AES/Rijndael and Serpent VHDL implementations with a 128-bit key are evaluated. The power traces are measured using a PicoScope 6406D oscilloscope. Leakage is evaluated using the non-specific univariate first-order Welch's t-test as described in [84]. For every evaluation, 1 million power traces are captured.

The necessary random data (random pairs to be swapped in the bijection, random masks, register precharge with random values) are generated externally and sent to the cryptographic device alongside the plaintext. This approach allows us to easily enable or disable specific countermeasures.

3.3.2 Results

We evaluate every possible combination of the proposed countermeasures:

(a) Unprotected

(b) Register Precharge

(c) Masking

(d) Masking + Register Precharge

(e) S-box Decomposition

(f) S-box Decomposition + Register Precharge

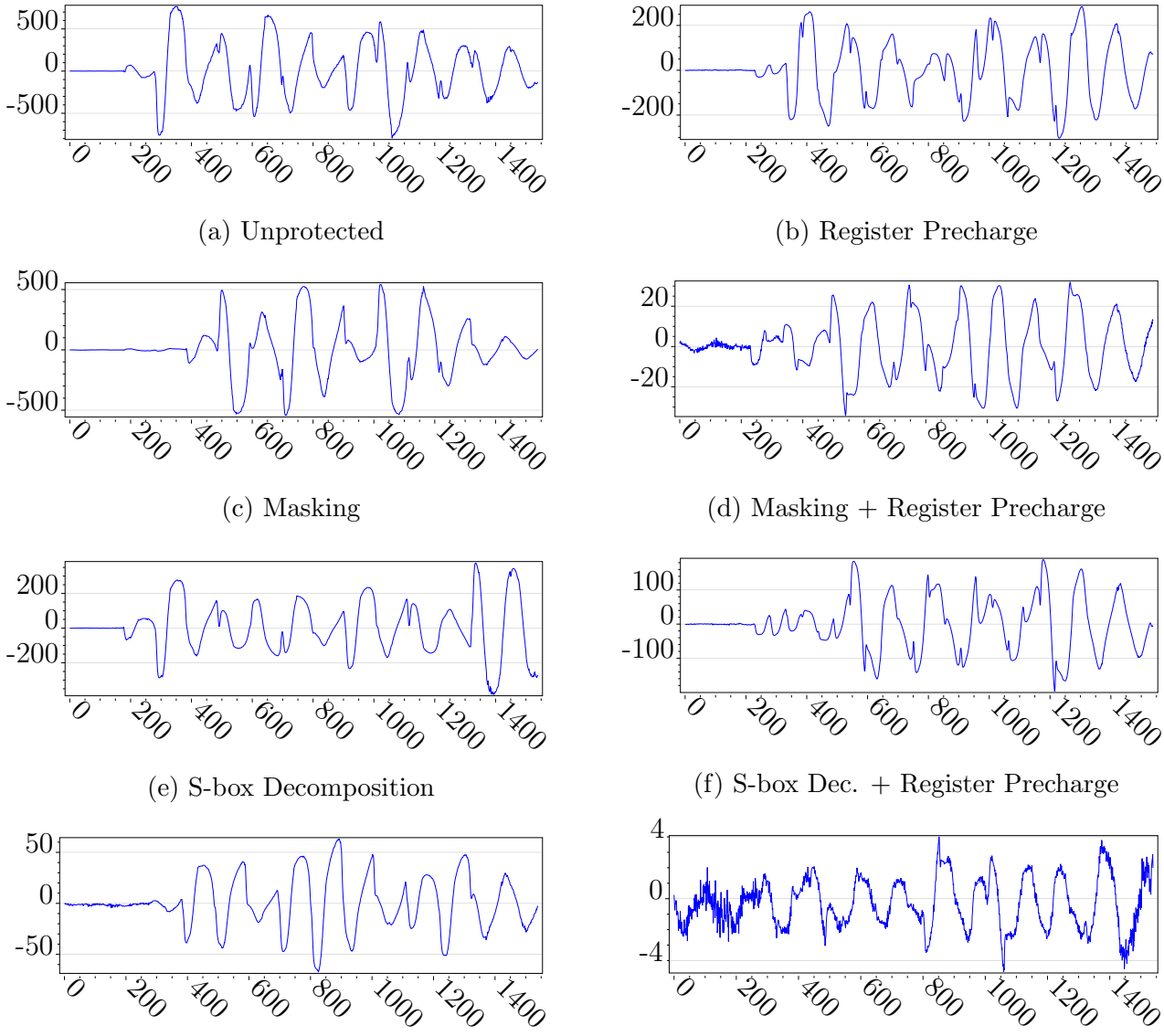
(g) S-box Decomposition + Masking

(h) S-box Decomposition + Masking + Register Precharge

For every implementation, 1 million power traces are measured and processed using a non-specific first-order t-test, as described earlier. Figure 3.2 depicts the t-values during the AES encryption and Figure 3.3 depicts the t-values during the Serpent encryption. The sensitive information leakage is the most prominent for the unprotected versions, as expected.

It is also visible that different countermeasures and their combinations have various influence on the significance of the detected leakage. Figures 3.2c and 3.3c show that a countermeasure based on masking only protects the first round of the cipher, while, starting from the second round, the leakage is comparable to the unprotected version (cf. Figures 3.2a and 3.3a). Figures 3.2d and 3.3d suggest that masking becomes more effective in combination with register precharge (which is expected, as discussed in Section 3.1.3).

Figures 3.2h and 3.3h show results with all three countermeasures combined. As can be seen, no significant first-order leakage is detected when evaluating these fully protected implementations. Therefore we need to combine all presented countermeasures to get first-order secured implementations.



(a) Unprotected

(b) Register Precharge

(c) Masking

(d) Masking + Register Precharge

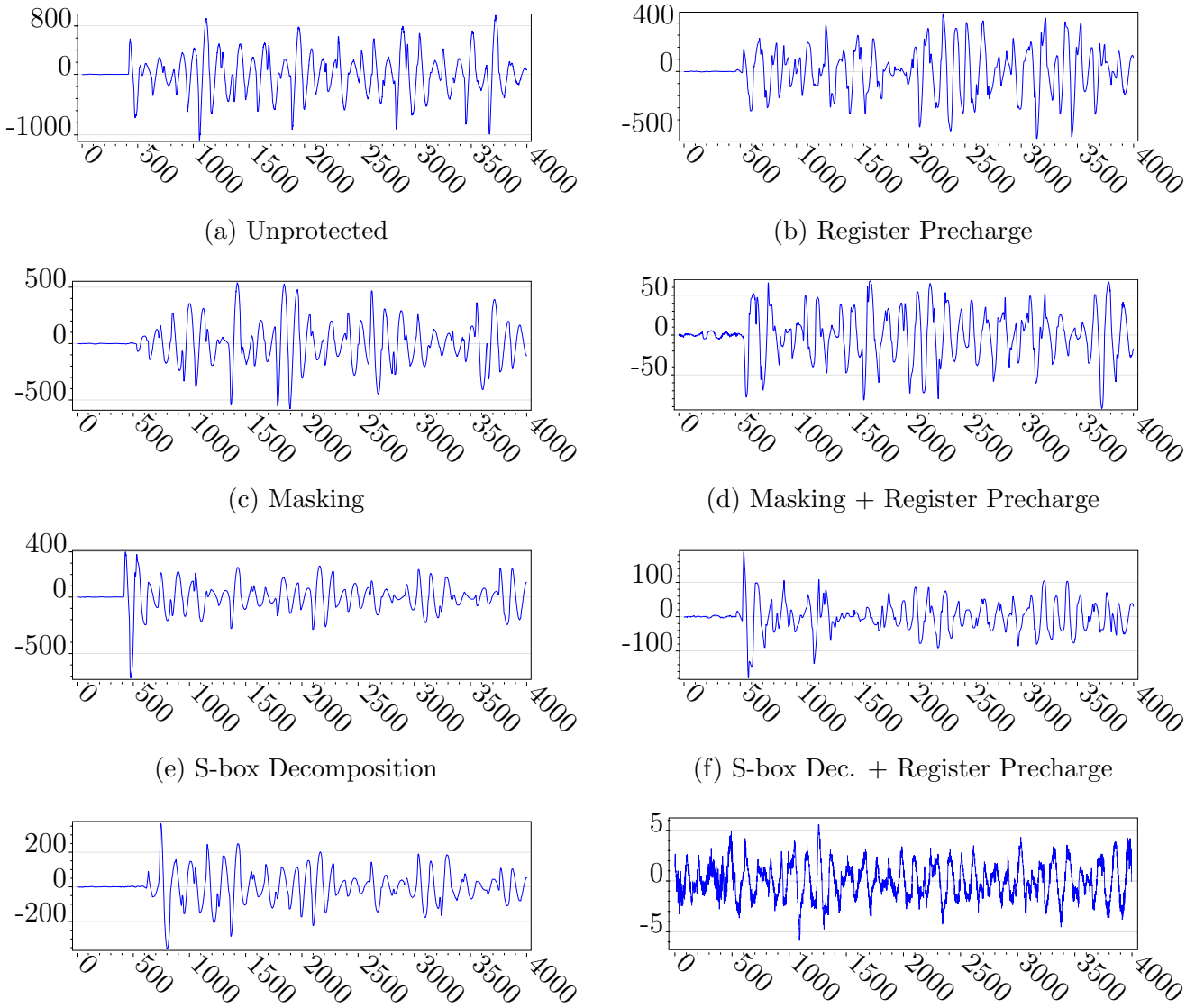
(e) S-box Decomposition

(f) S-box Dec. + Register Precharge

(g) S-box Decomposition + Masking

(h) S-box Dec. + Masking + Reg. P.

Figure 3.2: Results of the AES/Rijndael t-test, where the t-value is shown on the vertical axis and the time samples during encryption are shown on the horizontal axis



(a) Unprotected

(b) Register Precharge

(c) Masking

(d) Masking + Register Precharge

(e) S-box Decomposition

(f) S-box Dec. + Register Precharge

(g) S-box Decomposition + Masking

(h) S-box Dec. + Masking + Reg. P.

Figure 3.3: Results of the Serpent t-test, where the t-value is shown on the vertical axis and the time samples during encryption are shown on the horizontal axis

3.3.3 Second-Order Evaluation

Using the same power traces as we measured evaluating these implementations, Socha et al. in [88] detected apparent second-order leakage from Serpent encryption, while AES encryption second-order leakage was barely detectable. Furthermore, to provide more confidence about the implementation resilience, they attempted to break the protected AES implementation using second-order DPA and CPA attacks targeting both the first and last rounds. All attacks failed with 1.25 million power traces available.

3.4 Conclusion

In this chapter, we describe our implementation of countermeasures originally presented for PRESENT cipher in [82] and described in Section 2.2.3. We implemented these countermeasures for two of the Advanced Encryption Standard (AES) competition finalists, Rijndael [26] (winner of the competition, nowadays therefore known as the AES) and Serpent [9]. Leakage of our implementations is evaluated using the non-specific univariate first-order Welch's t-test as described in [84] for all combinations of countermeasures – starting with unprotected versions through solo usage of countermeasures ending with fully protected implementations. As can be seen in the results, no significant first-order leakage is detected when evaluating implementations fully protected by our countermeasures. This contribution was published in [A.3].

Socha et al. in [88] presented detected apparent second-order leakage from our Serpent implementation using the same power traces, while AES encryption second-order leakage was barely detectable. Furthermore, they attempted to break the protected AES implementation using second-order DPA and CPA attacks targeting both the first and last rounds and all attacks failed with 1.25 million power traces available.

