BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

HABILITATION THESIS

Brno, 2020 Ing. ZDENĚK MARTINÁSEK, Ph.D.

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS
ÚSTAV TELEKOMUNIKACÍ

COUNTER MEASURE TECHNIQUES FOR CRYPTOGRAPHIC ALGORITHMS ELIMINATING POWER ANALYSIS ATTACKS

TECHNIKY PROTIOPATŘENÍ KRYPTOGRAFICKÝCH ALGORITMŮ ELIMINUJÍCÍ ÚTOKY PROUDOVOU ANALÝZOU

HABILITATION THESIS
HABILITAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE

Ing. Zdeněk Martinásek, Ph.D.

BRNO 2020

ABSTRACT

This habilitation thesis guides the reader through the power analysis fundamentals including the countermeasure techniques. The text focuses on the practical aspects of the protected implementations of cryptographic algorithms. The results and observations try to support a more frequent realization of implementations utilizing the countermeasures in future. This will be possible, because the readers will understand the underlying concepts of the power analysis, how to protect the implementation and how to evaluate the real leakage of the cryptographic device. In the first part of the thesis, the fundamentals of the power analysis methods are provided. This knowledge introduces the basic building blocks to understand the main principle of the power analysis attacks and countermeasure techniques. In the second part, the power analysis of protected implementations is described from a practical point of view. In the last parts, we analyze which profiled attack has the lowest sensitivity to modifications of the characteristics of leakages. This is the contribution that reflects the real world situation because datasets often suffer from errors. Another contribution is the proposal of the hardware implementation where a hiding technique is utilized.

KEYWORDS

Power Analysis; Countermeasure; Side-Channel Analysis; DPA; DPA Contest; MLP

ABSTRAKT

Tato habilitační práce seznamuje zájemce s podstatou proudové analýzy včetně používaných metod protiopatření. Text práce se zaměřuje na praktické aspekty chráněných implementací kryptografických algoritmů. Dosažené výsledky se snaží podpořit častější budoucí výskyt implementací využívající techniky protiopatření. To bude možné, protože zájemce porozumí principům proudové analýzy, způsobům jak zabezpečit implementaci a ověření konečného vyzařování kryptografického zařízení. V první části práce, jsou představeny základy proudové analýzy. Tyto znalosti představují základní stavební kameny k pochopení používaných technik útoků a protiopatření. V druhé části je realizována proudová analýza chráněných implementací kryptografickýh algoritmů, která je popsána z praktického pohledu útočníka. Poslední část práce analyzuje, který profilující útok je nejméně citlivý na výskyt chyb v proudových průbězích. Tento vlastní přínos reflektuje skutečné situace z praxe. Další vlastní přínos spočívá v návrhu hardwarové implementace symetrického šifrování využívající maskování.

KLÍČOVÁ SLOVA

Proudová analýza; protiopatření; analýza postranním kanálem; DPA; DPA Contest; MLP

MARTINÁSEK, Zdeněk. *Counter measure techniques for cryptographic algorithms eliminating power analysis attacks*. Brno, 2020, 144 p. Habilitation thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications.

DECLARATION

I declare that I have written the Habilitation Thesis titled "Counter measure techniques for cryptographic algorithms eliminating power analysis attacks" independentlyand using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Habilitation Thesis, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENTS

I would first like to thank my family, especially my wife Krisztina, daughter Julie and son Zdeněk. I could not have done it without them and without their support. Second, I would like to thank my dear parents for all their support.

Thanks go out to my colleagues at Brno University of Technology, especially Prof. Kamil Vrba and Assoc. Prof. Václav Zeman for their assistance (academic, scientific, and otherwise) through the course of this work.

Contents

List of Figures

INTRODUCTION

Side-channel attacks analyze physical characteristics of cryptographic devices related to the execution of the implementation of a cryptographic algorithm. The physical analysis aims to extract stored sensitive information such as the secret key. From an industrial point of view, Side-Channel Analysis (SCA) leads to extremely effective and successful attacks against (certified and uncertified) industrial products such as embedded computers, smart cards, tokens or secure cryptographic algorithms such as symmetric AES (Advances Encryption Standard) or asymmetric RSA (Rivest Shamir Adleman). The rationale is that there is a relationship between the manipulated data, the executed operations and the physical properties observed during the execution of the cryptographic device. The physical properties that can be extracted are, for example, the execution time of a cryptographic algorithm, the electromagnetic emanation or the power consumption of the device.

In this habilitation thesis, we focus on side-channel attacks based on the power consumption called Power Analysis (PA) although our observation can be applied similarly to other physical properties. The purpose of this habilitation thesis is to provide description of countermeasure techniques that can be implemented in order to protect cryptographic algorithms against these types of attacks. Moreover, the thesis summarizes the author's results achieved in the field of power analysis attacks that aim to resilient implementations of cryptographic algorithms.

The text is structured in a way that a reader obtains basic theoretical knowledge about power analysis in the first chapter. In the following chapter, the overview of the countermeasure techniques is provided. The most important observations are explained by theoretical and practical examples that are based on unprotected and protected AES implementations on smart cards. It represents the expected pedagogical contribution of this thesis because a reader is able to understand the basic principles of power analysis attacks and the usage of countermeasure techniques. Based on this obtained knowledge, one is naturally able to propose a resilient implementation of any cryptographic algorithm. In the last sections, the main scientific contribution of the thesis is provided. The main contribution lies in the finding which profiled attack has the lowest sensitivity to modifications of the characteristics of leakages and we propose the hardware implementation utilizing the hiding technique in order to prevent power analysis attacks.

1 THESIS OVERVIEW

This chapter provides an overview of the thesis. Firstly, the motivation of the work is presented in Sec. 1.1. Secondly, the thesis objectives and main goals are described in Sec. 1.2. The contribution to current state, the relation to author's publication and pedagogical contribution is described in Sec. 1.3. Finally, the structure of the thesis is outlined in Sec. 1.4.

1.1 Motivation

Nowadays, the provision of security services in modern communication systems is crucial. The number and variety of electronic services utilizing various devices are rapidly growing and the need for better protection is becoming more and more pressing. Therefore, these deployed embedded systems and especially their security play an important role in our everyday life. Some typical applications of these systems are in critical domains like health care, banking sector, traffic management, data-centres, power-grids, etc. Side channel attacks are powerful attacks that employ the side channel leakage of embedded devices such as timing execution or power consumption to obtain sensitive information.

Power analysis measures and analyses the power consumption of cryptographic devices depending on their activity in order to obtain sensitive information, mostly the secret key. The power analysis is very popular with attackers because it is not necessary to possess any special device. Nowadays, power analysis represents an extremely effective and successful type of attacks to break confidential cryptographic algorithms such as AES [1, 113, 46], RSA (Rivest Shamir Adleman) [55, 35], Elliptic-Curve [109, 103], Diffie-Hellman [26] post-quantum schemes[62, 115] and cryptographic devices such as smart cards [24, 78]. Power analysis attacks can be prevented by means of countermeasure techniques that are divided into two basic groups: *masking* [104, 99] and *hiding* [27]. The goal of every countermeasure is to make the power consumption of a cryptographic device independent of intermediate values that are processed during its operation. However, these techniques can also be attacked very easily [83, 113, 78], therefore it is important to pay attention to correct implementations of countermeasures.

The motivation is evident, it is not enough to select a secure cryptographic algorithm in order to fulfill the desired security service, but it is also crucial to realize an implementation that does not leak sensitive information. The thesis describes the fundamentals of the power analysis methods and the countermeasure techniques including the possible attacks targeted on protected implementations from a practical point of view. The motivation is to raise awareness about power analysis and to support realization of protected cryptographic algorithms in future.

1.2 Goals

Based on the above written facts, the main goal of the thesis is to describe the fundamentals of the power analysis methods including the countermeasure techniques. Moreover, we focus on practical aspects of the protected implementation and description of the possible attacks. We want to support realization of protected cryptographic algorithms in future, because the readers will understand the underlying concepts of power analysis, how to protect the implementation and how to evaluate the real leakage of the cryptographic device. We contribute to this field by proposing a protected hardware implementation which is the last important goal of the thesis. The goals of the thesis are summarized below.

- The first goal of the thesis is to provide the basic theory regarding power analysis and countermeasure techniques.
- The second goal of the thesis is to provide the basic theory regarding the attacks targeting the countermeasure techniques.
- The third goal of the thesis is to provide theory regarding the attacks targeting the protected implementation from a practical point of view. All realized analyzes are performed with the help of publicly available power traces, therefore the obtained results can be easily verified by the readers.
- The fourth goal of the thesis is to propose a novel cryptographic algorithm that is protected by hiding. The implementation utilizes the independent operation and the hardware platform with a wide data-path.

1.3 Contribution and Relation to Author's Publications

The thesis is written in a way to have both scientific and pedagogical contribution. Therefore, it should serve the readers who are unfamiliar with the power analysis as well as students to gain fundamental knowledge about the power analysis methods including the countermeasure techniques. The theoretical chapter devoted to the description of the power analysis fundamentals allows the readers to easily understand how these attacks work. A lot of figures and school examples are used in the theoretical chapter in order to complement the text in an appropriate manner. In this way, the readers can observe for example a direct impact of countermeasure techniques on power consumption and, what is more important, on the resulting attack. Readers, who are the future designers of cryptographic systems, can profit from the knowledge obtained in order to design a more secure implementation and evaluate the resilience of the final implementation. Power analysis, has been intensively studied

by the author in the last decade [92, 84, 87, 95, 90, 85, 86, 91, 89, 94, 97, 80]. In the first two introductory subsections (Sec. 2.1 and Sec. 2.2), we build on the knowledge obtained (from above written works) and on the results of the Ph.D. thesis [79]. The main contribution of the Ph.D. thesis was the proposal of the power analysis attack based on a machine learning approach. The countermeasure methods were not touched in the Ph.D. thesis, none of the results presented in this thesis was published in the author's Ph.D. thesis. Naturally, the author's interests moved to the countermeasure methods after successfully dealing with power analysis attacks.

The following theoretical subsections (Sec. 2.3 and 2.4) are based on current results dealing with countermeasure methods [151, 157, 81, 148, 150, 156]. These sections also represent the pedagogical contribution of the thesis and parts of the chapter were used in a university textbook of the Information Security study program at Brno University of Technology, where the author is involved [163, 162]. Moreover, the contents of these sections was also presented by the author at invited lectures, for the Military Research Institute, Brno security meeting and the Smart Cards & Devices Forum.

After these introductory chapters, the text describes outcomes of author's own research on resilience of protected implementations. This chapter contains various results from selected author's publications such as [151, 157, 93, 70, 83]. In the first section (Sec. 3.1), pedagogical contribution is focused on power analysis of protected implementations. In this section, we describe masking and hiding countermeasure techniques including the power analysis from a practical point of view. More precisely, Boolean masking and shuffling of the crucial operations of the AES are attended in our education text as well as a short current state description. As an elementary school example, we bring in to play the DPA Contest because it is world wide known and freely available. Therefore, the reader can verify herself/himself the obtained results which is the best way to understand the explained issues. Parts of the chapter are used in the course System and Device Security of the Information Security study program at Brno University of Technology, where the author is involved.

The following sections (Sec. 3.2, 3.3 and 3.4) present the main scientific contribution of the thesis. The text is a part of the author's papers in the journals with an impact factor, namely, Computers & Security [83], IET Information Security [70] and Radioengineering [93]. In this part, we investigate the security of the improved implementation of AES (V4.2). We used the basic power side-channel techniques, in the same way as a potential adversary would do, and try to analyse the sensitive information. Our analysis, focused on exploiting the first-order leakage, discovered some mistakes. Based on the results, an adversary can launch a standard DPA attack aimed at the S-box output in order to recover the whole secret key. Moreover,

we focus on finding which profiled attack (among conventional profiled attacks and profiled attacks based on machine learning) has the lowest sensitivity to modifications of the characteristics of leakages. This is the contribution that reflects the real world situation because datasets often suffer from various errors or distortions in the measured leakages that may affect the efficiency of the attacks. This variability can occur due to several factors such as human errors, instrument malfunction (due to device ageing), variability across different devices or different acquisition campaigns.

In the last chapter (Chap. 4), we propose the hardware implementation where four parallel AES cores and a 512 bit data-path are utilized in order to protect the implementation. In fact, it is one way how to increase the noise of operations because several independent operations are executed in parallel (hiding technique). This section is the scientific contribution of the thesis and the amended version of the text is a part of the author's paper of the ASHES'18 ACM conference [154]. More details about the realized implementation are provided in the articles [153, 161, 159, 149, 160].