Chapter 4

Dummy Rounds Scheme

In this chapter, we present our hardware Dummy Rounds countermeasure scheme. In Section 4.1, we describe countermeasures used to inspire our method proposed in [A.1] for the first time. Then, we describe the initial principle of Dummy Rounds, its architecture and operations, and the scheme's control. We present an analysis published in [A.2] of the scheme in Section 4.2. In the analysis, we examine the architectural parameters of the scheme, its slot-level model and probabilities throughout the design and scheme flow. Section 4.3 presents primarily the final version of Dummy Rounds scheme datapath published in [A.4] followed by the controller description published in [A.5] in Section 4.4. Section 2.3 evaluates leakage assessment of the Dummy Rounds case study securing PRESENT cipher followed by the conclusion of this chapter.

4.1 Dummy Rounds Scheme Principle

Cryptographic algorithms are typically iterative; thus, they are implemented by so-called *rounds*, where each of them performs similar computations. Common classes of iterative ciphers are Feistel Networks [29] such as DES [93] or, more recently, Substitution-Permutation Networks [86] such as AES [26] or PRESENT [12]. The similarity greatly simplifies implementation, yet the iterations can be recognized, and some distinctive time points can be set as targets for cryptanalysis. One possible countermeasure is to *hide* them from an attacker. Our research intends to use additional rounds and randomization as a method of hiding power consumption, respectively shuffling both described in Section 2.2.2, preventing side-channel attacks.

The *Dummy Rounds* hardware SCA countermeasure scheme, as we propose, combines software hiding in time, shuffling and common hardware hiding of the circuitry power consumption. There are more parts of hardware design which are executed but their outputs are randomly used or not used for computation in every single clock cycle. So, the structure of the design is the same for every clock cycle and power consumption stays the same. Such a behavior can be seen as a kind of dynamic reconfiguration, used also in other methods [57, 82, 81] described in Section 2.2.3.

While the output of the computation in a single clock cycle changes randomly, the final result stays correct due to round scheduling. Hence, the decision which round to use, although randomized, must follow an algorithm, which we discuss later in this section.

4.1.1 Architecture and Operation

Let us assume a round-based cipher with C rounds. Also, let us assume that we can design a hardware implementation of a round so that at least m and no more than M rounds can be executed in a single clock cycle. Then the Dummy Rounds method can be applied as in Figure 4.1, where $m = 1$ and $M = 3$. Using a fourth input to the multiplexer, $m = 0$ can be implemented. The round control determines which successive result to use. The unused round results will also cause switching activity in each clock cycle, but their results will not be stored in the result register. Constant switching activity is also the principle of hardware countermeasure called hiding [27, 94, 99]. This method is feasible for both cipher structures, Feistel Networks [29] and Substitution-Permutation Networks [86].

There are two important design parameters in the Dummy Rounds application. The maximum number of rounds per clock cycle M determines clock frequency and influences both time and area overhead. The average number of actually used rounds determines the (constant) number of clock cycles needed for execution, and hence influences time overhead.

The constant number of clock cycles parameter avoids possible information leakage caused by extreme random values. Without the parameter, there would be a very small (but still higher than zero) probability, that the design will compute only one round (or other value of m parameter) in each clock cycle. In that case, the design could be attacked nearly as a design without any countermeasure. The only difference is the power consumption of the next implemented rounds. However, if there are assumptions of the first round values, the additional rounds can be predicted. The case of encryption using maximum possible count of clock cycles with M rounds computed is quite similar. With this parameter and a corresponding controller schedule, such a situation cannot occur.

Let us illustrate the architectural parameters on an implementation of the PRESENT cipher. The cipher has 31 rounds and one extra sub-key, which is considered to be another round, so there are total $C = 32$ rounds. Let us assume the original architecture has one round per clock cycle, which is common. Let us further assume we decided to implement $M = 3$ PRESENT rounds per clock cycle, which is a practical choice in most circumstances. With this hardware architecture, we need $N = 16$ clock cycles with 2 actually used rounds per clock cycle on average. The clock period will be approximately three times longer, but 16 clock cycles instead of 32 will be needed for an encryption. Therefore, the time overhead will be approximately 50%. The round logic dominates the design, so that the upper bound on area overhead is 200%.

4.1.2 Rounds Control

The rounds controller has two tasks. The first one is to assure that the correct number of rounds are executed within the designed clock count. The other one is to prevent uniform

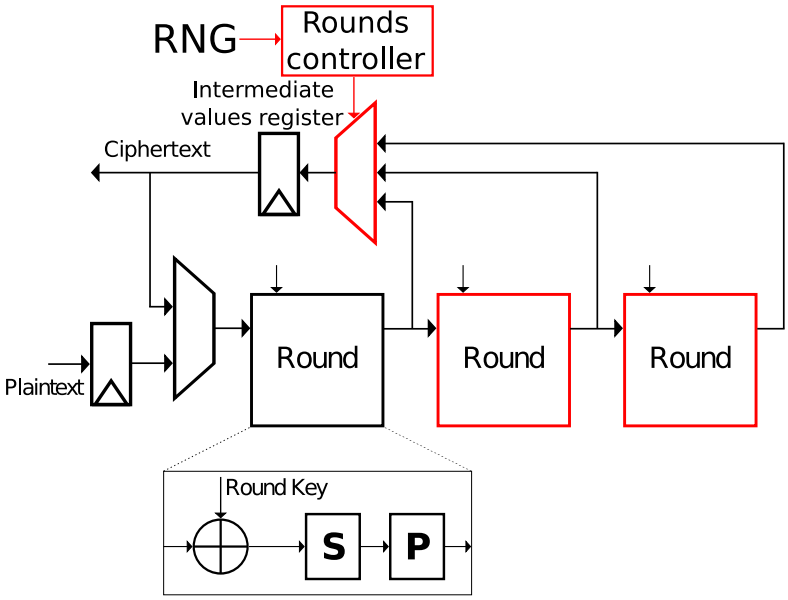


Figure 4.1: Dummy cycles countermeasure scheme.

computations to occur. In our case, the control is implemented in hardware and should be as simple as possible.

For the first task, the controller has to monitor the number of clock cycles executed and the number of used rounds. Let c_n be the number of rounds accepted up to the step n, $n \leq N$. Then, obviously,

$$c_n \leq Mn \tag{4.1}$$

$$c_n \geq mn \tag{4.2}$$

To be able to reach precisely C rounds at step N, the following must hold

$$c_n + m(N - n) \leq C \tag{4.3}$$

$$c_n + M(N - n) \geq C \tag{4.4}$$

For an example of the space delimited by these inequalities, refer to Figures 4.2 and 4.3. Notice how a small change in one parameter ($m = 0$ versus $m = 1$) can cause a large change in the controller state space.

When a controller decides at step n to perform s_n rounds in the next clock cycle, for the resulting number c_{n+1} of accepted rounds, Inequalities 4.3 and 4.4 must also hold, so that

$$s_n \leq C - m(N - n - 1) - c_n \tag{4.5}$$

$$s_n \geq C - M(N - n - 1) - c_n \tag{4.6}$$

These are minimal correctness ensuring requirements. A simple controller may not utilize the entire space delimited by Inequalities 4.1 thru 4.4. A more sophisticated controller

can react before Inequalities 4.3 and 4.4 apply and only modify the probabilities of future round counts to ensure better randomness of the process.

To continue the PRESENT example, $M = 3$ rounds are computed at each clock cycle. The output from a randomly chosen round (1 to 3) is stored in the output register. The round controller keeps the count of already evaluated rounds and clock cycles, so that for every encryption/decryption, the total clock cycles is 16 (in average, 2 rounds per clock cycle are evaluated). The controller must ensure that the number c_n of rounds accepted up to the step n remains in the permissible points in Figure 4.2.

With the architectural parameters chosen for the example (3 rounds per clock cycle, 2 rounds on average used in every of 16 clock cycles), there are 5 196 627 ways to evaluate 32 rounds total in 16 clock cycles, while constraint of at least one and at most three rounds is respected. The number of possible combinations has been counted empirically, as it is the count of possible sequences of 16 integers from one to three, where their sum is 32. However, a DPA attack usually targets the first or the last cycle (round) so not all number sequences are different from the attacker's point of view.

4.2 Dummy Rounds Scheme Analysis and Optimization

In previous section, we describe the Dummy Rounds scheme to make hardware implementations of Feistel Networks [29] and Substitution-Permutation Networks [86] more resistant against Side-channel attacks (SCA) such as Differential Power Analysis (DPA) [43, 1] and others non-profiled SCA attacks described in Section 2.1.1.

The Dummy Rounds scheme employs the fact that the cipher networks consists of similar *rounds*. It further assumes that the implementing hardware can execute $M > 1$ rounds in a clock cycle.

In each clock cycle, all the M rounds are cascaded. The controller chooses a random number $\mu, m \leq \mu \leq M$, where the minimum m is another architectural constant. The result of the first μ rounds is used as the result of that clock cycle. These rounds are the *active* rounds. The results from the rest of the rounds (the *redundant* rounds) are discarded, see Figure 4.1.

The randomness of the execution is supposed to hide the real computation from an attacker. To prevent redundant rounds from leaking data, they process random data rather than the real data from preceding rounds.

The choice of μ is limited in certain states of the algorithm. It may need to execute all active rounds in a given number of clock cycles, or there can be lack of unexecuted active rounds with respect to the minimum m.

In this analysis, we follow all the equations introduced above in section 4.1.2. An example of the state space resulting from $m = 1$, $M = 3$, $C = 32$, $N = 16$ is in Figure 4.2. This is the state space of the tested PRESENT implementation. We can see why clock cycle 1 is a problem. Due to $m = 1$, the first clock cycle *must* execute the first round as active, and the last clock cycle *must* execute the 32nd round as active. In the case of PRESENT, those are the rounds that leak most information [108].

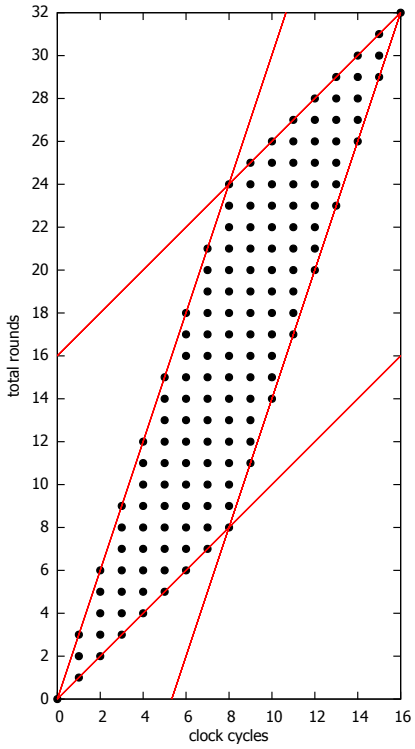


Figure 4.2: A state space for $m = 1$, $M = 3$, $C = 32$, $N = 16$. The lines represent Equations 4.1 to 4.4.

The states of the algorithm, together with transition probabilities, form a Markov chain. Using the state probabilities, we can calculate the probability that the round r was executed as active in a given state. In general, these probabilities vary with clock cycle number for any given round number. The clock cycle with the maximum round execution probability offers the best point for an attack on the given round. The gist of this contribution is to *design* the transition probabilities so that the probability of round execution remains the minimum possible over the entire computation.

4.2.1 Architectural Parameters

The problem with the first and last rounds follows directly from the fact that $m > 0$. There is no freedom and no randomness in the first and last clock cycle. Therefore, we have to fix $m = 0$ in all cases. The modified state space is in Figure 4.3.

As a remedy to the leak in clock cycle 0, the original proposal suggest to randomly postpone the beginning of the computation. This is precisely what can happen with $m = 0$: there can be a random number of redundant rounds at the beginning, and then some active rounds can occur. Therefore, any scheme with $m = 0$ fulfills this request as a special case.

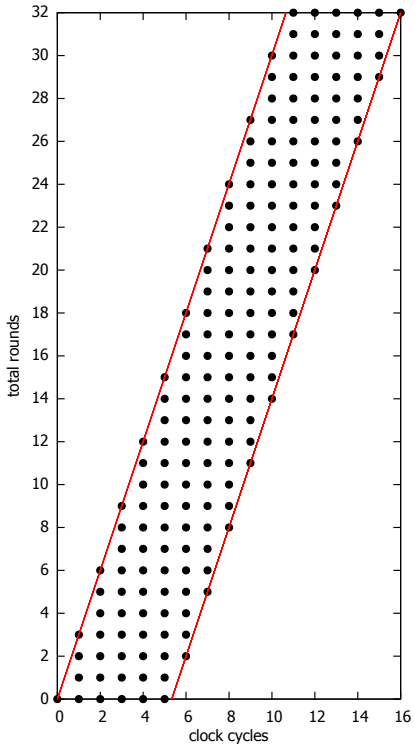


Figure 4.3: A state space for $m = 0$, $M = 3$, $C = 32$, $N = 16$. The lines represent Equations 4.1 and 4.3.

4.2.2 A Slot-Level Model and Round Control

In the above mentioned Markov chain, the transitions have a regular structure. Let $S_{n,r}$ be the state that has executed rounds $1 \dots r$ in the clock cycle n. From this state, transitions to states $S_{n+1,r+m}, \dots S_{n+1,r+M}$ are possible.

In the original proposal, M rounds are executed serially, and a random output is chosen. We have to suppose that the attacker can distinguish the execution of a particular round. Then, instead of N clock cycles, we model $K = MN + 1$ *slots*. Then, a yet simpler (but larger) model can be constructed.

Let $S_{k,r}$ be the state that has executed rounds $1 \dots r$ in the slot k. From this state, only two transitions are possible. Either, the next round will be taken as active, which leads to the state $S_{k+1,r+1}$. Or, the round is redundant, which transits to the state $S_{k+1,r}$. An example is in Figure 4.4.

This model is more general than the round control in the original proposal. That controller takes $m \dots M$ active rounds, and the rest is discarded, so that only thick lines in Figure 4.4 can be followed. Practically at no hardware cost, we can obtain finer control, more random operation and simpler analysis.

The model is still a Markov chain, thanks to $m = 0$. Without this restriction, it would lose the Markov property.

We made the following formal step to simplify the expressions describing the model

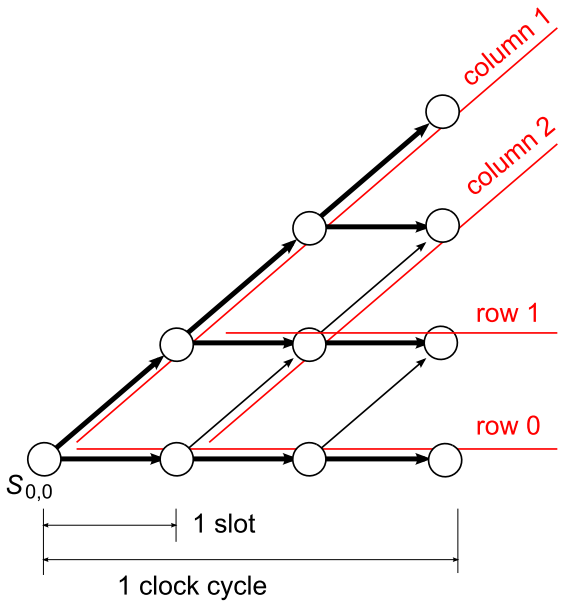


Figure 4.4: A part of a slot-level model with $m = 0$, $M = 3$

(esp. their indices). With $m = 0$, the accessible part of the state space is always a rhomboid. Therefore, we use an alternate coordinate system of rows and columns along the sides of the rhomboid (Figure 4.4). As in the previous model, the row r corresponds to the completed sequence $1 \dots r$.

The width W of the rhomboid is

$$W = MN - C - 1 \tag{4.7}$$

Whereas the architectural parameter M expresses the overhead in hardware, W captures the overall relative overhead in work effort, and, thus, in energy consumption. The unprotected computation has, of course, $M = 1$ and $W = 1$.

4.2.3 Probabilities Analysis

In the above described model, let

- $s_{c,r}$ be the probability of the state $S_{c,r}$ in column c and row r,
- $p_{c,r}$ be the probability, that the next round will be active in the state $S_{c,r}$,
- $\rho_{c,r}$ be the probability, that the model will arrive at $S_{c,r}$ by executing the round r.

Then, the correctness of the computation requires that

$$p(W, r) = 1, r = 0 \dots C - 1 \tag{4.8}$$

and

$$p(c, C) = 0, c = 1 \dots W \tag{4.9}$$

The initial state has probability 1, that is,

$$s_{1,0} = 1 \tag{4.10}$$

For chosen probabilities $p_{c,r}, c = 1 \dots W-1, r = 0 \dots C-1$, state and round execution probabilities can be calculated as

$$s_{c,0} = s_{c-1,0}(1 - p_{c-1}), c = 1 \dots W \tag{4.11}$$

$$s_{c,r} = s_{c,r-1}p_{c,r-1} + s_{c-1,0}(1 - p_{c-1}), c = 1 \dots W, r = 1 \dots C \tag{4.12}$$

$$\rho_{c,r} = s_{c,r-1}p_{c,r-1}, c = 1 \dots W, r = 1 \dots C \tag{4.13}$$

The calculation proceeds from bottom row up, and within a row, from left to right. Refer also to Figures 4.5 and 4.6.

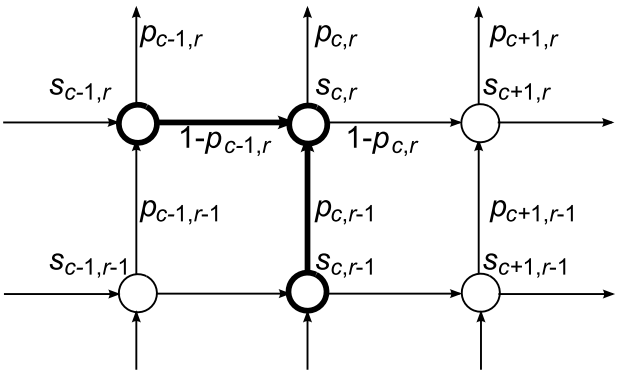


Figure 4.5: State probability derivation in a slot-level model

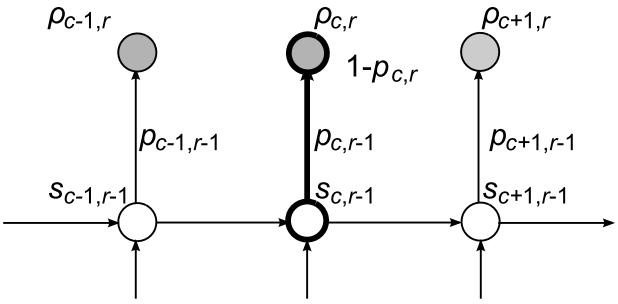


Figure 4.6: Round execution probability in a slot-level model

4.2.4 Probability Design

Let us consider an attack to round r. The weakest point for this attack is the slot $c + r$ (in the original coordinates) where the round probability $\rho_{c,r}$ is maximum. Yet, the round r must be executed at some time, that is

$$\sum_{c=1}^{W} \rho_{c,r} = 1, r = 1 \dots C \tag{4.14}$$

For optimum protection, we thus require

$$\rho_{c,r} \stackrel{!}{=} 1/W, c = 1 \dots W, r = 1 \dots C \tag{4.15}$$

Using Equation 4.13, we obtain

$$p_{c,r} \stackrel{!}{=} 1/W s_{c,r-1}, c = 1 \dots W, r = 1 \dots C \tag{4.16}$$

Combining Equations 4.11, 4.12 and 4.16, we again define a calculation that proceeds bottom-up and left-to-right and gives the optimum transition probabilities together with state probabilities as a by-product. Notice that there is no choice in the process, that is, all the transition probabilities follow from the requirement in Equation 4.15 and there is only one solution.

There are explicit formulas for the transition probabilities. It can be proven (by a rather tedious proof) that the above recurrent computation gives

$$p_{c,0} = \frac{1}{W - c + 1}, c = 1 \dots W \tag{4.17}$$

$$p_{c,r} = 1, c = 1 \dots W, r = 1 \dots C - 1 \tag{4.18}$$

This means that the optimum protection executes a number of redundant rounds first, given by transition probabilities in Equation 4.17. Then, it executes all rounds as active, and finally executes redundant rounds to the required number of slots. Refer also to Figure 4.7.

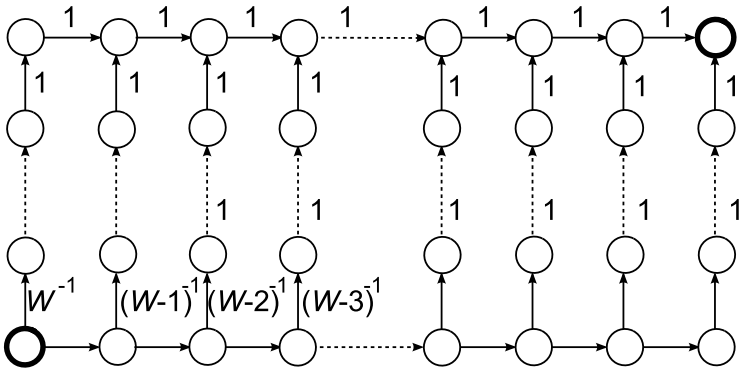


Figure 4.7: Optimum trajectories in a slot-level model

4.2.5 Analysis Results

Section 4.2.4 proves that there is always an optimum solution which satisfies Equation 4.15. An attack to any round must collect more traces to achieve certain probability, that the

desired round has been executed with a given probability in the collected traces. For hit probability h, the relative increase q in the number of traces is

$$q = \frac{\log(1-h)}{\log(1-1/W)} \tag{4.19}$$

It can be seen that the amount of protection depends on work effort only. The function is, unfortunately, almost linear in the practical range of work effort, see Table 4.1. With an average work effort, around 40 times the number of traces are required to collect compared with the unprotected circuit.

W	q	W	q
2	7	11	49
3	12	12	53
4	17	13	58
5	21	1 4	63
6	26	15	67
7	30	16	72
8	35	17	76
9	40	18	81
10	44	19	86

Table 4.1: Multiples of required traces q as a function of work effort W for $h = 0.99$

4.3 Final Dummy Rounds Scheme Datapath Design

In this section, we describe our implementation of the Dummy Rounds scheme described at first in Section 4.1. In this section, we describe further optimizitations solving problems of the initial implementation and we also implement modifications proposed in analysis of the scheme described in Section 4.2.

4.3.1 Design A

At first, we have reimplemented Dummy Rounds for PRESENT cipher without any proposed optimizations. We named this version as *Design A* and we have done several measurements for this design to compare it with first proposed Dummy Rounds implementation. The leakage assessment scenarios are the same.