The contribution of the text can be summarized as follows.

- ***Pedagogical contribution***: In the first two introductory sections (Sec. 2.1 and Sec. 2.2), we describe the fundamentals of the power analysis methods. In particular, we explain the principle of profiling power analysis attacks utilizing the standard Gaussian approach, profiling based on machine learning and non-profiling power analysis attacks based on the Correlation coefficient and Difference of Means. This knowledge introduces the basic building blocks to understand the main principle of the power analysis attacks. In order to design a secure implementation, it is crucial to understand the essence of the attack. The following theoretical section (Sec. 2.3) describes the basic countermeasure techniques. However, these techniques can also be attacked very easily in practice, therefore it is crucial to pay attention to the correct implementation of the countermeasure. This issue is addressed in Sec. 2.4 and in Sec. 3.1, where we describe the masking and hiding countermeasure techniques including the power analysis from a practical point of view. More precisely, the Boolean masking and shuffling are attended in our education text as well as a short current state description. As an elementary school example, we bring in to play the DPA Contest because it is world wide known and freely available. Therefore, the reader can verify herself/himself the obtained results. Parts of the chapter were used in a university textbook of the Information Security study program at Brno University of Technology, where the author is involved. Moreover, the contents of the sections was presented by the author at invited lectures for the Military Research Institute and the Smart Cards & Devices Forum.

- ***Scientific contribution***: Chap. 3 and Chap. 4 contain various results from selected author's publications in journals with an impact factor. In this part (Sec. 3.2), we investigate the security of improved protected implementations. Our analysis, focused on exploiting the first-order leakage, discovered some lacks that an adversary can use and realize a standard DPA attack. Moreover, we focus on the finding which profiled attack has the lowest sensitivity to modifications of the characteristics of leakages (3.3 and 3.4). This is the contribution that reflects the real world situation because datasets often suffer from errors or distortions in the measured leakages. In the last chapter (Chap. 4), we propose the hardware implementation where four parallel encryption cores and a 512 bit data-path are utilized in order to protect the implementation. It is one possibility how to increase the noise of operations because several independent operations are executed in parallel. The text presents the full description of the architecture, simulation results and the results of the practical implementation on the NFB-200G2QL network card based on the Xilinx Virtex UltraScale+ chip.

1.4 Structure

The text of the thesis is structured into 5 chapters. Chap. 1 Thesis Overview gives the general overview of the scope and goals of the thesis. The chapter contains text describing the main motivation (Sec. 1.1), goals (Sec. 1.2), contribution (Sec. 1.3) and text structure (Sec.0 1.4).

Chap. 2 Power Analysis Fundamentals provides readers with the fundamentals of the power analysis methods, namely profiling power analysis attacks (Sec. 2.1) and non-profiling power analysis attacks (Sec. 2.2). Moreover, the chapter provides the description of basic countermeasure techniques, namely countermeasure methods (Sec. 2.3) and attacks on countermeasure methods (Sec. 2.4).

In Chap. 3, the description of the practical attacks targeting protected implementations is provided, all examples are based on a publicly available dataset (power traces). Furthermore, the text contains the analysis of the current state of the techniques utilized to protect AES implementations.

In Chap. 4, the proposal of a cryptographic algorithm that is protected by hiding is described. The implementation utilizes the independent operation and the hardware platform with a wide data-path. The text presents full description of the architecture, simulation results and the results of the practical implementation on the network card.

2 Power Analysis Fundamentals

This chapter that represents the pedagogical contribution of the thesis contains the theory necessary for understanding the power analysis. Parts of the chapter are used in a university textbook of the Information Security study program at Brno University of Technology, where the author is involved [163, 162]. We provide a brief overview of well know techniques that are utilized nowadays in order to attack implementations of cryptographic protocols. We introduce basic approaches of power analysis including the basic countermeasure methods.

The power analysis (PA) was introduced by Kocher and generally includes two basic methods: simple PA and differential PA [58]. In the simple power analysis (SPA), the adversary tries to determine the secret key directly from the measured traces. A typical example is the attack on an asymmetric cryptographic algorithm implementation [55]. On the other hand, the goal of the differential power analysis (DPA) attacks is to reveal the secret key of the cryptographic device by using a large number of power traces that were recorded while the device was encrypting or decrypting input data.

From a different perspective, one can divide the power analysis attacks into two main categories, namely profiling and non-profiling attacks. In *profiling attacks*, the adversary needs physical access to a pair of identical (similar) devices that we call the profiling device and the target device. Basically, these attacks consist of two phases: the profiling phase and the attack phase. In the first phase (profiling), the adversary analyzes the profiling device in order to approximate the leakage behavior and in the second phase (attack), the adversary attacks the target device. Typical examples are the Template-based Attack (TA) [21, 78, 24] and the Stochastic Approach (SA) [129, 128]. Practical aspects of template attacks have been discussed in [124, 42]. The profiling phase of TA was improved in [5, 7, 24]; in recent years, the cryptographic community has been exploring the potential of TA based on machine learning approaches [54, 68, 8, 69, 71, 81, 23, 50]. By contrast, *non-profiling attacks* are one-phase attacks that perform the attack directly on the target device. The adversary measures a set of power traces from known plain text and compares these real power traces with hypothetical power consumption values. Such values were previously calculated based on a secret key hypothesis and a power consumption model [19, 2]. The comparison can be carried out by using diverse statistical methods [78]. After the analysis, only the correct key hypothesis will show dependency between the hypothetical and the actual power consumption measured.

2.1 Profiling Power Analysis Attacks

In the following, we use capital letters for random variables and small caps for their realizations. We use sans serif font for functions (e.g., F) and calligraphic fonts for sets (e.g., $\mathcal{A}$). We denote the conditional probability of a random variable A given B with $\Pr[A \mid B]$. Let l_y be a leakage measured (power trace) on a device that manipulates a target value y (also known as label and class). Let l_y^j be the j-th measured leakage associated with the target value y. Let $l_y(t)$ be the t-th time sample (also known as a feature) of the leakage trace l_y. This sample represents the interesting point of a leakage. We consider contexts where each trace l_y represents a vector of n_s interesting points.

$$l_y = [l_y(t) \in \mathbb{R} \mid t \in [1; n_s]]. \tag{2.1}$$

The samples are defined as the sum of a data-dependent leakage function (denoted as δ) and a random part representing the noise (denoted as ϵ), that is:

$$l_y(t) = \delta_t(y) + \epsilon_t. \tag{2.2}$$

We will sometimes omit the subscripts for simplicity. The profiling set $\mathcal{L}_{\mathrm{PS}}$ (sometimes denoted as a training set or a learning set) represents a set of N_p (profiling) leakages measured on a device under control and similar to the target device. This set of leakages allows during the profiling step to estimate a parameter θ used in the profiled model (denoted $\mathsf{A}(\mathcal{L}_{\mathrm{AS}}, \theta)$) that returns, during the attack step, the most probably target value y based on an attacking set $\mathcal{L}_{\mathrm{AS}}$ (that contains attack leakages) obtained by measuring the target device. Algorithm 1 summarises how a profiled model $\mathsf{A}(\cdot, \cdot)$ predicts the target value with a profiling and an attacking sets.

Algorithm 1 How to predict the most likely target value associated with $\mathcal{L}_{\mathrm{AS}}$.

Require: A profiling set $\mathcal{L}_{\mathrm{PS}}$ and an attacking set $\mathcal{L}_{\mathrm{AS}}$

Ensure: The prediction $\hat{y}$ of a profiled model $\mathsf{A}(\cdot, \cdot)$

1. Profiling step:
   (a) Implement the crypto algorithm on a controlled device (similar to the target device)
   (b) Collect a set of profiling leakages for each target value on the controlled device
   (c) Estimate the parameter θ with the profiling set (denoted $\mathcal{L}_{\mathrm{PS}}$)
2. Attack step:
   (a) Collect a set of attack leakages (denoted $\mathcal{L}_{\mathrm{AS}}$) on the target device
   (b) $\hat{y} = \mathsf{A}(\mathcal{L}_{\mathrm{AS}}, \hat{\theta})$

2.1.1 Standart Template Attack

Side-channel attack based on templates is a typical example of the profiling attack and exploits how the power consumption depends on the processed data. In template attacks (TA), a multivariate normal distribution is used to characterize the power traces of the profiling device. In other words TAs use the profiling set $\mathcal{L}_{\mathrm{PS}}$ in order to estimate a leakage model per target value y denoted as $\hat{\Pr}_{\mathrm{model}}\left[l_y \mid \hat{\theta}_y\right]$ where $\hat{\theta}_y$ represents the (estimated) parameters of the leakage probability density function. During the attack step, template attacks use an attacking set $\mathcal{L}_{\mathrm{AS}}$ and select the target value $\hat{y}$ maximizing the product of posterior probabilities:

$$\hat{y} = \mathsf{A}(\mathcal{L}_{\mathrm{AS}}, \hat{\theta}) = \underset{y}{\operatorname{argmax}} \prod_{l \in \mathcal{L}_{\mathrm{AS}}} \frac{\hat{\Pr}_{\mathrm{model}}\left[l \mid \hat{\theta}_y\right] \cdot \Pr[y]}{\hat{\Pr}_{\mathrm{model}}[l]}, \tag{2.3}$$

where $\hat{\theta}$ represents the set of parameters. We consider that the parameter $\hat{\theta}_y$ corresponds to the mean vector $\hat{\mu}_y$ and the covariance matrix $\hat{\Sigma}_y$ of the Gaussian (leakage) probability density function associated with the target value y as proposed by the seminal work of Chari *et al.* [22], i.e.:

$$\hat{\Pr}_{\mathrm{model}}\left[l \mid \hat{\theta}_y = \{\hat{\mu}_y, \hat{\Sigma}_y\}\right] = \frac{1}{\sqrt{(2\pi)^{n_s} \det(\hat{\Sigma}_y)}} e^{-\frac{1}{2}(l-\hat{\mu}_y)\hat{\Sigma}_y^{-1}(l-\hat{\mu}_y)^{\top}}, \tag{2.4}$$

where $\det(\hat{\Sigma})$ denotes the determinant of the matrix $\hat{\Sigma}$. We call this conventional template attack as the Classical Template Attack (CTA) in the following. Furthermore, we consider the Efficient Template Attack (ETA) suggested by Choudary *et al.* [24] in which we pool the covariance matrices across all the target values. In other words, ETA estimates one covariance matrix with all the leakages obtained in the profiling set. As might be expected, the template that leads to the highest probability indicates the correct prediction of the target value y:

$$\underset{\forall y}{\operatorname{argmax}}\, \hat{\Pr}_{\mathrm{model}}\left[l \mid \{\hat{\mu}_y, \hat{\Sigma}_y\}\right]. \tag{2.5}$$

During the attack phase, some difficulties occur that are related to the covariance matrix. First, the size of the covariance matrix grows quadratically with the number of points in the trace. Therefore, we focus the profiling phase only on the interesting points in power trace that leakage information. In this respect, the selection of interesting points is a crucial aspect of every profiling power analysis attacks. Some usual techniques utilized to localize interesting points are: Normalized Inter-Class Variance (NICV) [13], Sum Of Squared pairwise Differences (SOSD) [36], Sum Of Squared pairwise T-differences (SOST) [36], Principal Components Analysis (PCA) [96, 9] or Pearson Correlation [78]. Second, the covariance matrix tends to be badly conditioned [24]. This means, we run into numerical problems during the inversion,

which needs to be done in Eq. 2.4. Also, the values that are calculated in the exponent tend to be very small, which often leads to more numerical problems. To avoid the exponentiation, one can apply the logarithm to Eq. 2.4 and then, the template which leads to the smallest absolute value of the logarithm indicates the desire value:

$$\ln \hat{\Pr}_{\text{model}}\left[l \mid \hat{\theta}_y\right] = -\frac{1}{2}(\ln((2\cdot\pi)^{n_s}\cdot\det(\hat{\Sigma}_y)) + (l-\hat{\mu}_y)\cdot\hat{\Sigma}_y^{-1}\cdot(l-\hat{\mu}_y)^{\top}), \quad (2.6)$$

$$\underset{\forall y}{\operatorname{argmin}} \left|\hat{\Pr}_{\text{model}}\left[l \mid \{\hat{\mu}_y, \hat{\Sigma}_y\}\right]\right|. \quad (2.7)$$

One can set the covariance matrix equal to the identity matrix to avoid problems with the inversion of the covariance matrix. This essentially means that we do not take the covariances between the points into account. A template that only consists of a mean vector is called a *reduced template*. Setting the covariance matrix equal to the identity matrix simplifies the multivariate normal distribution:

$$\ln \hat{\Pr}_{\text{model}}\left[l \mid \hat{\theta}_y\right] = -\frac{1}{2}(\ln((2\cdot\pi)^{n_s}) + (l-\hat{\mu}_y)\cdot(l-\hat{\mu}_y)^{\top}). \quad (2.8)$$

Elementary Example of the Template Attack

In the previous text, template attacks that allow exploiting the data dependency in power traces were introduced. Therefore, the following text describes a practical example of these attacks. In this elementary example, we investigate how to apply the concept of a template attack that is aimed at a real cryptographic module constituted by the micro-controller PIC16F84A. At first, the AES algorithm was implemented into a cryptographic module using the assembly language. We measured power traces that correspond with the `AddRoundKey` operation that is executed by `MOV` and `XOR` instructions for various input data. Measured power traces of the `MOV` instruction are depicted in Fig. 2.1. From the power traces we can observe the following facts.