4.3.2 Design B

The initial design did not allow a lot of configuration, so we implemented another version of PRESENT with Dummy rounds countermeasure from the ground up. This time the

number of valid rounds in each clock cycle was not determined by the in-circuit generator, but the random values are sent together with plaintext and the key. This modification gives us the possibility to have one bitstream and use it for more (not so much random) scenarios. We have also implemented an option of the first round as dummy one. It that case, the intermediate result is written again in the same register. The design is in Figure 4.8.

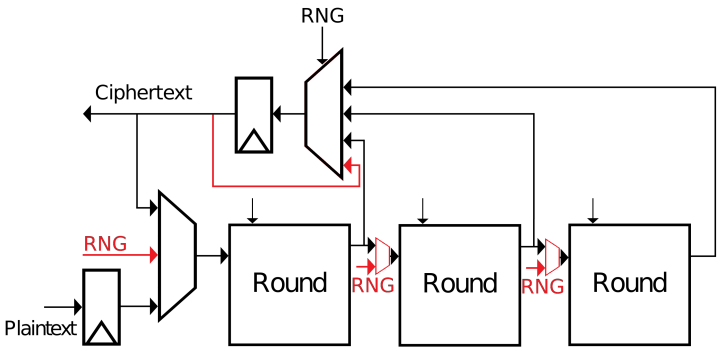


Figure 4.8: Desing B implementing option of firts dummy round.

4.3.3 Design C

As seen from the Table 4.2 later in the results, the usage of *empty cycles*, where all rounds are dummy, worsened the results of the t-test significantly. During these cycles, all rounds process random values and no new intermediate data are available. Therefore the values stored in registers do not change, and power consumption differs significantly in comparison with active cycles, where new computed intermediate value is stored into the registers.

In pursuit of making power consumption more even and less dependent on specific configuration, dummy registers were added into the circuit. These registers were used only during *empty cycles* and a random value is then written in there overwriting another random value. Usage of the dummy register causes a random power consumption similar to overwriting an intermediate value in the *real* register. The design is in Figure 4.9.

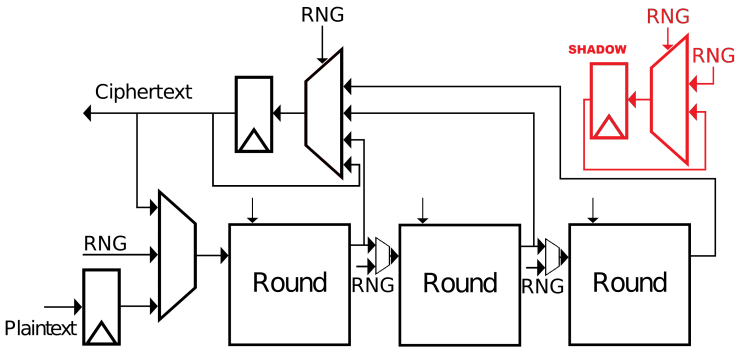


Figure 4.9: Desing C implementing dummy (shadow) register for empty cycles.

4.3.4 Design D

The first implementation of dummy registers was more of an ad-hoc approach than a rigorous solution. To make the most out of added registers, their effect needed to be extended to active rounds without adding any leakage.

Fortunately, the next version satisfied both requirements. Valid and dummy registers became indistinguishable, and their contents switched after each clock cycle. The switching means that new random value will always overwrite valid data and vice versa. An overall number of changes in every register should be completely random, even in cases of multiple consequent empty cycles. Design implementing this countermeasure enhancement is shown in Figure 4.10.

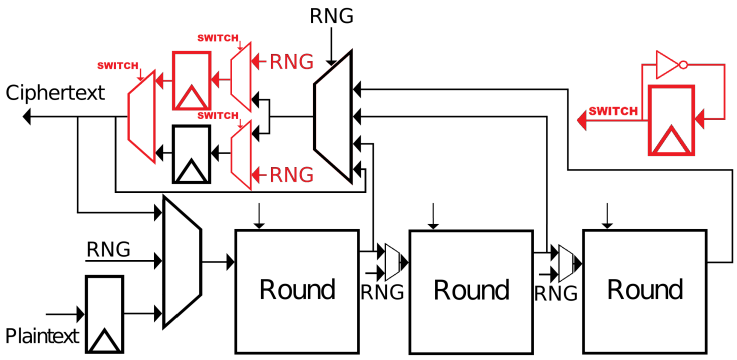


Figure 4.10: Desing D implementing *switching registers*.

4.4 Final Dummy Rounds Scheme Controller Design

The leakage assessment (described in detail in next Section 2.3) of the Dummy Rounds Scheme design described above revealed that potential leakage remains within the first clock cycle of encryption, even if no active rounds are processed during the first cycle. To tackle this problem, we designed a new Dummy Rounds controller described in this section and evaluated together with the datapath in the next section.

4.4.1 Dummy Rounds Controller Modification

The test vector leakage assessment using Welch's t-test [84] described in Section 2.3.1 showed that the leakage of Design D from Section 4.3 has been significantly reduced, compared to first version described in Section 4.1 and reimplemented in previous section as Design A. The maximum leakage is now at the beginning of encryption, see Figure 4.20, reaching maximum t-value of 14.27. As discussed in [A.2], this leakage is caused by the presence of the plaintext in the working registers since the beginning of the encryption, even if multiple empty clock cycles ($\mu_i = 0$) are executed after the start. Similar problem

happens by the end of the encryption, when the working register holds the ciphertext until the last clock cycle even if the execution is filled with series of empty clock cycles.

To tackle this problem, we modified the controller of Design D to fill the working registers with random data after the start of each encryption. The registers are loaded with the plaintext and the key just before these data are needed, i.e., just before the beginning of the first non-empty clock cycle. In the end of the encryption, once the correct ciphertext is computed and processed by the other parts of the design (e.g. the communication protocol part), it is overwritten with random data.

4.5 Dummy Rounds Scheme Leakage Assessment

4.5.1 Measurement Setup

We implemented the Design D of PRESENT cipher [12] with both the original and enhanced controller. The design was implemented in SAKURA-G board [37], which is equipped with Xilinx Spartan 6 FPGA. The design implements $M = 3$ rounds, with at least $m = 0$ rounds to be executed in every clock cycle throughout the course of $N_{max} = 35$ clock cycles for the entire encryption. The design is clocked at 1.5 MHz.

All versions of design have been implemented and evaluated on the SAKURA-G board [37]. Our first goal was to directly compare our newly implemented *Design A* with initial Dummy rounds implementation results. For that reason, only 100 000 power traces were measured during these scenarios.

Since *Design B* further all the designs implement $M = 3$ rounds, with at least $m = 0$ rounds to be executed in every clock cycle throughout the course of $N_{max} = 35$ clock cycles for the entire encryption. The design is clocked at 1.5 MHz. We raised the amount of measured power traces raised to 1 000 000 according to [84] in the rest of the scenarios, where our new implementation, including all new latest enhancements, is evaluated.

The traces were collected by PicoScope 6404D oscilloscope [95] at the sampling frequency of 312 MS/s. Hence, every clock cycle is covered by 208 samples. SICAK toolkit [89] controlled the measurement—it communicated with the design, collected the power traces from the oscilloscope, and evaluated them with the Welch's t-test. To provide support for our version of PRESENT cipher specialized measurement plug-in was also developed. Using this plug-in is possible to easily create configurations of (pseudo)random runs of the encryptions through the (pseudo)random numbers sent together with key and plaintext.

4.5.2 Results

We have evaluated several designs, where Designs A, B, C and D are described in the Datapath Section 4.3 and implement corresponding optimizations. Scenario E uses Design C (without *switching registers*), scenario F then full Design D. Both of them also uses enhanced master controller with better random values usage and better work with plaintext on the input of encryption.

Table 4.2: Measurement scenarios

Design	Setup	Max. t-value
A.01	1 round per cycle, 32 cycles	142.32
A.02	2 rounds per cycle, 16 cycles	288.96
A.03	8x3 + 8x1 rounds per cycle, 16 cycles	353.03
A.04	alternating 3 and 1 rounds per cycle, 16 cycles	242.87
A.06	random 1 to 3 rounds per cycle, 16 cycles	19.98
B.09	random cycles	60.31
B.10	random cycles, first clock cycle not empty	47.99
B.11	random cycles, first clock cycle empty	1291.42
C.12	random cycles, first clock cycle empty	19.16
D.13	random cycles	14.27
E.18	random cycles without switching registers	11.83
F.19	random cycles with switching registers	8.63

There is a list of designs summarizing their differences:

- **Design A** - This design implements countermeasures used in [A.1]. For this design all the scenarios are measured with only 100 000 power traces, because of its comparability with initial Dummy rounds version results.
- **Design B** - Design has the same countermeasures, but there is no PRNG in the design. The random values are transmitted into the FPGA board together with key and plaintext. We have still used 64-bit linear feedback shift register for our experimental evaluation.
- **Design C** - Design C in comparison with design B includes *dummy registers*. Dummy registers are used as intermediate value storage in case of no active round in that clock cycle.
- **Design D** - In comparison with design C, where the value in only one set of intermediate values is changed, in design D are changed values in all the registers in every clock cycle. Current valid data are stored into registers containing a random value and vice versa, which we further call as *switching registers*.
- **Desings/Scenarios E and F** - These scenarios were measured on a design including an enhanced master controller described in Controller Section 4.4.

Here you can see the table of used designs and scenarios and their maximal t-values and also graphs with measures t-values in time, where vertical lines show edges of clock cycles.

Design A gives better results for the strictly random scenario. The maximal t-value 19.98 is still approximately four times bigger than the allowed threshold according to [84]

with only 100 000 measured power traces. There is still the most significant problem after the first clock cycle, as it has been proposed and discussed in [A.2]. The result of random Dummy Rounds scenario is visible in Figure 4.15.

For Design B, the maximal t-values are bigger than for scenario A.06. However, it is also because of 1 000 000 measured power traces per scenario. It can be seen as a paradox, but the best result has the scenario with some active rounds during the first clock cycle. There is enormous t-value of 1291.42 when there is no active round in the first clock cycle. This is because no change in the register after the first cycle makes the switching activity in two first cycles wholly dependent on used plaintext.

This problem is well solved with *dummy (shadow) register* in Design C. The modification gives result t-value 19.16, which is the best to this point. The t-values of this scenario are in Figure 4.19. With Design D, the *switching registers* modification gives even better result with maximal t-value of 14.27 in Figure 4.20.

With the controller modification, scenario F.19 gives the best result ever, with 8.63 getting close to threshold 4.5. It is important to say that this is a better result than the results of all countermeasures used in [82] without combining them. The graphs with the results of the last two scenarios are in Figures 4.21 and 4.22. The modification of the Dummy Rounds controller described in Section 4.4 and evaluated in this section, combined with internal modifications from [A.4] described in Section 4.3 (Design D), makes the Dummy Rounds even more competitive with other SCA hardware-level countermeasures. For example, countermeasures proposed in [82] (S-box decomposition, register precharge, masking) provided sufficient security level only if all of them were employed to protect the design together. Sole countermeasures exceeded the level of 4.5 several times.

Considering other aspects, the Dummy Rounds scheme offers an implicit trade-off between security and overhead. In the setup used in this case study, the area overhead is roughly 200 % (3 rounds blocks instead of just 1) compared to countermeasures proposed in [82], where area overhead of logic (LUTs) is roughly 400 % (most of them occupied by decomposed and masked S-boxes).

4.6 Conclusion

This chapter describes our Dummy Rounds hardware SCA countermeasure scheme, which combines software hiding in time, shuffling and common hardware hiding of the circuitry power consumption. All our contributions to Dummy Rounds scheme were published in [A.1, A.2, A.4, A.5]. The main idea of the scheme is that there are more parts of hardware design (typically rounds of iterative cryptographic algorithms) that are all executed. Still, their outputs are randomly used or not used for computation in every clock cycle. So, the design structure is the same for every clock cycle, and power consumption stays the same. While the computation output in a single clock cycle changes partially randomly, the final result stays correct due to round scheduling.

This chapter includes an analysis of the scheme and optimizations of the scheme datapath and controller, resolving from the analysis. Compared with other countermeasures

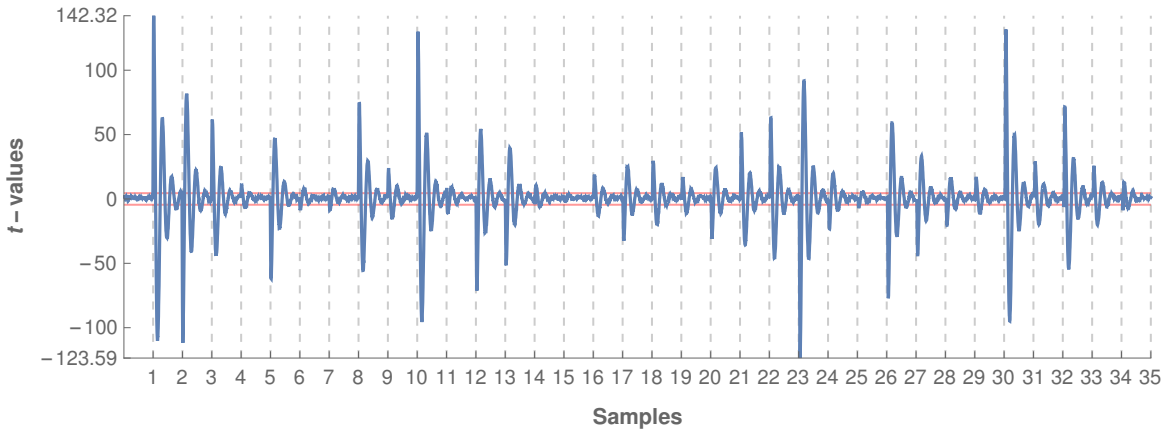


Figure 4.11: Scenario A.01 t-values.

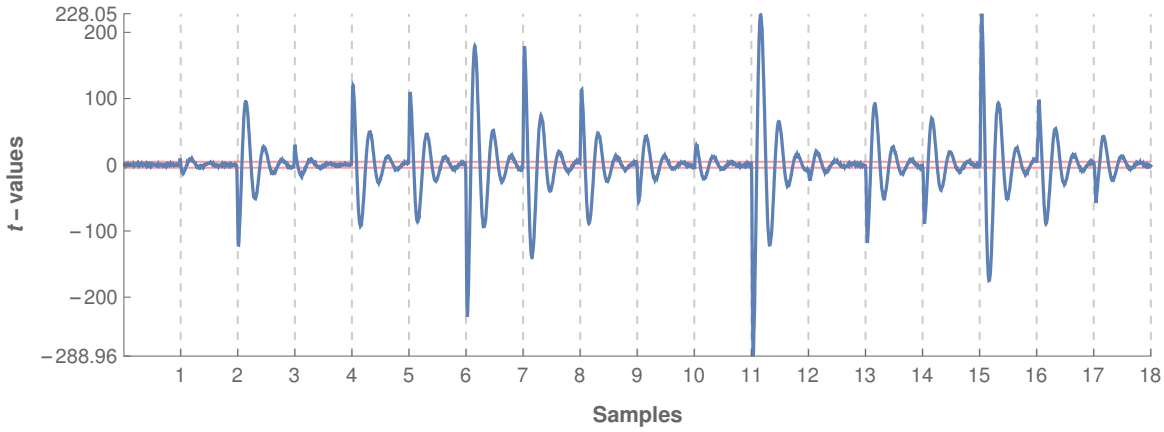


Figure 4.12: Scenario A.02 t-values.

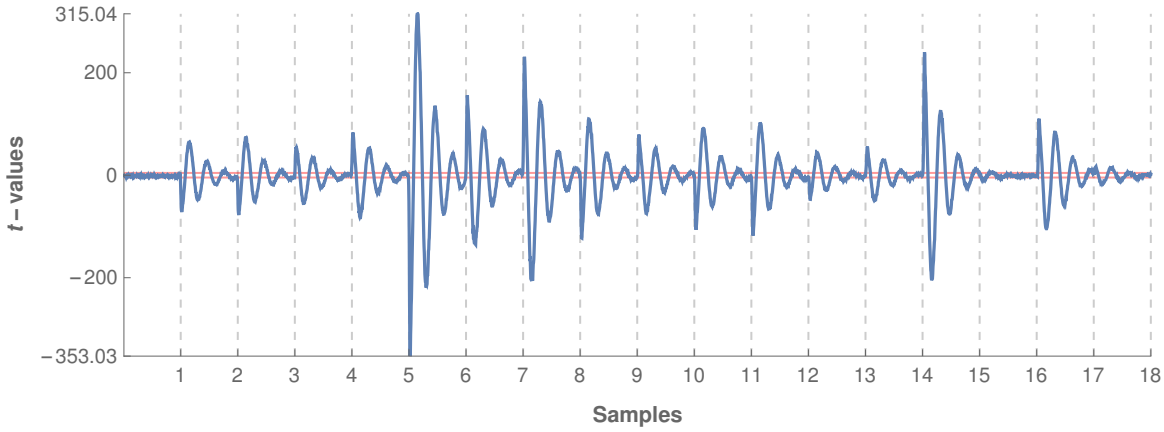


Figure 4.13: Scenario A.03 t-values.

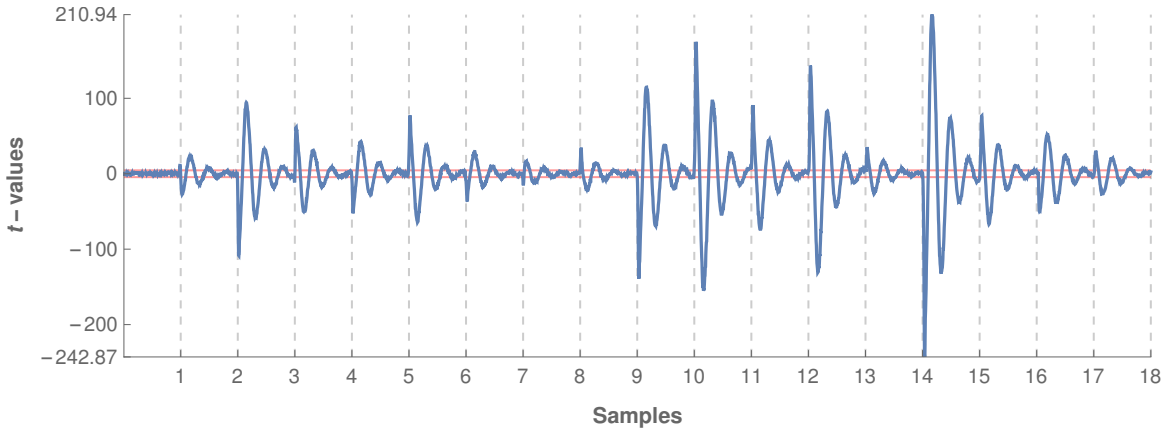


Figure 4.14: Scenario A.04 t-values.

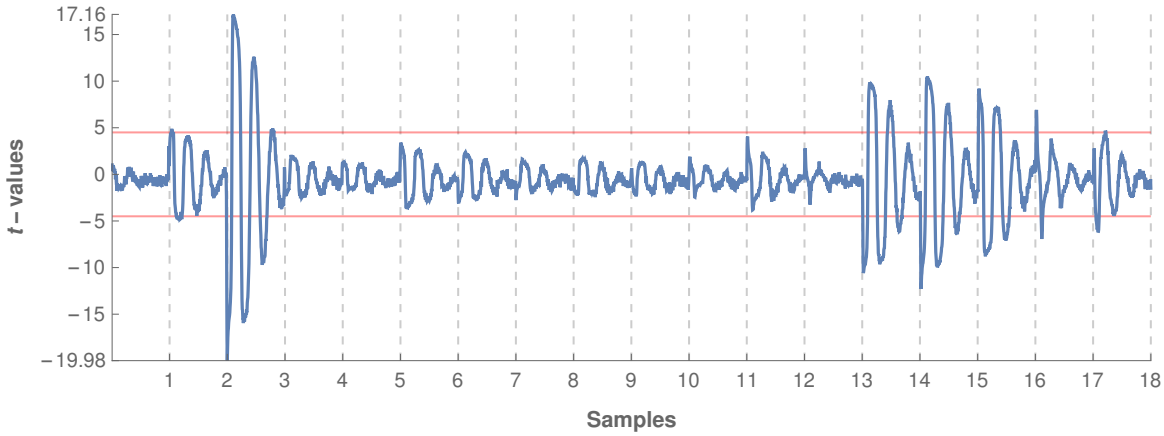


Figure 4.15: Scenario A.06 t-values.

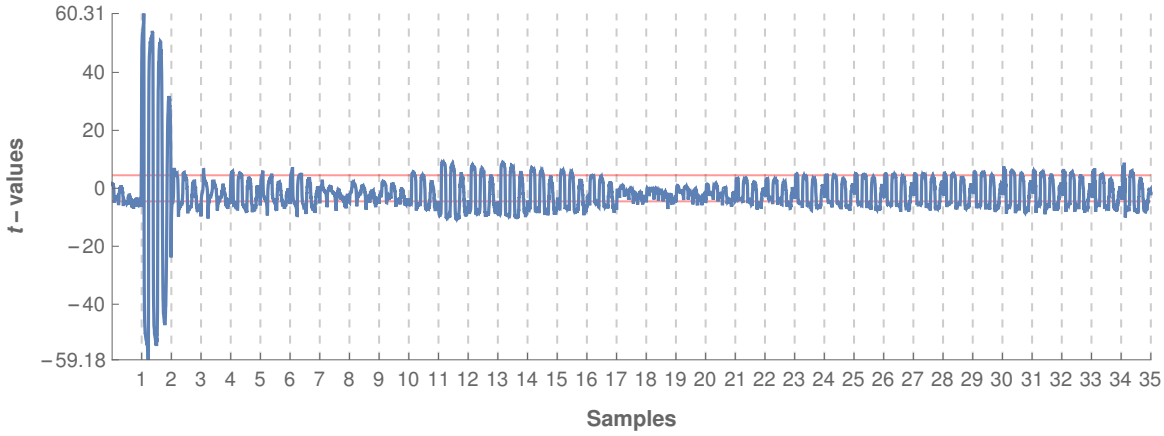


Figure 4.16: Scenario B.09 t-values.