- Several points within the power trace are proportional or inversely proportional to the Hamming weight (HW) of the processed data (it depends on the type of the data-bus, in our case, the power consumption is inversely proportional to the Hamming weight because our micro-controller has a precharged data-bus).
- Moreover, 9 groups corresponding with the Hemming weight of processed data are clearly distinguished and the distances between the groups are nearly equal.
- Therefore, we assume that it is possible to build templates that allow us to classify `MOV` instructions according to the Hamming weight. These templates allow us to deduce the Hamming weight of the processed data during the attack phase.

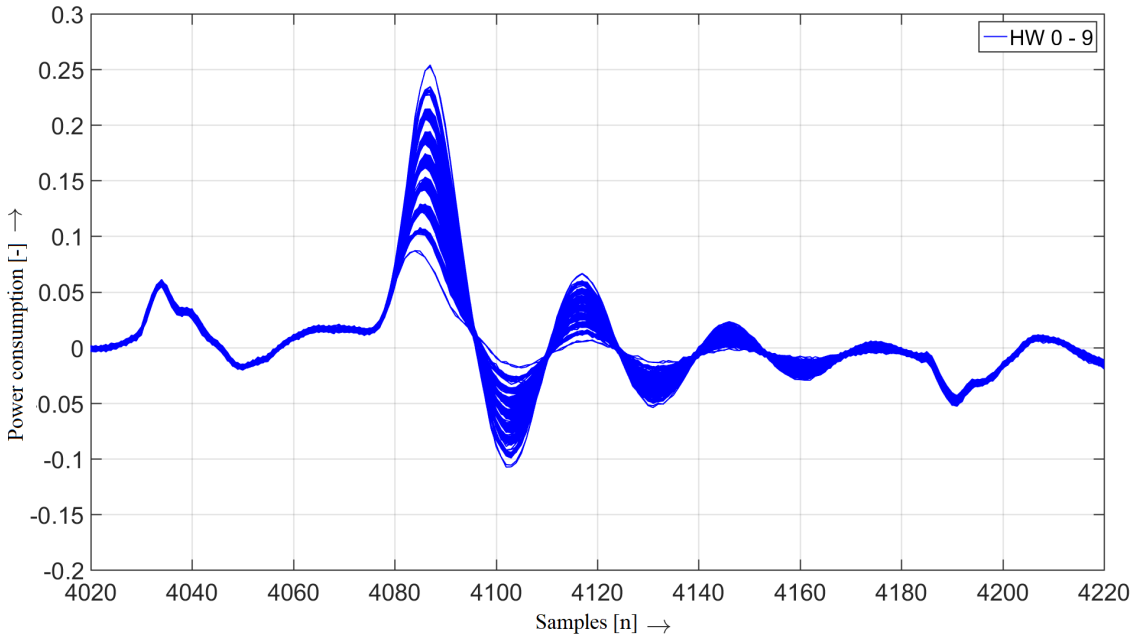


Fig. 2.1: Power consumption of `MOV` instruction.

In this example, we need to built nine templates altogether to cover every Hamming weight of processed data (8 bit micro-controller, $y = (0, \ldots, 8)$). Remember that a template consists of a mean vector and a covariance matrix, $h_y = (\hat{\mu}_y, \hat{\Sigma}_y)$. As pointed out before, the size of the covariance matrix grows quadratically with the number of points therefore we have to identify the interesting points in order to minimize the templates size. We localized the interesting points, for which the power traces differ significantly, directly from Fig. 2.1. Finally, we took four of those points in order to create the templates ($n = (4091, 4105, 4119, 4134)$). The example of the template $h_0 = (\hat{\mu}_0, \hat{\Sigma}_0)$ that operates on data with the Hamming weight 0 is given by the following equation:

$$\hat{\Sigma}_0 = \begin{pmatrix} 2,94 & -1,45 & 0,61 & -0,14 \\ -1,45 & 1,02 & -0,65 & 0,23 \\ 0,61 & -0,65 & 1,07 & -0,34 \\ -0,14 & 0,23 & -0,34 & 0,28 \end{pmatrix} \cdot 10^{-4}, \tag{2.9}$$

$$\hat{\mu}_0 = \begin{pmatrix} 0,47 & 0,32 & 0,17 & 0,18 \end{pmatrix} \tag{2.10}$$

The profiling phase is finished when the desired templates are built. Now, we examine how well these templates match to nine power traces, stored in the matrix $\mathcal{L}_{\mathrm{AS}}$, that have been acquired during the attack phase. The first power trace l_0, i.e. the first row of $\mathcal{L}_{\mathrm{AS}}$, corresponds to the execution of the `MOV` instruction with data of Hamming weight 0, the second power trace l_1 corresponds to data with Hamming weight 1, etc.

$$\mathcal{L}_{\mathrm{AS}} = \begin{pmatrix} 0,48 & -0,33 & 0,18 & -0,19 \\ 0,73 & -0,41 & 0,23 & -0,24 \\ 0,95 & -0,54 & 0,30 & -0,31 \\ 1,14 & -0,66 & 0,37 & -0,36 \\ 1,36 & -0,72 & 0,41 & -0,39 \\ 1,54 & -0,82 & 0,47 & -0,43 \\ 1,72 & -0,92 & 0,50 & -0,45 \\ 1,91 & -1,00 & 0,59 & -0,50 \\ 2,05 & -1,11 & 0,65 & -0,55 \end{pmatrix}. \tag{2.11}$$

Now, we match our templates prepared in the profiling phase with the first power trace l_1 of $\mathcal{L}_{\mathrm{AS}}$ by computing the equation 2.8. The results are:

$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_0) = 15,05 \tag{2.12}$$
$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_1) = -28,36 \tag{2.13}$$
$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_2) = -93,40 \tag{2.14}$$
$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_3) = -166,37 \tag{2.15}$$
$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_4) = -291,90 \tag{2.16}$$
$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_5) = -543,02 \tag{2.17}$$
$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_6) = -953,24 \tag{2.18}$$
$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_7) = -2,06 \cdot 10^3 \tag{2.19}$$
$$\ln \hat{\Pr}_{\mathrm{model}}(l_0; h_8) = -2,50 \cdot 10^4 \tag{2.20}$$

The template h_0 fits best the first power trace l_0, since this provides the lowest absolute value. Indeed the trace l_0 was acquired where the `MOV` instruction processed data with the Hamming weight equal to 0. To finalize the attack phase explanation, Tab. 2.1 summarizes calculation of the template matching in integer numbers for every row of the matrix. We observe, that every of the 9 target values was revealed correctly and our prepared templates can be utilized to uncover the HW of the processed data.

It is obvious, that our example signified the basic utilization of a standard template attack. Similarly, as in our simple example, an adversary is able to prepare templates for various intermediate values. As an illustration, the output of the first S-box of the AES algorithm is selected and 256 templates prepared in order to reveal the secret key are stored. Another typical example is a template preparation to reveal mask values of a resilient implementation where countermeasure techniques are implemented.

Tab. 2.1: Results of template matching of whole $\mathcal{L}_{\mathrm{AS}}$.

	l_0	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
h_0	15	-222	-702	-1233	-2216	-3051	-3822	-5410	-6337
h_1	-28	9	-47	-182	-368	-610	-940	-1264	-1646
h_2	-93	-8	11	-35	-121	-247	-410	-636	-865
h_3	-166	-58	4	8	-22	-85	-181	-318	-471
h_4	-292	-151	-38	5	8	-17	-71	-174	-300
h_5	-543	-329	-145	-45	-2	10	-20	-97	-208
h_6	-953	-625	-354	-177	-69	-4	9	-30	-110
h_7	-2067	-1429	-932	-568	-296	-113	-17	9	-36
h_8	-25016	-17357	-12407	-8642	-4826	-2606	-999	-163	13

2.1.2 Template Attack Based on Machine Learning

In recent years, the cryptographic community has explored new approaches in power analysis based on machine learning (ML). In general, machine learning approaches can be classified as supervised [61] and unsupervised learning [43]. Intuitively, in supervised learning, the machine is presented with a set of training data with the label and the goal is to determine the general function that associates the data with the label. In unsupervised learning, the machine is presented with a set of unlabeled data, and the machine tries to determine the hidden structure of the data. From the description above, one can clearly see an analogy between machine learning approaches and power analysis attacks. More specifically, profiling attacks are a supervised learning problem, where ML techniques are used for a model creation of the target device. The most commonly used techniques are listed in the following text.

Support Vector Machines

Support Vector Machines (SVM) are the most successful techniques in classification [28]. In a binary classification setting (e.g., y = -1 or y = 1), if the two classes are separable, SVM compute from the profiling set a separating hyperplane $w^\top T + b$ (where w and b are estimated values) allowing to estimate the target value $\hat{y}$ from a leakage l according to the decision rule:

$$\hat{y} = \mathsf{A}(l, \theta) = \begin{cases} 1 & w^\top l + b > 0 \\ -1 & \text{otherwise} \end{cases}, \tag{2.21}$$

where $\theta = \{w \in \mathbb{R}^{n_s}, b \in \mathbb{R}\}$, and $\{-1, 1\}$ represents the space of target values.

In order to reduce the error due to noise in the profiling leakages, SVM select the hyperplane with the maximal margin, where the margin is the sum of the distances from the hyperplane to the closest profiling leakages of each of the two classes. Cortes *et al.* [28] show that solving the following convex optimisation problem allows to select the value of w and b that maximise the margin:

$$\min_{w} \frac{1}{2}(w^\top w), \tag{2.22}$$

subject to:

$$y(w^\top l_y^j + b) \geq 1 \quad \forall j, y \tag{2.23}$$

in the case of binary labels $y \in \{-1, 1\}$. By introducing Lagrange multipliers (denoted by $\alpha_{j,y} \in \mathbb{R}$), Cortes *et al.* show that the convex optimisation problem can be solved with a linear weighted sum of the profiling leakages. As a result, the decision rule becomes:

$$\hat{y} = \begin{cases} 1 & w^\top l + b > 0 \Leftrightarrow \left(\sum_{l_y^j \in \mathcal{L}_{\mathrm{PS}}} \alpha_{j,y} \times y \times l_y^j\right)^\top l + b > 0 \\ -1 & \text{otherwise} \end{cases}. \tag{2.24}$$

In a compact manner, we write the decision rule as follows:

$$\hat{y} = \begin{cases} 1 & \sum_{l_y^j \in \mathcal{L}_{\mathrm{PS}}} \alpha_{j,y} \times y \times \phi\left(l_y^j, l\right) + b > 0 \\ -1 & \text{otherwise} \end{cases}, \tag{2.25}$$

where ϕ performs the product of two vectors. An interesting feature of the SVM is that it is possible to adapt the classifier to nonlinear classification tasks by performing a nonlinear transformation φ of the leakages, the decision rule becomes:

$$\hat{y} = \begin{cases} 1 & \sum_{l_y^j \in \mathcal{L}_{\mathrm{PS}}} \alpha_{j,y} \times y \times \phi\left(\varphi\left(l_y^j\right), \varphi\left(l\right)\right) + b > 0 \\ -1 & \text{otherwise} \end{cases}. \tag{2.26}$$

We suppose that $\kappa(\cdot, \cdot)$ (called the kernel function) performs the transformation $\phi\left(\varphi\left(\cdot\right), \varphi\left(\cdot\right)\right)$ leading to the decision rule:

$$\hat{y} = \begin{cases} 1 & \sum_{l_y^j \in \mathcal{L}_{\mathrm{PS}}} \alpha_{j,y} \times y \times \kappa\left(l_y^j, l\right) + b > 0 \\ -1 & \text{otherwise} \end{cases}. \tag{2.27}$$

Our experiments that are presented in the following section (Sec. 3.3) consider a Radial Basis kernel Function κ (RBF), which is a commonly encountered solution. The radial basis kernel function maps the leakages into an infinite dimensional Hilbert space in order to find a hyperplane that efficiently discriminates the leakages. RBF is defined by a meta-parameter γ related to the complexity of the model. In our experiments, we set γ equal to $\frac{1}{n_s}$, which is a natural choice to compensate

the increase of the model complexity due to the increase of the number of points per leakage. SVM can be generalised to multi-class problems. In our experiments (Sec. 3.3), we consider the "*one-against-all*" approach. In a one-against-all strategy, the adversary builds one binary support vector machine for each target value in order to separate leakages of that target value from leakages of other target values.

Random Forests

Random Forests (RF) represent a set of Decision Trees (DT). DT are structured as diagrams made of nodes and directed edges, where nodes can be of three types: root (i.e., the top node in the tree), internal and leaf. We consider DT in which (1) the value associated to a leaf is a label, (2) each edge is associated to a test on the value of a feature, and (3) each internal node has one incoming edge from a node called the parent node and two outcoming edges to two nodes (called left child and right child).

In the profiling step, the DT generator first associates the whole profiling set to the root. Then the generator splits the set associated to the node in two subsets (called left set and right set) based on a feature that most effectively discriminates the set of leakages associated to different target values. Each newly created subset is associated with a child node: the left set (respectively the right set) is associated to the left child (respectively the right child). The tree generator repeats this process on each derived subset in a recursive manner, until the gain to split the subset is less than some threshold. Eventually, the learning algorithm assigns to each leaf the majority class of leakages in that node. The tree construction may be followed by an additional step called the pruning step in which the DT is simplified by substituting a single leaf in place of a whole sub-tree. In the attack step, the model predicts the label by applying the classification rules (represented by the conditions along the path from the root to a leaf) to the unlabelled leakage to be classified.

RF were introduced by Breiman in 2001 to address the problem of instability in large DT, where by instability we denote the sensitivity of a DT structure to small changes in the profiling set (also known as the variance issue) [15]. In order to reduce the variance, RF rely on the principle of models averaging by building a number of DT and returning the most consensual prediction. This means that the predicted output $\hat{y}$ of an attack leakage is calculated through a majority vote of the set of trees. RF are based on two aspects. First each tree is constructed with a different set of profiling leakages through the boostrapping method. This method builds a profiling set (called a *bootstrap sample*) for each DT by sampling with replacement the original profiling set. Secondly, each tree is built by adopting a random partitioning criterion. This idea allows to obtain decorrelated trees, thus

improving the accuracy of the resulting RF. More precisely, in conventional DT each node is split using the best split among all features. In the case of RF, each node is split using the best among a subset of features randomly chosen at that node. In our practical experiments in power analysis attacks (Sec. 3.3), we consider a subset of $\sqrt{n_s}$ features. Also, unlike conventional DT, the trees of the RF are fully grown and are not pruned. In other words, each leaf contains leakages associated to the same target value. This implies the null profiling error but a large variance and consequently a small success rate for each single tree. The average of trees represents a remedy to the variance issue, and allows the design of an overall more accurate predictor.

Multilayer Perceptrons

We use MultiLayer Perceptrons (MLP) executing basic functions called neurons (also known as perceptrons) that output values between -1 and 1. A neuron generates the output value y by executing the composition of two functions f and g, that is:

$$y = \mathsf{f}\left(\mathsf{g}\left(x, \theta\right)\right), \tag{2.28}$$

where $x = [x(0), x(1), ..., x(N)]$ is the input vector, $\mathsf{f}\left(\cdot\right)$ is a nonlinear function (called the activation function), and $\mathsf{g}\left(\cdot, \theta\right)$ is a linear function (parameterized by θ) allowing to transform a vector of real numbers to a scalar. Our experiments consider the nonlinear weighted sum function, equation 2.28 can be rewritten as:

$$y = \mathsf{f}\left(\mathsf{g}\left(x, \theta\right)\right) = \mathsf{f}\left(\sum_{i=0}^{N} w(i)x(i) + \sigma\right), \tag{2.29}$$

where θ is represent with weights $[w(0), w(1), ..., w(N)]$ and bias σ. The MLP increase the capacity of a neuron by grouping neurons in two or more layers. The connection between the i-th neuron and the j-th neuron is defined by the weight $w_j(i)$ and σ_j (where $\theta_j = [w_j(0), w_j(1), ..., \sigma_j]$ is the parameter of the j-th neuron). The first, the last and the middle layers are called respectively input, output and hidden layers.

The input of a neuron in a (hidden or output) layer equals to the weighted output of the neurons associated to the previous layer. In power analysis attacks, the input layer contains n_s neurons (i.e., one input neuron per feature) while the output layer contains Y neurons (where Y is the number of possible target values). As a result, based on one leakage l, each neuron from the input layer (1) manipulates one point in the leakage l, and (2) forwards the result of the manipulation to the next layer. Eventually, each neuron from the output layer provides a score for each target value associated to the input leakage l, and the predicted value $\hat{y}$ represents the target value having the highest score.

The profiling step adjusts each parameter θ_j to achieve a desired output value. For this, the network of neurons uses a supervised learning technique called the backpropagation algorithm that minimizes for each leakage in the profiling set the difference between the target value and the target value generated by the network. For the sake of shortness, we refer to the book of Bishop [14] for a deeper introduction to multilayer perceptrons and to [88, 92, 81] for a presentation of MLP in power analysis. Our power analysis experiments use two-layers neural networks containing 200 neurons in the hidden layer and using the sigmoid function as the nonlinear function $\mathsf{f}(\cdot)$.

Elementary Example of Machine Learning Approach

As in the previous standard template attack,in this section we provide a practical elementary example of a template attack based on machine learning. We chose the MLP as a representative because it is the most used technique together with the SVM approach. One more time, we utilize power traces that correspond with the AES implementation on PIC16F84A and our attack reveals the HW of the `MOV` instruction during the `AddRoundKey` operation. The profiling set $\mathcal{L}_{\mathrm{PS}}$, same as attacking set $\mathcal{L}_{\mathrm{PS}}$, contains 500 power traces. Implementation of machine learning algorithms can be accomplished in various environments. In Fig. 2.2, our implementation in RapidMiner studio is depicted. We chose this implementation due its simplicity.

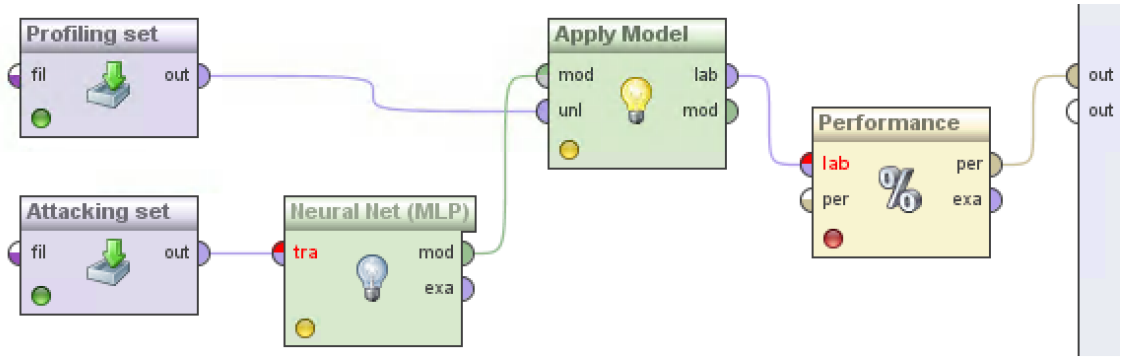


Fig. 2.2: Example of template attack based on MLP.

The result of the attack, or rather classification, in the context of machine learning, is mostly in a form of a confusion matrix. This matrix evaluates all guesses carried out during the attack phase and it is often used together with the guessing entropy as a metric of different side-channel attacks comparison [135, 34]. Interested readers can consult [132] to obtain additional explanation about performance measurements for classification, e.g. confusion matrix, precision, recall.

The resulting confusion matrix corresponding with the HW revelation is shown in Table 2.2. Each column of the table corresponds to the correct values of the target (in this example HW of a byte) and each row corresponds to the predicted values. For example, HW with zero value (the first column) was wrongly estimated two times as $HW = 1$. This HW was classified 53 times correctly. On the other hand, there was no wrong estimate for the HW values equal to $2, 3, 4, 5, 6, 7$ and 8. Accuracy of the classification was $99,47\%$.

Tab. 2.2: Confusion matrix for HW classification utilizing the MLP.

true	HW 0	HW 1	HW 2	HW 3	HW 4	HW 5	HW 6	HW 7	HW 8
pred. HW 0	53	2	0	0	0	0	0	0	0
pred. HW 1	1	57	0	0	0	0	0	0	0
pred. HW 2	0	0	71	0	0	0	0	0	0
pred. HW 3	0	0	0	69	0	0	0	0	0
pred. HW 4	0	0	0	0	56	0	0	0	0
pred. HW 5	0	0	0	0	0	58	0	0	0
pred. HW 6	0	0	0	0	0	0	72	0	0
pred. HW 7	0	0	0	0	0	0	0	69	0
pred. HW 8	0	0	0	0	0	0	0	0	67

2.2 Non-profiling Power Analysis Attacks

A setup of the Diferential Power Analysis (DPA) deploying the correlation coefficient and the Hamming weight power consumption model represents a typical example of non-profiled attacks. Therefore the following text describes the general schema of the DPA attacks [78]. DPA requires a large amount of measured waveforms of current consumption of cryptographic devices which encrypt or decrypt the input data. DPA uses mathematical tools, namely the statistical analysis and techniques to determine the secret key from measured power consumptions. The general schema of DPA composed of five steps that are graphically shown in Fig. 2.3 [80][1].

In the **first** step of the DPA attack, the attacker determines the intermediate value of the encryption algorithm, which is performed by a cryptographic device. This internal value must be a function $f(d, k)$, where d are known input data (typically plain text or cipher text) and k represents a small part of the secret key, which the attacker wants to establish (mostly the first byte).

[1] In the typical example of a DPA attack, the intermediate value is the output of the SBOX operation in the first round of the AES algorithm. The figure follows this application

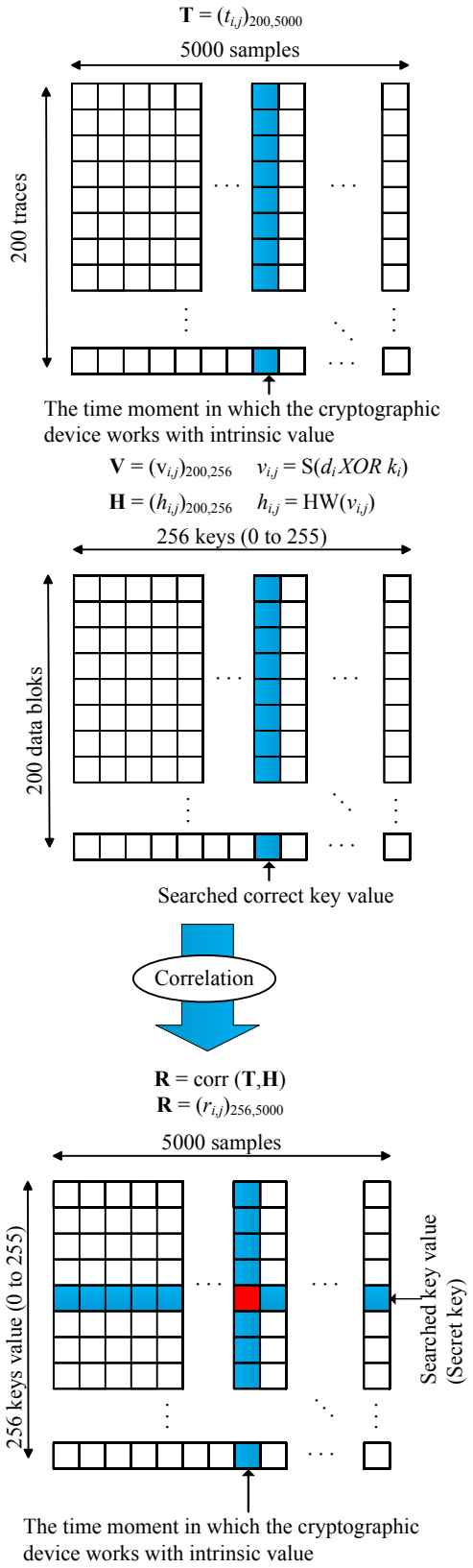


Fig. 2.3: Diagram of the general schema of DPA.