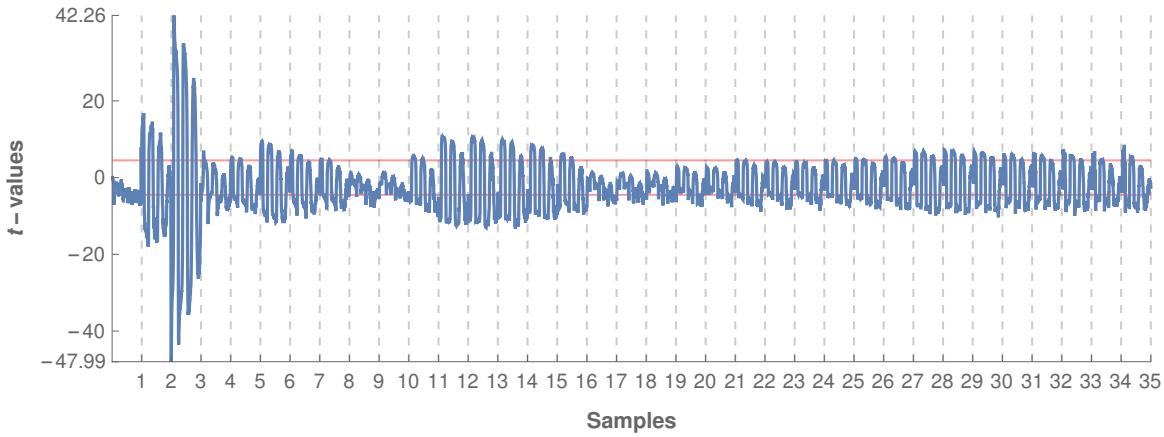


Figure 4.17: Scenario B.10 t-values.

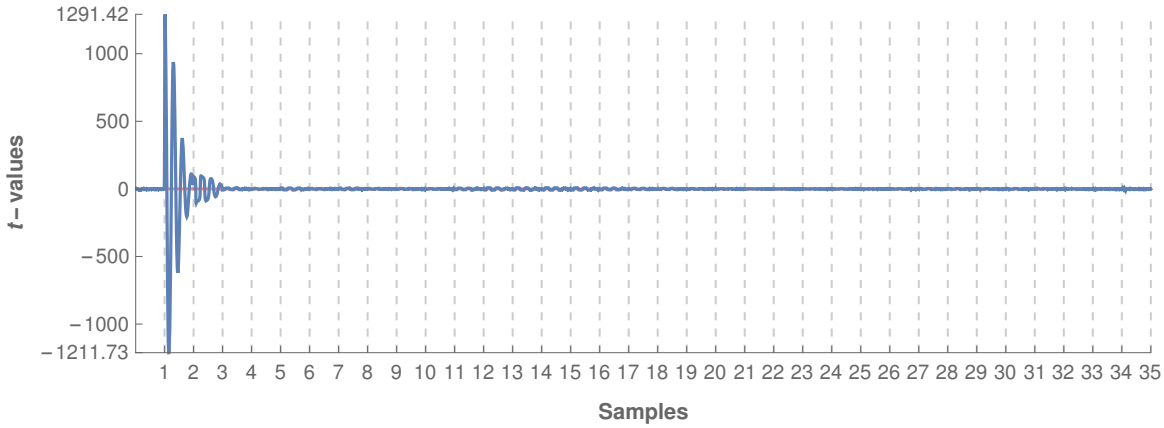


Figure 4.18: Scenario B.11 t-values.

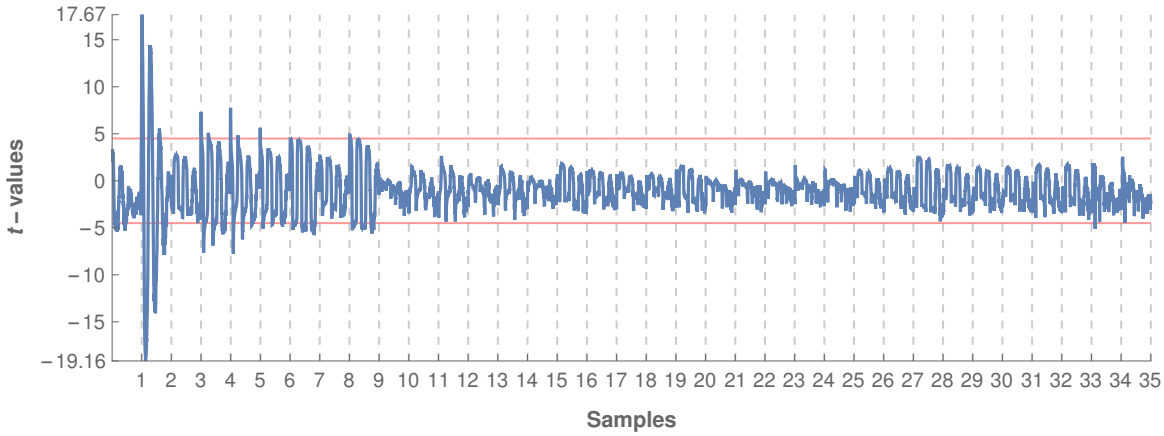


Figure 4.19: Scenario C.12 t-values.

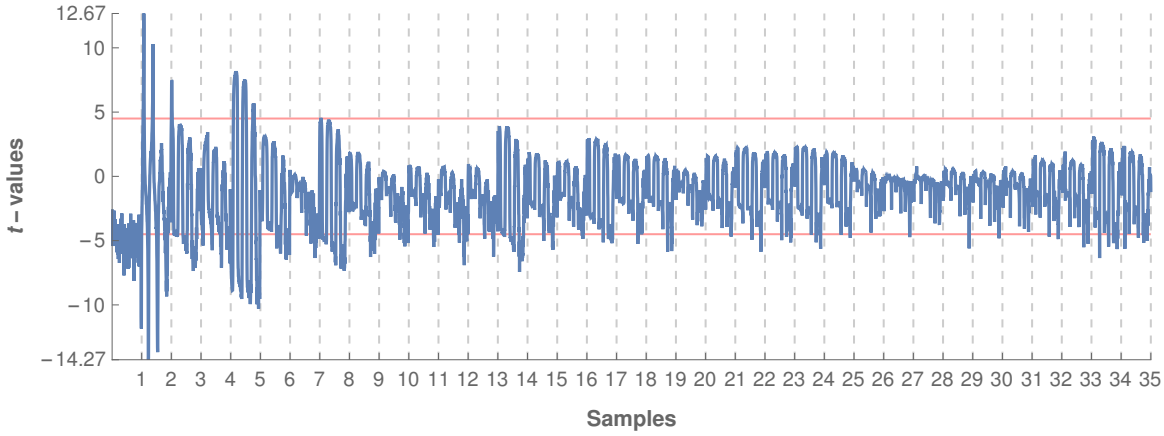


Figure 4.20: Scenario D.13 t-values.

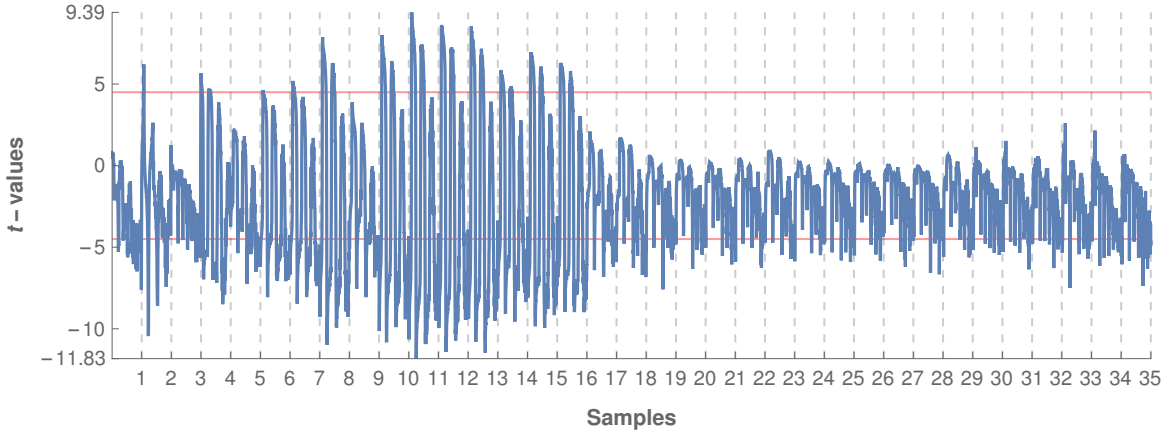


Figure 4.21: Scenario E.18 t-values.

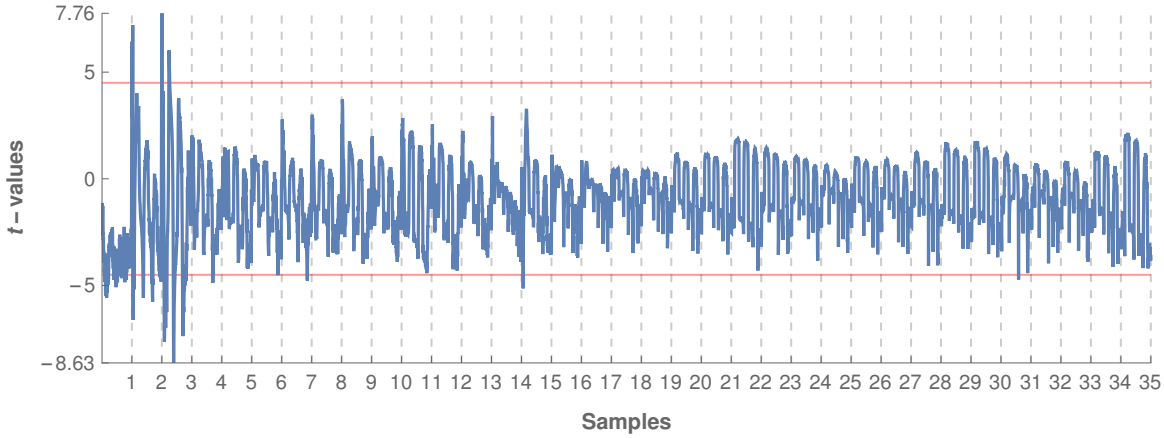


Figure 4.22: Scenario F.19 (with best results) t-values.

described in Section 2.2.3, our Dummy Rounds scheme has several advantages and disadvantages. The Dummy Rounds performance is competitive with countermeasures proposed in [82]. If used alone, none of the countermeasures in [82] fulfils the requirement of the t-test value threshold. The requirement is fulfilled when all the proposed countermeasures are combined. Our Dummy Rounds scheme also does not fulfil the threshold, but its results are better than the other countermeasures used alone. However, our final Dummy Rounds scheme includes Shadow register, which is in its principle the same as Register precharge proposed in [82].

The Dummy Rounds scheme is straightforward to implement. The round part of the design is copied without any modifications; only the controller has to be modified. The Dummy Rounds scheme also offers an implicit trade-off between security and area overhead, which is given by a number of the round block copies (it is always $M = 3$ in our experimentally evaluated case study).

On the other hand, the Dummy Rounds is not provably secure contrary to, e.g. Threshold Implementation [67], Domain Oriented Masking [35], and other similar schemes. However, compared to masking schemes used with more complex cryptographic algorithms such as AES, the area overhead of Dummy Rounds can be much lower. Considering still $M = 3$, the area overhead of Dummy Rounds is still about 200 % for any round-based cipher. Contrary to, e.g. Threshold Implementation implemented for AES, even if the state-of-the-art implementation proposed by Moradi et al. [63], where the area overhead is approx. 360 %.

Chapter 5

Conclusions

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

In Chapter 2, we present the theoretical background and state of the art of the topic. We introduce side-channel analysis, side-channel attacks, and countermeasures, primarily focusing on countermeasures against these attacks used in FPGA implementations. Although our research is based on hardware approaches, we also introduce some countermeasures and their typical software implementations, which inspired our research.

Our implementations of AES and Serpent secured by countermeasures proposed earlier for PRESENT cipher are presented in Chapter 3. We have measured no leakage using first-order non-specific Welch's t-test for these implementations. Further Socha et al. in [88] presented leakage assessment of our implementations with second-order t-test, where only apparent leakage from Serpent encryption was detected. Both implementations further resisted second-order DPA/CPA attacks.