The **second** step of the DPA attack is a measurement of the power consumption of the cryptographic device which is encrypting or decrypting different blocks of data D. For all encryption and decryption operations, the attacker needs to know the value of the processed data d, which is directly used for calculation of the intrinsic value determined in the first step. Known values are written in the vector $\mathbf{d} = (d_1, \ldots, d_D)'$, where d_i indicates the result of the i-th block of the processed input data. The attacker records the power consumption during performing these operations. To each power consumption $\mathbf{t_i}' = (t_{i,1}, \ldots, t_{i,T})$, where T denotes the duration of the power trace, corresponds one value of processed data d_i. The attacker measures the power consumption of each block of processed data D, therefore the traces can be written as a matrix $\mathbf{T}$ with dimensions $D \times T$. Precision and correct alignment of the measured power consumption (synchronization) is critical for DPA attacks. In other words, it means that the value of the current consumption in any column of the matrix must correspond to the same operation.

The **third** step is the calculation of a hypothetical intrinsic value for every possible key estimation k. Possible key values can be written as a vector $\mathbf{k} = (k_1, \ldots, k_K)$, where K is the total number of possible keys. Individual elements of the vector are called hypotheses or estimates of the key. The attacker is able to easily calculate the hypothetical intrinsic value as a function $f(d, k)$ for all cryptographic operations D and for all hypotheses keys K from known data vector $\mathbf{d}$ and from the key hypothesis. The result is the matrix $\mathbf{V}$.

The **fourth** step is mapping hypotheses intrinsic values, it means matrix $\mathbf{V}$, into a matrix $\mathbf{H}$ which is representing the hypotheses values of the power consumption. The simulation of power consumption is used for this purpose. The created model of power consumption assigns for each hypothetical intrinsic value $v_{i,j}$ a hypothetical power consumption value $h_{i,j}$. The quality of the simulation strongly depends on the knowledge of the attacker about the analyzed device. The better the simulation of the attacker matches the actual power consumption characteristics of the device, the more effective the DPA attack is.

In the **fifth** step, the hypothetical values of power consumption which depend on keys hypotheses are compared with measured power traces. The result is a matrix $\mathbf{R}$. Each element $r_{i,j}$ of the matrix $\mathbf{R}$ is the comparison between the columns $\mathbf{h_i}$ and $\mathbf{t_j}$. The comparison is based on statistical methods which are discussed in the following chapter. All methods have the same main attribute that the higher the value $r_{i,j}$ is, the better the columns $\mathbf{h_i}$ and $\mathbf{t_j}$ match.

2.2.1 Correlation Coefficient

The correlation coefficient is one of the best known methods to determine the linear relationship between two random variables. Therefore, it is also a suitable method for performing DPA attacks. A very well defined theory exists for the correlation coefficient which can be used to model the static properties of DPA attacks. The correlation coefficient is defined by the covariance as follows:

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sqrt{\sigma^2(X) \cdot \sigma^2(Y)}}. \tag{2.30}$$

It is a dimensionless quantity and it can only take values between plus and minus one $-1 \leq \rho \leq 1$. Value -1 of the correlation coefficient denotes indirect dependence (change in one group is accompanied by an opposite change in the second group). Value 0 indicates that detectable statistic dependence between values of the two groups does not exist. If the correlation coefficient is equal to 1, it indicates a direct dependence or a perfect correlation between the values of the two groups. Also ρ is typically unknown and must be estimated. The estimator r is defined by the following equation:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x}) \cdot (y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \cdot \sum_{i=1}^{n}(y_i - \overline{y})^2}}. \tag{2.31}$$

In DPA attacks, the correlation coefficient is used to determine the linear dependence between the columns h_i and t_j were $i = 1, \ldots, K$ and $j = 1, \ldots, T$. The result is a matrix $\mathbf{R}$ containing correlation coefficients and each value $r_{i,j}$ is based on elements of D from the columns h_i and t_j. Using the previous definition of the correlation coefficient, we can rewrite Eq. 2.31 as:

$$r_{i,j} = \frac{\sum_{d=1}^{D}(h_{d,i} - \overline{h_i}) \cdot (t_{d,j} - \overline{t_j})}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \overline{h_i})^2 \cdot \sum_{d=1}^{D}(t_{d,j} - \overline{t_j})^2}}, \tag{2.32}$$

where $\overline{h_i}$ and $\overline{t_j}$ indicate average values of the columns h_i and t_j .

Elementary Example of DPA Attack

In the example, we consider a smart card based on the chip Atmel ATMega-163, and we decided to choose the output byte of the first S-box of the unmasked software AES implementation. This intermediate value is a function of the first byte of plain text and the first byte of the secret key (the first step). In our test-bed, we recorded the power consumption of the smart card during the first round of AES while it has encrypted 300 different plain texts. This second step led to the matrix $\mathbf{T}$ that contains power traces. The following step of the DPA attack lies in calculation of

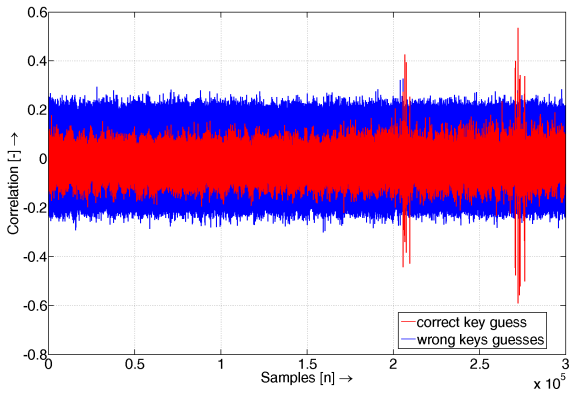


Fig. 2.4: Result of DPA attack based on correlation coefficient.

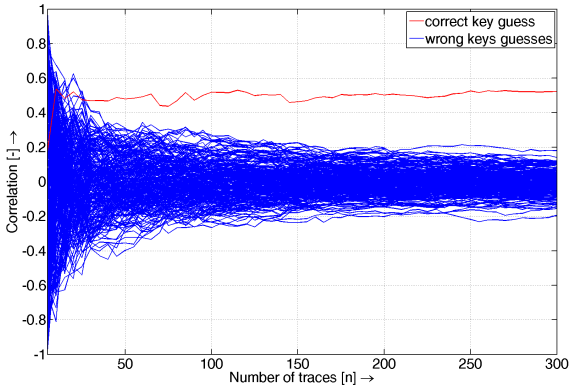


Fig. 2.5: Size of the correlation depending on the number of power traces.

hypothetical intermediate values based on the 300 known plain texts. In other words, we have to calculate $v_{i,j} = SBOX(d_i \oplus k_j)$, where $d_1, \ldots, d_{300}$ are the first bytes of all plain texts and k_j for $j = 0, \ldots, 256$ represents the key hypothesis. Resulting matrix $\mathbf{V}$ has hence the size of 300×256 values. The fourth step comprises mapping of the matrix $\mathbf{V}$ to the matrix of hypothetical power consumption values denoted as $\mathbf{H}$. In the example attack, we utilized a Hamming weight power simulation model (power simulation models are explained in Sec. 2.2.3) because it corresponds to the device under attack in the best way. During the last step, we calculated the correlation coefficients between all columns of the matrix $\mathbf{H}$ and all columns of the matrix $\mathbf{T}$. The result of this calculation is a matrix $\mathbf{R}$ of correlation coefficients.

In practice, there exist several different ways how to visualize the resulting matrix $\mathbf{R}$. One of these ways is to display each row of the matrix in a single plot and the plot with the highest peak is distinguished by using different colors. Naturally, this marked plot should correspond with the correct key hypothesis. We present this example in Fig. 2.4. It can be observed that there are very high peaks in the plot only for the correct key hypothesis. In fact, all other values of $\mathbf{R}$ are significantly smaller. This fact provides to the attacker basically two pieces of information. The value of the first secret key byte and the time intervals where `SubBytes` operation is performed for the first byte of the plain text. In our example, the first byte of the secret key equals to 130 and the `SubBytes` operation is performed at $2 \cdot 10^5$ and at $2.7 \cdot 10^5$. Moreover, we observe that the chosen intermediate value is used in several instructions. This is very typical for software implementations because after the intermediate value is calculated, it is usually moved from a register to memory and later it is loaded back from memory to the register for subsequent operations of the algorithm.

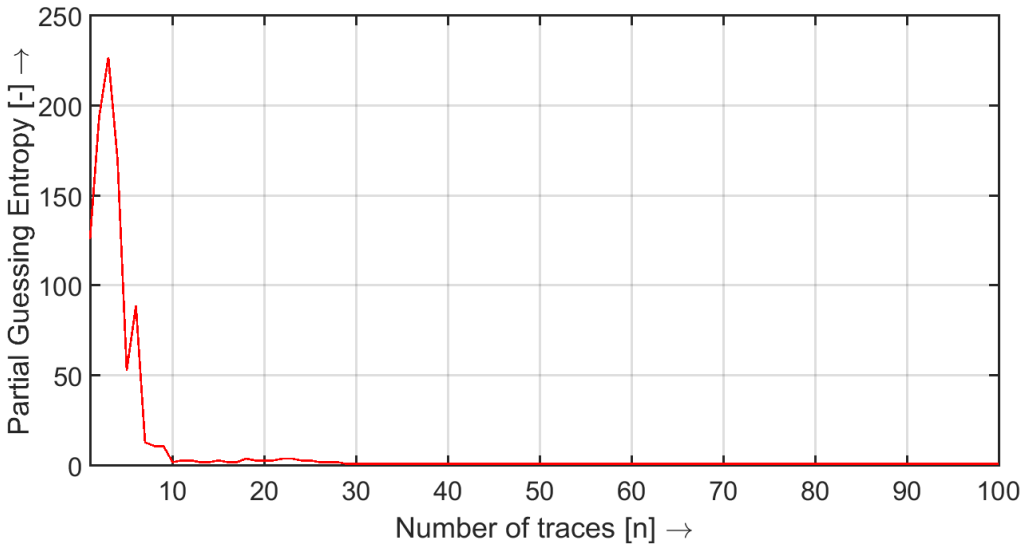


Fig. 2.6: Example of PGE for the first key byte.

Frequently, in scientific and education literature, dependence of the correlation coefficient on the number of power traces is depicted in a plot in order to demonstrate the complexity of the attack realization. This complexity is given by the number of power traces necessary to reveal the secret key with certainty. This plot is depicted in Fig. 2.5. It is conspicuous that an attacker needs about 30 power traces in our example attack (this observation applies also for the remaining key bytes).

To express the effectiveness of a DPA attack, one often utilizes a metric denoted as guessing entropy (GE). Furthermore, the guessing entropy is a metrics suitable for the evaluation of various implementations of power side-channel attacks. It describes the amount of attacker's work necessary to learn the complete key. Therefore the subsequent sections employ this metric . The guessing entropy is defined as follows: let $\mathbf{g} = [p_1, p_2, \ldots, p_N]$ contain the probabilities $p_1 \geq p_2, \geq \ldots, \geq p_N$ of all possible key candidates after attack realization. Indices i correspond with the correct key in $\mathbf{g}$. After the realization of S experiments, one obtains a matrix $\mathbf{G} = [g_1, ..., g_S]$ and a corresponding vector $\mathbf{i} = [i_1, \ldots, i_S]$. Then the guessing entropy determines the average position of the correct key:

$$GE = \frac{1}{S} \sum_{x=1}^{S} i_x. \tag{2.33}$$

In other words, the guessing entropy describes the average number of guesses, required for recovering the secret key [134, 49]. In the following text, we use this value as a metric for individual key bytes revelation (sometimes, the value is denoted as the partial guessing entropy PGE). The plot of PGE is depicted in Fig. 2.6. It confirms that an attacker needs 29 power traces to reveal the firs key byte.

2.2.2 Difference of Means

The basis of statistical methods based on the difference of means is a comparison of two measured groups by calculating the difference of the mean values of these groups. A systematic description of the method is given in [58] and the optimization of this method is described in [84]. This method uses a different technique to determine the relationship between the columns of the matrices **H** and **T**.

The attacker creates a binary matrix **H** which divides the measured power traces into two groups. Sequence of zeros and ones in each column **H** is a function of the input data d and estimates the key value k_i. In order to determine if the estimate of the key k_i is correct, the attacker can divide the matrix **T** into two sets of lines (two sets of power consumption) by h_i. The first set contains the lines of **T**, where the index corresponds to the positions of zeros in the vector h_i. The second set contains the remaining rows of **T**. Subsequently, the attacker calculates the means of the rows. Vector $m_{0i}^{'}$ denotes the averages of the rows in the first set and $m_{1i}^{'}$ denotes the means of the second set. Estimation k_i of the key is correct, if there is a marked difference between $m_{0i}^{'}$ and $m_{1i}^{'}$. This difference indicates the relationship between h_{ck} and some of the columns **T**. As in the previous case, this difference indicates the point at time, where the intrinsic values corresponding to h_{ck} are processed. In other moments, the differences between mean vectors are zero. The result of the attack is the matrix **R**, where each row corresponds to the difference between vectors $m_{0i}^{'}$ and $m_{1i}^{'}$ for one estimate key value. Equations to calculate **R** according to the difference of means method are given:

$$m_{1i,j}^{'} = \frac{1}{n_{1i}} \cdot \sum_{l=1}^{n} h_{l,i} \cdot t_{l,j}, \tag{2.34}$$

$$m_{0i,j}^{'} = \frac{1}{n_{0i}} \cdot \sum_{l=1}^{n} (1 - h_{l,i}) \cdot t_{l,j}, \tag{2.35}$$

$$n_{1,i} = \sum_{l=1}^{n} h_{l,i}, \tag{2.36}$$

$$n_{0i} = \sum_{l=1}^{n} (1 - h_{l,i}), \tag{2.37}$$

$$\mathbf{R} = \mathbf{M_1} - \mathbf{M_0}, \tag{2.38}$$

where n is the number of rows of the matrix **H**, in other words, this parameter represents the number of measured power consumptions. Fig. 2.7 depicts the result of the differential power analysis based on Difference of Means. The attack aims at the first byte of AES that was implemented on the smart card. Red color corresponds with the correct guess key byte and the blue color represents the wrong key byte guesses. It is clear, that the row of the matrix **R** with the highest peak corresponds with the correct key guess.

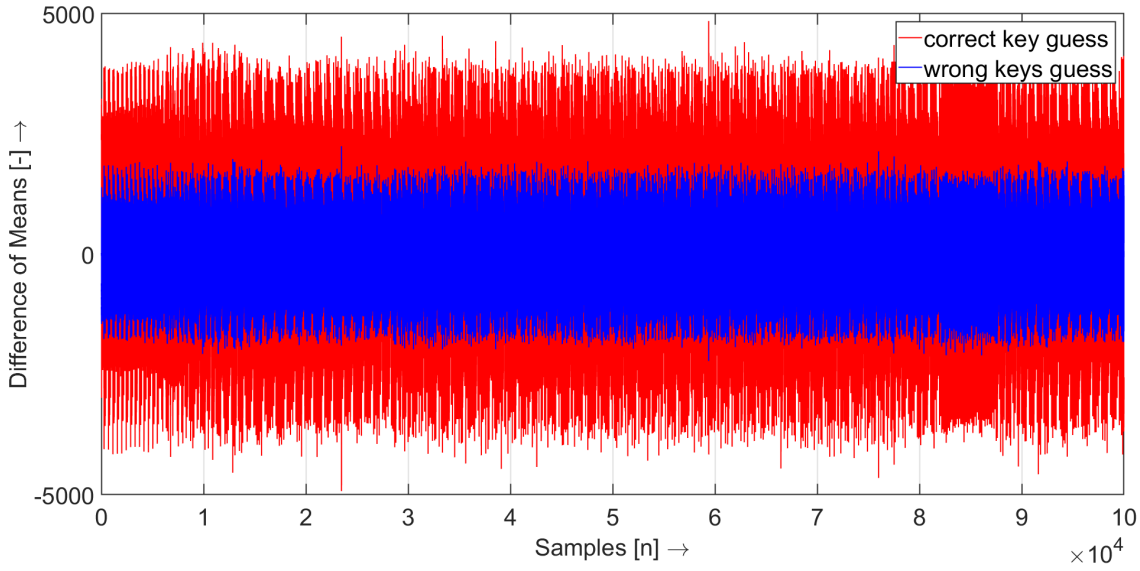


Fig. 2.7: Example of resulting matrix **R** for Difference of Means.

2.2.3 Power Simulation Models

A really crucial process is the mapping of hypothetical intrinsic values to hypothetical values of power consumption during the fourth step of the DPA attack (Sec. 2.2). For this purpose power simulation models are used. These models are mostly easy because the attacker does not have any detailed knowledge about the device under attack.

Hamming weight model (HW) is the basic power simulation model and it is usually used when the attacker does not have any information about the netlist of the device and about the processing data. Hamming weight equals to the number of non-zero symbols in a symbol sequence. The attacker expects that the power consumption is directly proportional to the number of non-zero bits in the processed data. The data values that were processed before or after this value are ignored. For this reason, the HW model is not suitable for simulation of the CMOS (Complementary Metal–Oxide–Semiconductor) circuits but the experimental results from practice show that the Hamming weight of currently processed data is dependent on the power consumption of CMOS circuits and it can be used. The Hamming weight model is depicted in Fig. 2.8 with blue color.

The second basic power simulation model is the **Hamming distance model** (HD). The attacker expects that the power consumption is directly proportional to the number of changed data values in the processed data. The HD model is very suited to describe the power consumption of data buses. The attacker can map the data which are transmitted via a data bus to the value of power consumption without the knowledge of the device netlist. Power consumption, which is caused by a change

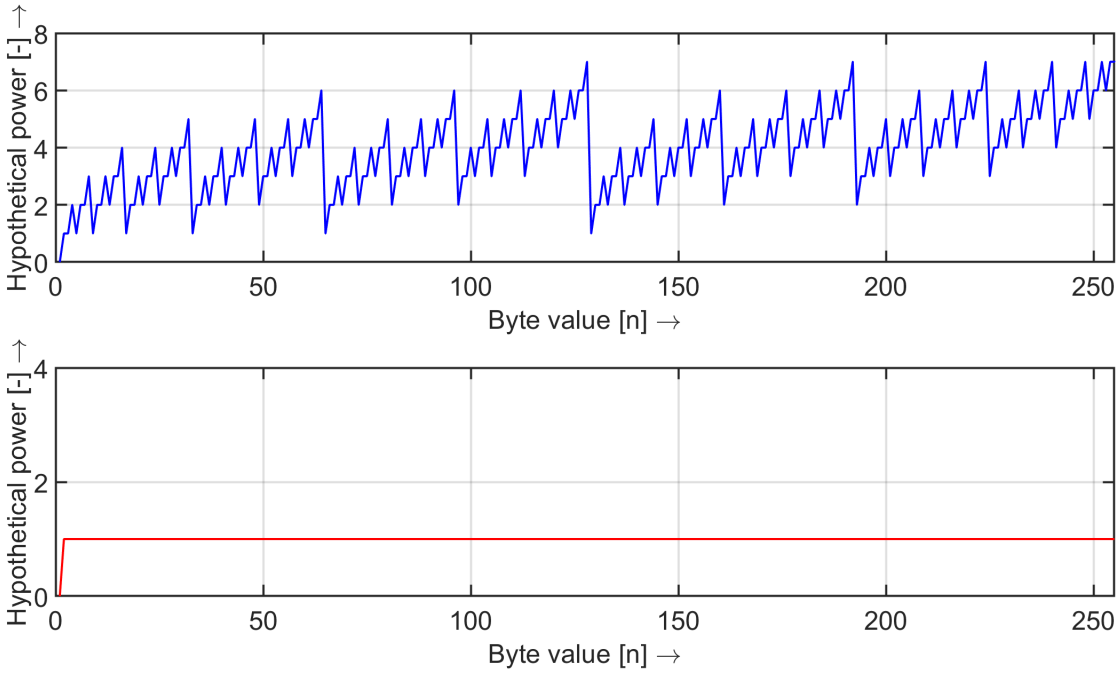


Fig. 2.8: Power consumption model of Hamming weight.

of the data bus value from v_0 to v_1 is proportional to $HD(v_0, v_1) = HW(v_0 \oplus v_1)$. Similarly, it can be applied to other buses such as the address buses.

The third well know power consumption model is the **Zero-vale model** (ZV). This model expects that for each intermediate value equal to zero, we set the hypothetical power consumption also to zero value. In all other cases, the model assumes the hypothetical power consumption equal to one. It is clear, that the attacker will need more power traces to realize a successful attack because only one intermediate value leakages information. This model is typically used to attack the hardware implementation of a cryptographic algorithm that utilized the FPGA (Field-Programmable Gate Array) platform. The ZV model is depicted in Fig. 2.8 with red color. Table 2.3 summarizes the most common applications of power consumption models in power analysis attacks.

Tab. 2.3: Application of power consumption model in power analysis attacks.

Name	Calculation in DPA	Application
HW model	$h_{i,j} = HW(S(d_i \oplus k_j))$	Software implementation (smart card)
HD model	$h_{i,j} = HD((d_i \oplus k_j), S(d_i \oplus k_j))$	CMOS microcontrollers
ZV model	$h_{i,j} = ZV(S(d_i \oplus k_j))$	Hardware implementation FPGA

2.3 Countermeasure Methods

PA attacks can be prevented by means of countermeasure techniques. The goal of every countermeasure is to make the power consumption of a cryptographic device independent of intermediate values that are processed during its operation phase. Generally, countermeasure techniques are divided into two basic groups: *masking* [104, 99] and *hiding* [27]. In the masking approach, each intermediate value is concealed by a random mask. By contrast, *hiding* tries to break the link between the power consumption and the processed data values.

2.3.1 Hiding

Hiding utilizes two approaches to achieve the power consumption independent of the intermediate values and independent of the operations that are performed. The first approach is to build devices whose power consumption is random. In practice, it means that in each clock cycle a random amount of power is consumed. The second approach is to build devices whose power consumption is constant for all operations and for all data values. The result is that equal amounts of power are consumed in each clock cycle. In other words, the ideal goal of hiding is that the power consumption is perfectly random or constant. However, this goal can not be reached in practice but there are several methods how to get as close as possible to this goal. The methods utilized are divided into two groups. The first group of methods randomizes the power consumption by performing the operations of the executed cryptographic algorithms at different times. These methods affect only the time dimension of the power consumption. On the other hand, the second group of methods affects only the amplitude dimension of the power consumption. These methods directly change the power consumption characteristics of the performed operations. The main approach of hiding methods are depicted in Fig. 2.9 where power traces corresponding with Hamming weight 1 and 8 are ploted.