In Chapter 4, we present our hardware Dummy Rounds countermeasure scheme. The chapter starts with describing the principle and where the inspiration comes from. Analysis of the scheme resulting in a proposal of modifications follows, and these proposed and some more modifications are presented further. Finally, the leakage assessment of our implementations using Welch's t-test and a discussion of the results are presented.

5.1 Summary

The implementations of two more complex cryptographic algorithms, specifically secured by previously proposed countermeasures, were proposed. Both implementations were evaluated as first-order leakage-free. These implementations were in later work of the community evaluated as second-order secured. There was an apparent leakage detected in the leakage assessment; however, both implementations then resisted second-order DPA/CPA attacks.

Our Dummy Rounds scheme is a hardware SCA countermeasure, which is not provable secure. Still, it is straightforward to implement and offers an implicit trade-off between security and overhead. The final version of the scheme, with all the modifications coming

from the analysis of the scheme, is competitive with other non-provable countermeasures used in hardware, especially FPGAs, offering still easier implementation and lower overhead, especially with more complex cryptographic algorithms.

All results were presented and discussed in the scientific community and published in proceedings of 5 international conferences, most of papers were cited in WoS and/or Scopus databases.

5.2 Contributions of the Dissertation Thesis

1. Second-order DPA and CPA attacks resistance implementations of AES and Serpent: This work describes an implementation of AES and Serpent secured with previously proposed countermeasures implemented for PRESENT. This work evaluates the implementations as leakage-free using the non-specific univariate first-order Welch's t-test. In later work, these implementations resisted second-order DPA and CPA attacks.

2. Hardware Dummy Rounds countermeasure scheme: We present our proposed and continually evolved Dummy Rounds countermeasure. Its results are competitive with some other solo-used previously proposed countermeasures. The advantages of our Dummy Rounds are a straightforward approach to implementing the scheme and an implicit trade-off between security and area overhead.

3. Dummy Rounds linear overhead regardless of an implemented algorithm: Although Dummy Rounds has high area overhead in comparison with other countermeasures considering lightweight ciphers (PRESENT is the case study), it has good area overhead results in comparison with the same countermeasures considering more complex ciphers.

4. Bringing attention to the control part of the circuit, as opposed to the usual focus on side channels of the datapath, optimization of the control algorithm and safe controller design.

5.3 Future Work

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

- It would be interesting to implement more complex cryptographic algorithms such as AES or Serpent with Dummy Rounds scheme. It could offer an interesting trade-off between its security and implementation results, especially compared to a provable secure implementation, such as threshold implementation.

- Consider using the Dummy Rounds scheme as a combined countermeasure against side-channel and fault analyses. The redundancy in dummy rounds (not used for the result in that clock cycle) could be used.

Bibliography

[1] Mehdi-Laurent Akkar, Régis Bevan, Paul Dischamp, and Didier Moyart. Power analysis, what is now possible... In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, pages 489–502, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[2] Jean-Claude Bajard, Laurent Imbert, Pierre-Yvan Liardet, and Yannick Teglia. Leak resistant arithmetic. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, pages 62–75, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[3] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. pages 64–81, 11 2014. doi:10.1007/978-3-319-16763-3_5.

[4] Timo Bartkewitz and Kerstin Lemke-Rust. Efficient template attacks based on probabilistic multi-class support vector machines. In *Smart Card Research and Advanced Applications: 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers 11*, pages 263–276. Springer, 2013.

[5] Lejla Batina, Benedikt Gierlichs, and Kerstin Lemke-Rust. Comparative evaluation of rank correlation based dpa on an aes prototype chip. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *Information Security*, pages 341–354, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-85886-7_24.

[6] A. Battistello and C. Giraud. Fault analysis of infective aes computations. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 101–107, Aug 2013. doi:10.1109/FDTC.2013.12.

[7] Ali Galip Bayrak, Nikola Velickovic, Paolo Ienne, and Wayne Burleson. An architecture-independent instruction shuffler to protect against side-channel attacks. *ACM Trans. Archit. Code Optim.*, 8(4), jan 2012. doi:10.1145/2086696.2086699.

[8] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.

[9] Eli Biham, Ross Anderson, and Lars Knudsen. Serpent: A new block cipher proposal. In *International Workshop on Fast Software Encryption*, pages 222–238. Springer, 1998.

[10] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20*, pages 326–343. Springer, 2014.

[11] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Trade-offs for threshold implementations illustrated on aes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.

[12] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 450–466, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[13] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer, 2004.

[14] Frank Markham Brown. *Boolean reasoning: the logic of Boolean equations*. Springer Science & Business Media, 2012.

[15] M. Bucci, R. Luzzi, M. Guglielmo, and A. Trifiletti. A countermeasure against differential power analysis based on random delay insertion. In *2005 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3547–3550 Vol. 4, 2005. doi:10.1109/ISCAS.2005.1465395.

[16] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 45–68, Cham, 2017. Springer International Publishing.

[17] Cecile Canovas and Jessy Clediere. What do s-boxes say in differential side channel attacks? Cryptology ePrint Archive, Paper 2005/311, 2005. https://eprint.iacr.org/2005/311. URL: https://eprint.iacr.org/2005/311.

[18] Suresh Chari, Charanjit S Jutla, Josyula R Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Annual International Cryptology Conference*, pages 398–412. Springer, 1999.

[19] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*, pages 13–28. Springer, 2003.

[20] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. Statistical measurement of information leakage. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 390–404, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[21] Tom Chothia and Apratim Guha. A statistical test for information leaks using continuous mutual information. pages 177–190, 06 2011. doi:10.1109/CSF.2011.19.

[22] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, pages 252–263, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[23] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. volume 2009, pages 156–170, 01 2009.

[24] Frederick Emory Croxton and Dudley Johnstone Cowden. *Applied general statistics*. Prentice-Hall, 1940.

[25] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In *International Conference on Smart Card Research and Advanced Applications*, pages 277–284. Springer, 1998.

[26] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael, 1999.

[27] J. L. Danger, S. Guilley, S. Bhasin, and M. Nassar. Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors. In *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*, pages 1–8, Nov 2009. doi:10.1109/ICSCS.2009.5412599.

[28] Bert den Boer, Kerstin Lemke, and Guntram Wicke. A dpa attack against the modular reduction within a crt implementation of rsa. In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*, pages 228–243. Springer, 2003.

[29] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973.

[30] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. volume 6544, pages 262–280, 08 2010. doi:10.1007/978-3-642-19574-7_18.

[31] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 426–442. Springer, 2008.

[32] Benedikt Gierlichs, Jörn-Marc Schmidt, and Michael Tunstall. Infective computation and dummy rounds: Fault protection for block ciphers without check-before-output. In Alejandro Hevia and Gregory Neven, editors, *Progress in Cryptology – LATINCRYPT 2012*, pages 305–321, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[33] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[34] Jovan Golic and Christophe Tymen. Multiplicative masking and power analysis of aes. volume 2523, pages 198–212, 08 2002. doi:10.1007/3-540-36400-5_16.

[35] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. page 3, 10 2016. ACM Workshop on Theory of Implementation Security, TIS '16 ; Conference date: 24-10-2016. URL: https://www.cosic.esat.kuleuven.be/events/acm-ccs2016/, doi:10.1145/2996366.2996426.

[36] Tim Güneysu and Amir Moradi. Generic side-channel countermeasures for reconfigurable devices. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, pages 33–48, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[37] H. Guntur, J. Ishii, and A. Satoh. Side-channel attack user reference architecture board SAKURA-G. In *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, pages 271–274, Oct 2014. doi:10.1109/GCCE.2014.7031104.

[38] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An aes smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security*, pages 239–252, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[39] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Applications of machine learning techniques in side-channel attacks: a survey. *Journal of Cryptographic Engineering*, 10:135–162, 2020.

[40] Annelie Heuser and Michael Zohner. Intelligent machine homicide. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 249–264, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-29912-4_18.

[41] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293–302, 2011.

[42] Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. In *Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, page 219–235, Berlin, Heidelberg, 2014. Springer-Verlag. doi:10.1007/978-3-319-08302-5_15.

[43] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[44] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, pages 104–113. Springer, 1996.

[45] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[46] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. Power analysis attack: an approach based on machine learning. *International Journal of Applied Cryptography*, 3(2):97–115, 2014.

[47] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *Journal of Cryptographic Engineering*, 8:301–313, 2018.

[48] Itamar Levi, Davide Bellizia, David Bol, and François-Xavier Standaert. Ask less, get more: Side-channel signal hiding, revisited. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(12):4904–4917, 2020. doi:10.1109/TCSI.2020.3005338.

[49] Oleksiy Lisovets, David Knichel, Thorben Moos, and Amir Moradi. Let's take it offline: Boosting brute-force attacks on iphone's user authentication through sca. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):496–519, Jul. 2021. URL: https://tches.iacr.org/index.php/TCHES/article/view/8984, doi:10.46586/tches.v2021.i3.496-519.

[50] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6*, pages 3–26. Springer, 2016.

[51] Stefan Mangard. A simple power-analysis (spa) attack on implementations of the aes key expansion. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology — ICISC 2002*, pages 343–358, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[52] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[53] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked aes hardware implementations. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 157–171. Springer, 2005.

[54] Zdenek Martinasek, Lukas Malina, and Krisztina Trasy. Profiling power analysis attack based on multi-layer perceptron network. *Computational Problems in Science and Engineering*, pages 317–339, 2015.

[55] Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. *Radioengineering*, 22(2):586–594, 2013.

[56] David May, Henk L. Muller, and Nigel P. Smart. Non-deterministic processors. In Vijay Varadharajan and Yi Mu, editors, *Information Security and Privacy*, pages 115–129, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[57] Nele Mentens, Benedikt Gierlichs, and Ingrid Verbauwhede. Power and fault analysis resistance in hardware through dynamic reconfiguration. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, pages 346–362, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[58] Daniel Mesquita, Benoit Badrignans, Lionel Torres, Gilles Sassatelli, Michel Robert, and Fernando Moraes. A cryptographic coarse grain reconfigurable architecture robust against dpa. pages 1–8, 03 2007. doi:10.1109/IPDPS.2007.370380.

[59] Thomas Messerges, Ezzy Dabbish, and Robert Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transaction on Computers*, 51(5):541–552, 01 2002.

[60] Thomas S Messerges. Securing the aes finalists against power analysis attacks. In *International Workshop on Fast Software Encryption*, pages 150–164. Springer, 2000.

[61] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. Glitch-resistant masking revisited: or why proofs in the robust probing model are needed. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):256–292, Feb. 2019. URL: https://tches.iacr.org/index.php/TCHES/article/view/7392, doi:10.13154/tches.v2019.i2.256-292.

[62] Thorben Moos, Felix Wegener, and Amir Moradi. Dl-la: Deep learning leakage assessment: A modern roadmap for sca evaluations. Cryptology ePrint Archive, Paper 2019/505, 2019. https://eprint.iacr.org/2019/505. URL: https://eprint.iacr.org/2019/505.

[63] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of aes. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 69–88, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-20465-4_6.

[64] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. Leakage detection with the x2-test. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):209–237, 2018.

[65] James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the development of the advanced encryption standard (aes). *Journal of Research of the National Institute of Standards and Technology*, 106(3):511, 2001.

[66] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *International conference on information and communications security*, pages 529–545. Springer, 2006.

[67] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *Journal of Cryptology*, 24(2):292–321, Apr 2011. doi:10.1007/s00145-010-9085-7.

[68] Colin O'Flynn and Zhizhang Chen. Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection. *Journal of Cryptographic Engineering*, 5:53–69, 2015.