Time Dimension

In the second step of the DPA attacks, the power traces have to be correctly aligned. If this condition is not fulfilled, the attack requires significantly more power traces or does not work. Therefore, one can affect the time dimension to randomize the execution of the cryptographic algorithms. In other words, the cryptographic module performs the operations of the algorithms at different moments of time during each execution. This behaviour of a cryptographic algorithm makes the power consumption random. The more random the execution of an algorithm is, the more difficult is the realization of the DPA attack. The most commonly utilized techniques to randomize the algorithm execution are the random insertion of *dummy operations*

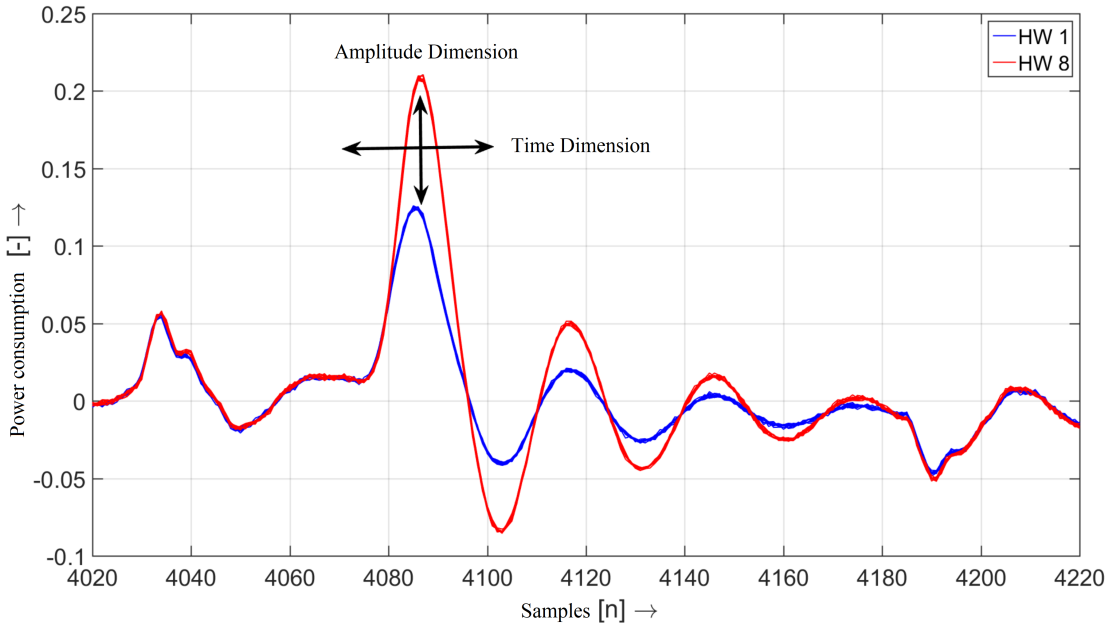


Fig. 2.9: Main groups of Hiding methods.

and the *shuffling of operations*. During the random insertion of dummy operations, the dummy operations are randomly inserted before, during, and after the execution of the cryptographic algorithm. Randomly generated number of dummy operations is utilized during each algorithm execution. It is crucial of this method that the total number of inserted operations is equal for all executions. It is clear that the more dummy operations are used, more positions vary in time dimension, the more random the power consumption is. On the other hand, the more dummy operations are inserted, the lower the throughput of the implementation is. In practice, a compromise has to be found for individual implementation of specific devices. An alternative to the method, one can use the operation shuffling. The main goal of this method is to randomly change the order of the execution sequence of operations that can be performed in an arbitrary (random) order. The typical example of this method is shuffling of S-box operations. This operation is realized in every round and look-ups for individual bytes are independent of each other. Therefore, the S-box operations for 16 bytes of the AES state can be performed in an arbitrary order. The main principle of shuffling AES S-box operations is depicted in Fig. 2.10. The figure presents also the place affected by the countermeasure in the power trace. The main advantage of shuffling is the fact, that this method does not affect the throughput of the algorithm (minimally). The disadvantage of shuffling lies in the fact that the number of operations that can be shuffled in a cryptographic algorithm is limited because not every operation is possible to be executed in an arbitrary order.

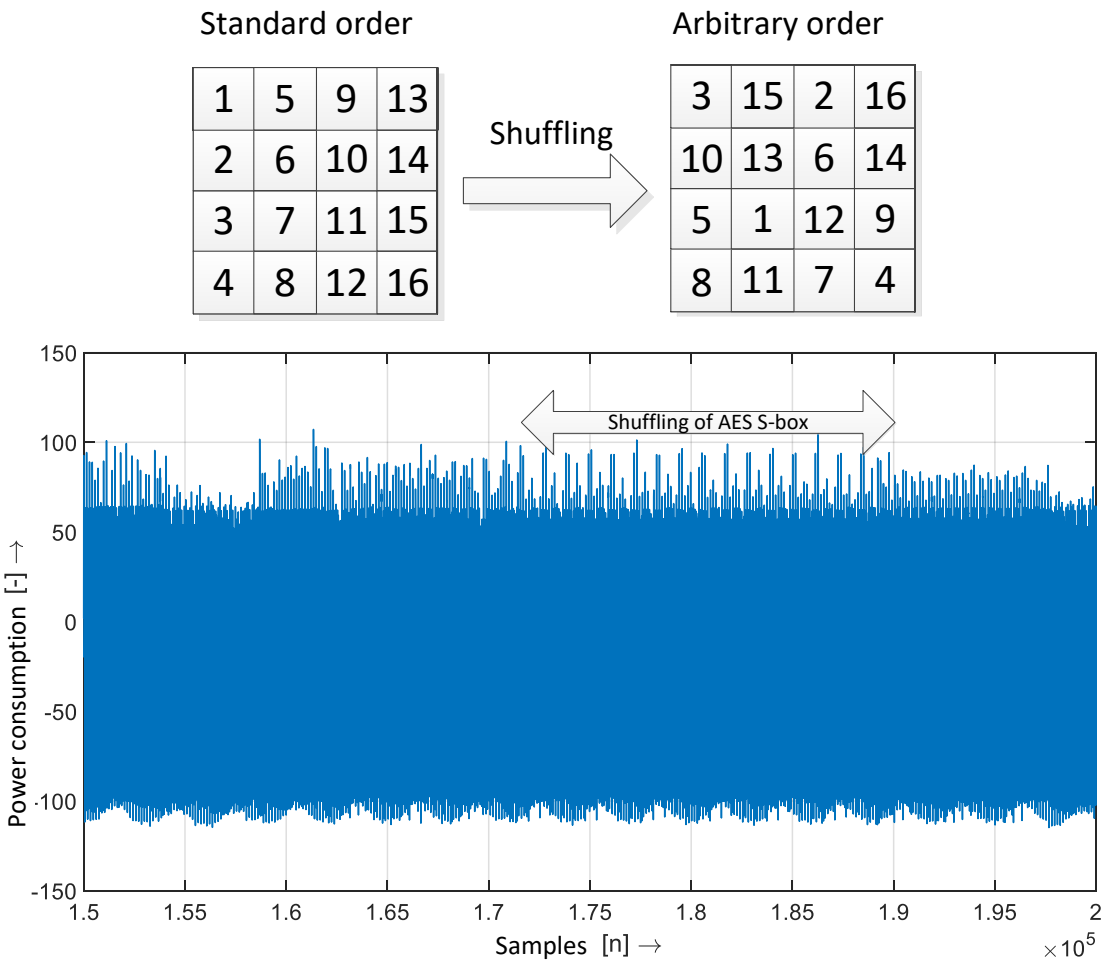


Fig. 2.10: Example of AES S-box shuffling.

Amplitude Dimension

These methods try to change directly the amplitude of the power consumption of the performed operations. The main goal is to obtain the power consumption of the device either equal or random. In practice, these techniques apply reduction of the signal-to-noise ratio (SNR) for operations executed. The SNR can be lowered by increasing the noise or by lowering the useful signal (that leakages the sensitive information). The simplest way to increase the noise of operations is to performer several independent operations in parallel. The important fact is to utilize independent operations. In practice, hardware platforms for cryptographic algorithms with a wide data-path are a more suitable solution for this method. The wider the data-path, the harder the attack is to realize. It is clear, because one point of a power trace (time interval) depends on more operations and more data computed. Another logical way to increase the noise is to utilize dedicated noise generators in the device.

2.3.2 Masking

Masking methods randomize the intermediate values that are processed by the cryptographic device in order to obtain the power consumption independent. The main advantage of this approach is that it can be implemented at the algorithm level and it is not necessary to affect the power consumption itself. Designers can implement masking in software or in hardware. In other words, the dependency of intermediate values on power consumption is broken by a masking method, therefore the attacker can not map the hypothetical intermediate value in the third step of the DPA attack. For these reasons, masking techniques have been extensively discussed in the scientific community. Numerous articles have been published that explain different types of masking schemes. In a masked implementation of a cryptographic algorithm, each intermediate value v is concealed by a random value m that is called mask: $v_m = v * m$.

The mask m is generated internally inside the cryptographic device during the execution of the algorithm. Moreover masks vary for each execution, therefore utilized masks are not known to the attacker. The operation $*$ is typically defined according to the cryptographic algorithm. Most often the operation is the Boolean exclusive-or (denoted as $\oplus$), modular addition (denoted as $+$), or modular multiplication(denoted as x). In case of modular addition and modular multiplication, the modulus is chosen according to the cryptographic algorithm. In practical implementations, the masks are applied to the plain text (plain text blinding) or the secret key (key whitening). It is clear that the implementation needs to be changed in order to process the masked intermediate values and in order to keep track of the masks. The result of the encryption realized with a masked implementation is logically also masked. Therefore, the masks have to be removed at the end of the encryption in order to obtain the cipher text. Masking scheme has to specify how all intermediate values are masked, are changed throughout the algorithm and are removed at the end of the algorithm. It is important that all intermediate values are masked all the time and this must be fulfilled for intermediate values that are calculated based on previous values. Table 2.4 summarizes the basic methods of masking.

Tab. 2.4: Masking methods according to the operation.

Name	Calculation of mask
Boolean masking	$v_m = v \oplus m$
Arithmetic modular addition masking	$v_m = v + m(mod\ n)$
Arithmetic modular multiplication masking	$v_m = v \times m(mod\ n)$

2.4 Attacks on Countermeasure Methods

In previous section 2.3, we have introduced two types of hiding countermeasures. In this section, we analyze the effectiveness of these countermeasures against DPA attacks by determining their effect. As a metric of comparision, we utilize $\rho_{ck,ct}$ that is the correlation between the hypothetical power consumption for the correct key hypothesis H_{ck} and the power consumption measured at the time sample ct. This is the time moment when the device processes the intermediate result at which the DPA attack is targeted. The metric $\rho_{ck,ct}$ determines the number of power traces that are needed to perform successful DPA attacks, see Fig. 2.5, that shows the point where the correct key hypothesis is separated from the others (25 power traces). Suppose, the power consumption of a point of a power trace can be modelled as the sum of the exploitable power consumption P_{exp}, the switching noise $P_{swNoise}$, the electronic noise $P_{elNoise}$, and the constant component $P_{constant}$ [78]:

$$P_{total} = P_{exp} + P_{swNoise} + P_{elNoise} + P_{constant}. \tag{2.39}$$

In this assumption, the correlation $\rho_{ck,ct}$ corresponds to $\rho(H_{ck}, P_{total})$. Masking methods cause that attacked intermediate values are processed at a different time in each power trace, i.e. ct is randomly distributed. The statistical distribution of ct depends on the way how the masking methods are implemented. If shuffling is implemented, ct is typically uniformly distributed, on the other hand, if the random insertion of dummy operations is implemented, ct is mostly binomial or uniformly distributed. In the cases, where these methods are combined, the resulting distribution of ct is the superposition of the corresponding distributions. Independent of the shape of the ct distribution, we denote the maximum of this distribution by $\hat{p}$ and power consumption that is located at this position by $\hat{P}_{total}$. For DPA attacks that target at misaligned power traces, the correlation for the correct key hypothesis can be calculated as follows [78]:

$$\rho(H_{ck}, \hat{P}_{total}) = \rho(H_{ck}, P_{total}) \cdot \hat{p} \sqrt{\frac{Var(P_{total})}{Var(\hat{P_{total}})}}. \tag{2.40}$$

For countermeasures that affect the amplitude in order to lower the SNR of the operations processing the attacked intermediate result reduces the correlation for the correct hypotheses as follows:

$$\rho(H_{ck}, P_{total}) = \frac{\rho(H_{ck}, P_{exp})}{\sqrt{1 + \frac{1}{SNR}}}. \tag{2.41}$$