[69] Sanjay Pant. *Design and Analysis of Power Distribution Networks in VLSI Circuits*. PhD thesis, 2008.

[70] Sikhar Patranabis, Abhishek Chakraborty, and Debdeep Mukhopadhyay. Fault tolerant infective countermeasure for aes. In Rajat Subhra Chakraborty, Peter Schwabe, and Jon Solworth, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 190–209, Cham, 2015. Springer International Publishing.

[71] Sikhar Patranabis and Debdeep Mukhopadhyay. *Infective Countermeasures Against Fault Analysis*, pages 197–211. Springer Singapore, Singapore, 2018. doi:10.1007/978-981-10-1387-4_10.

[72] Bruce B Pedersen. Programmable logic device with improved security, August 28 2012. US Patent 8,255,702.

[73] Thomas Plos. Evaluation of the detached power supply as side-channel analysis countermeasure for passive uhf rfid tags. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, pages 444–458, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[74] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 142–159, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[75] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *Smart Card Programming and Security*, pages 200–210. Springer, 2001.

[76] Christian Rechberger and Elisabeth Oswald. Practical template attacks. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications*, pages 440–456, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/978-3-540-31815-6_35.

[77] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 764–783, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[78] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-order masking and shuffling for software implementations of block ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 171–188, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[79] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978. doi:10.1145/359340.359342.

[80] Sasdrich, Mischke, Moradi, and Güneysu. Side–channel protection by randomizing look-up tables on reconfigurable hardware. COSADE, 2015.

[81] Pascal Sasdrich, Amir Moradi, and Tim Güneysu. Hiding higher-order side-channel leakage. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, pages 131–146, Cham, 2017. Springer International Publishing.

[82] Pascal Sasdrich, Amir Moradi, Oliver Mischke, and Tim Güneysu. Achieving side-channel protection with dynamic logic reconfiguration on modern fpgas. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 130–136. IEEE, 2015.

[83] Falk Schellenberg, Dennis RE Gnad, Amir Moradi, and Mehdi B Tahoori. An inside job: Remote power analysis attacks on fpgas. in 2018 design, automation & test in europe conference & exhibition (date), 2018.

[84] Tobias Schneider and Amir Moradi. Leakage assessment methodology. *Journal of Cryptographic Engineering*, 6(2):85–99, Jun 2016. doi:10.1007/s13389-016-0120-y.

[85] Adi Shamir. Protecting smart cards from passive power analysis with detached power supplies. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, pages 71–77, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[86] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, Oct 1949. doi:10.1002/j.1538-7305.1949.tb00928.x.

[87] Rafael Soares, Ney Calazans, Victor Lomné, Philippe Maurine, Lionel Torres, and Michel Robert. Evaluating the robustness of secure triple track logic through prototyping. In *Proceedings of the 21st Annual Symposium on Integrated Circuits and System Design*, SBCCI '08, page 193–198, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1404371.1404425.

[88] Petr Socha, Jan Brejník, Josep Balasch, Martin Novotný, and Nele Mentens. Side-channel countermeasures utilizing dynamic logic reconfiguration: Protecting aes/rijndael and serpent encryption in hardware. *Microprocessors and Microsystems*, 78:103208, 2020. URL: https://www.sciencedirect.com/science/article/pii/S0141933120303732, doi:10.1016/j.micpro.2020.103208.

[89] Petr Socha, Vojtech Miskovsky, and Martin Novotny. Sicak: An open-source side-channel analysis toolkit. In *8th Workshop on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2019)*, 05 2019.

[90] François-Xavier Standaert. Introduction to side-channel attacks. *Secure integrated circuits and systems*, pages 27–42, 2010.

[91] François-Xavier Standaert, Benedikt Gierlichs, and Ingrid Verbauwhede. Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected cmos devices. In Pil Joong Lee and Jung Hee Cheon, editors, *Information Security and Cryptology – ICISC 2008*, pages 253–267, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-00730-9_16.

[92] François-Xavier Standaert, Berna Ors, and Bart Preneel. Power analysis of an fpga: Implementation of rijndael: Is pipelining a dpa countermeasure? volume 3156, pages 30–44, 01 2004. doi:10.1007/b99451.

[93] Data Encryption Standard et al. Federal information processing standards publication 46. *National Bureau of Standards, US Department of Commerce*, 4, 1977.

[94] Daisuke Suzuki and Minoru Saeki. Security evaluation of DPA countermeasures using dual-rail pre-charge logic style. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 255–269, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[95] Pico Technology. PicoScope®6000 Series. [online]. [rev. 2016], [cited 10. 2. 2020]. URL: https://www.picotech.com/oscilloscope/6000/picoscope-6000-overview.

[96] Stefan Tillich, Christoph Herbst, and Stefan Mangard. Protecting AES software implementations on 32-bit processors against power analysis. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 141–157, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[97] Kris Tiri, David Hwang, Alireza Hodjat, Bo-Cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. Prototype ic with wddl and differential routing – dpa resistance assessment. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 354–365, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[98] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, volume 1, pages 246–251. IEEE, 2004.

[99] Chenyang Tu, Jian Zhou, Neng Gao, Zeyi Liu, Yuan Ma, and Zongbin Liu. Qrl: A high performance quadruple-rail logic for resisting dpa on fpga implementations. In Sihan Qing, Eiji Okamoto, Kwangjo Kim, and Dongmei Liu, editors, *Information and Communications Security*, pages 184–198, Cham, 2016. Springer International Publishing.

[100] Michael Tunstall and Olivier Benoit. Efficient use of random delays in embedded software. In Damien Sauveron, Konstantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, pages 27–38, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[101] Harshal Tupsamudre, Shikha Bisht, and Debdeep Mukhopadhyay. Destroying fault invariant with randomization. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, pages 93–111, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[102] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 740–757, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[103] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual information analysis: How, when and why? In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 429–443, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-04138-9_30.

[104] Bernard L Welch. The generalization of student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[105] Carolyn Whitnall, Elisabeth Oswald, and Luke Mather. An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications*, pages 234–251, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-27257-8_15.

[106] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The myth of generic dpa... and the magic of learning. In Josh Benaloh, editor, *Topics in Cryptology – CT-RSA 2014*, pages 183–205, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-04852-9_10.

[107] Xilinx. Spartan-6 libraries guide for hdl designs. URL: https://www.xilinx.com/support/documentation/sw_\protect\penalty\z@manuals/xilinx14_\protect\penalty\z@5/spartan6_\protect\penalty\z@hdl.pdf.

[108] Jing Zhang, Dawu Gu, Zheng Guo, and Lei Zhang. Differential power cryptanalysis attacks against PRESENT implementation. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 6, pages V6–61–V6–65, Aug 2010. doi:10.1109/ICACTE.2010.5579367.

[109] Mark Zhao and G Edward Suh. Fpga-based remote power side-channel attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 229–244. IEEE, 2018.

Reviewed Publications of the Author Relevant to the Thesis

[A.1] Jeřábek, S.; Schmidt, J.; Novotný, M.; Miškovský, V. Dummy Rounds as a DPA countermeasure in hardware. In: *Proceedings of the 21st Euromicro Conference on Digital System Design*. Piscataway, NJ, 2018.

The paper has been cited in:

- Higgins, C.; McDonald, L.; Ijaz Ul Haq, M.; Hakak, S. IoT hardware-based security: A generalized review of threats and countermeasures. *Security and Privacy in the Internet of Things: Architectures, Techniques, and Applications*. 1:261–296, 2021.

[A.2] Jeřábek, S.; Schmidt, J. Analyzing and Optimizing the Dummy Rounds Scheme. In: *Proceedings of the 22nd International Symposium on Design and Diagnostics of Electronic Circuits and Systems*. Piscataway, NJ, 2019.

The paper has been cited in:

- Socha, P.; Miškovský, V.; Novotný, M. High-level synthesis, cryptography, and side-channel countermeasures: A comprehensive evaluation. *MICROPROCESSORS AND MICROSYSTEMS*. 85, 2021.
- Socha, P.; Novotný, M. Towards High-Level Synthesis of Polymorphic Side-Channel Countermeasures. *2020 23RD EUROMICRO CONFERENCE ON DIGITAL SYSTEM DESIGN (DSD 2020)*. 193–199, 2020.

[A.3] Socha, P.; Brejník, J.; Jeřábek, S.; Novotný, M.; Mentens, N. Dynamic Logic Reconfiguration Based Side-Channel Protection of AES and Serpent. In: *Proceedings of the 22nd Euromicro Conference on Digital Systems Design*. Los Alamitos, CA, 2019.

The paper has been cited in:

- Macchetti, M.; Pelletier, H.; Groux, C. A New Fast and Side-Channel Resistant AES Hardware Architecture. *2023 IEEE International Conference on Cyber Security and Resilience (CSR) Proceedings*. 572–579, 2023.

[A.4] Moucha, P.; Jeřábek, S.; Novotný, M. Novel Dummy Rounds Schemes as a DPA Countermeasure in PRESENT Cipher. In: *Proceedings of the 23rd International Symposium on Design and Diagnostics of Electronic Circuits and Systems*. Piscataway, NJ, 2020.

The paper has been cited in:

- Luo, H.; Chen, W.; Ming, X.; Wu, Y. General Differential Fault Attack on PRESENT and GIFT Cipher With Nibble. *IEEE ACCESS*. 9:37697–37706, 2021.
- Soares, R.; Lima, V.; Lellis, R.; Finkenauer, P.; Camargo, V. Hardware countermeasures against power analysis attacks: A survey from past to present. *Journal of Integrated Circuits and Systems*. 16, 2021.

[A.5] Moucha, P.; Jeřábek, S.; Novotný, M. Novel Controller for Dummy Rounds Scheme DPA Countermeasure. In: *Proceedings of the 23rd Euromicro Conference on Digital Systems Design*. Los Alamitos, CA, 2020.

Remaining Publications of the Author Relevant to the Thesis

[A.6] Jeřábek, S.; Schmidt, J.; Novotný, M. *Dynamická rekonfigurace jako opatření proti DPA*. In: *Počítačové architektúry & diagnostika PAD 2017 - Zborník príspevkov*. Bratislava, Slovakia, 2017.

[A.7] Jeřábek, S.; Schmidt, J.; Novotný, M. *Dynamic Reconfiguration as Countermeasure against DPA*. In: *Proceedings of the Work in Progress Session SEAA/DSD 2017*. Linz, Austria, 2017.

[A.8] Jeřábek, S. *Differential power analysis countermeasures in programmable hardware technical research report*. Technical Report, Faculty of Information Technology, Prague, Czech Republic, 2018.

[A.9] Jeřábek, S. *Differential Power Analysis Countermeasures in Programmable Hardware*. Ph.D. Minimum Thesis, Faculty of Information Technology, Prague, Czech Republic, 2018.

[A.10] Jeřábek, S.; Schmidt, J.; Novotný, M. *Dummy Rounds jako opatření proti DPA v hardwaru*. In: *Počítačové architektury & diagnostika PAD 2018*. Pilsen, Czech Republic, 2018.

[A.11] Jeřábek, S. *Analýza Dummy Rounds jako opatření proti DPA v hardwaru*. In: *Sborník příspěvků PAD 2019*. Prague, Czech Republic, 2019.

Remaining Publications of the Author

[A.12] Jeřábek, S.; Buček, J.; Schmidt, J.; Novotný, M. Emulator of Contactless Smart Cards in FPGA. In: *Proceedings of the 6th Mediterranean Conference on Embedded Computing (MECO 2017)*. 2017.