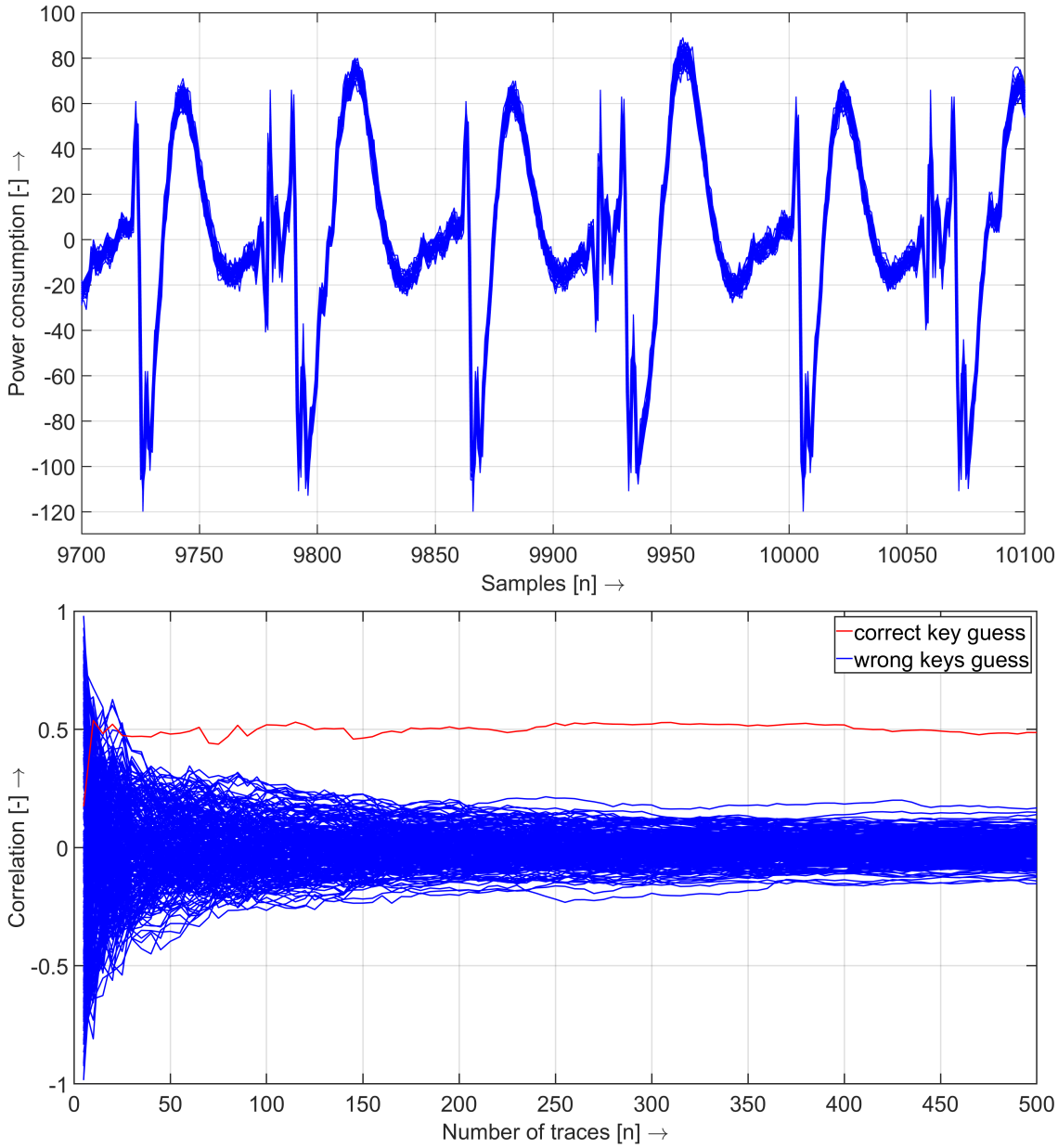


Fig. 2.11: Comparison of DPA attack efficiency result based on alignment traces.

2.4.1 Attack on Hiding

The following text describes the comparison of the efficiency of DPA attacks based on the metric $\rho(H_{ck}, P_{total})$ and the number of power traces. The efficiency of DPA attack based on power traces that are correctly aligned is depicted in Fig.2.11.

It is obvious that in this scenario the attacker needs 25 power traces to reveal the first byte of secret key. On other hand, the efficiency of DPA attacks based on power traces that are masked in time domain is depicted in Fig.2.12. In this scenario, the power traces are misaligned by a maximum of 30 samples. The misalignment

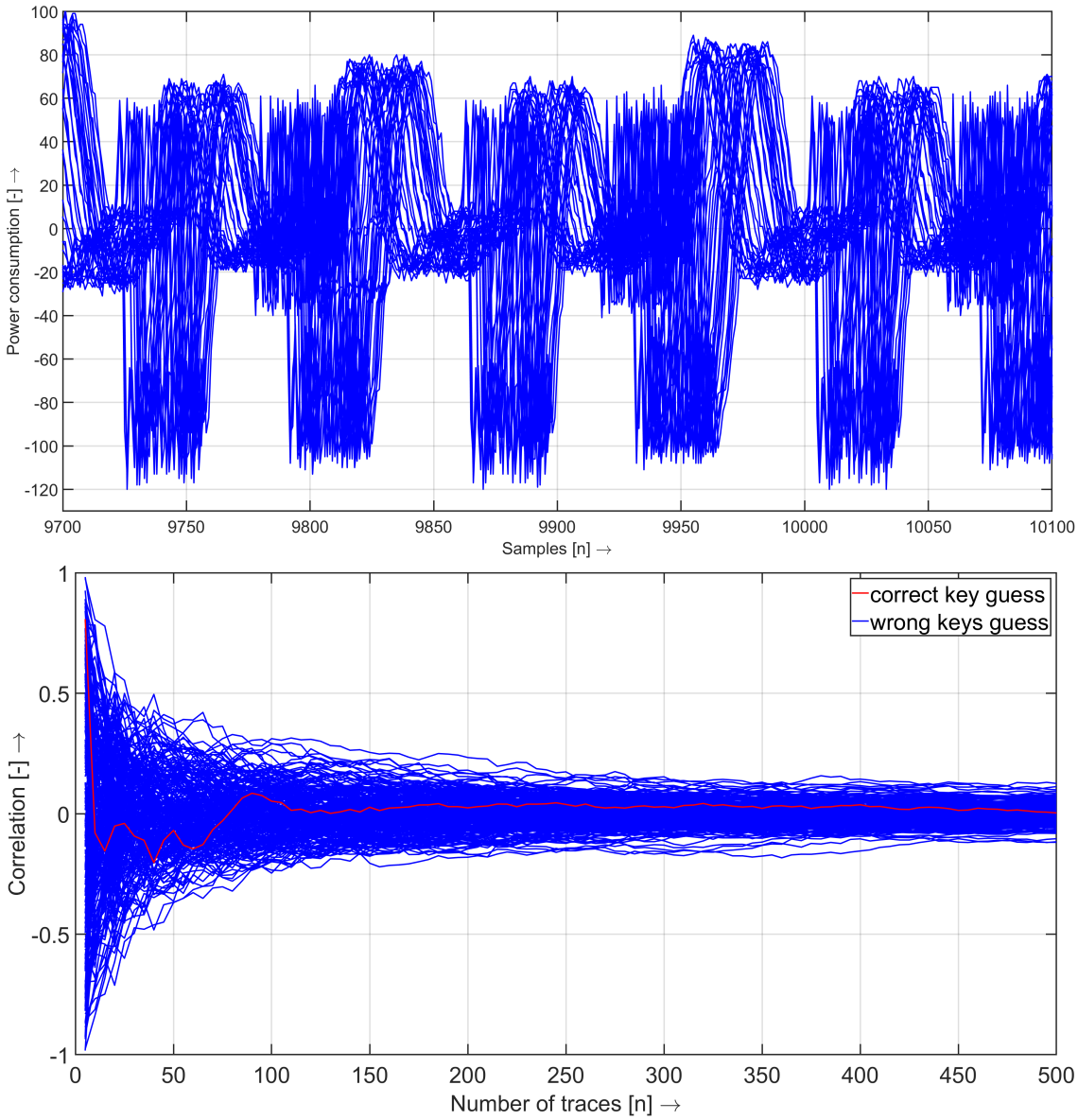


Fig. 2.12: Comparison of DPA attack efficiency result based on misalignment traces.

is uniformly distributed. In this scenario, the attacker can not reveal the value of the first secret key byte even with the use of maximum 500 power traces. In practice, the attacker realizes the visual inspection of the power traces measured and can align the traces. If the alignment is successful, the correct key hypothesis leads to the correlation and correct key revelation. The alignment of power traces is usually done based on pattern matching. This means that a part of the first power trace is selected as a pattern and the attacker tries to find this pattern in all other power traces. In the last step, the power traces are aligned based on the identified individual position of patterns.

In the last elementary comparison scenario, the signal-to-noise ratio is equal to 5 for the power traces measured. The efficiency of DPA attacks based on power traces with noise is depicted in Fig.2.13. It is obvious that the attacker needs many more power traces in order to reveal the secret key, 300 power traces were necessary. Moreover, the correlation coefficient is lower than we estimated in equation 2.41. The correlation coefficient is about 0.2.

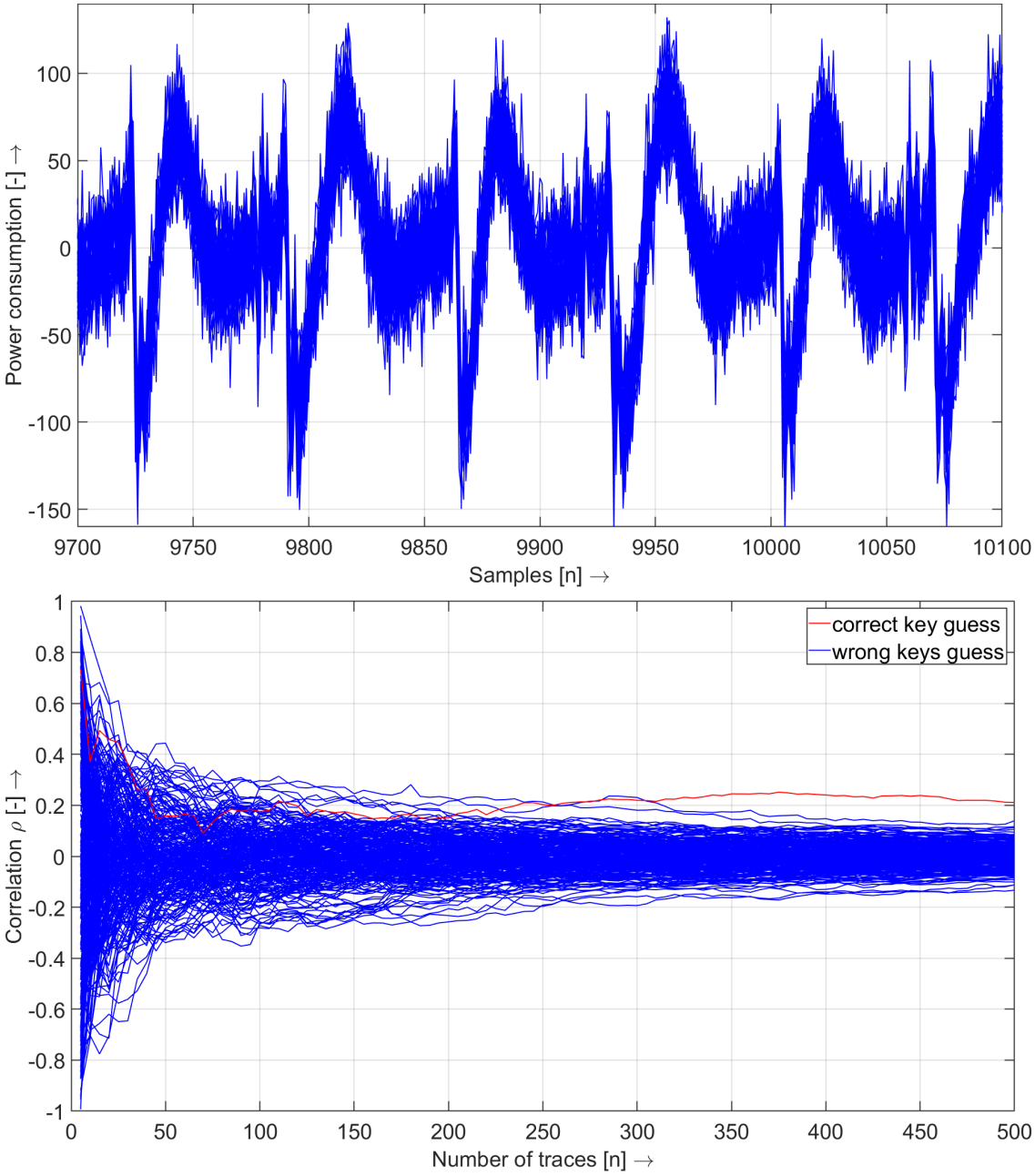


Fig. 2.13: Comparison of DPA attack efficiency based on traces with $SNR = 5$.

2.4.2 Attack on Masking

The use of masking schemes to counteract power analysis attacks is popular because it can be implemented in software. Therefore it is not necessary to alter power consumption characteristics themselves. Many researchers have studied the security of masking schemes and their implementations and it has turned out that every masking scheme can be attacked and the designers should combine masking with hiding techniques. In the following text, we discuss different types of power analysis attacks on masking schemes.

Generally, higher-order DPA attacks exploit the joint leakage of several intermediate values. In practice, the implementations of masking schemes conceal several intermediate values by the same mask, typically for computational reasons. Therefore, it is not necessary to study higher-order DPA attacks in general, but it is sufficient to concentrate on higher-order DPA attacks that exploit the leakage related to two intermediate values. Therefore, the following text is focused only on second-order DPA attacks. Second-order DPA attacks exploit the leakage of two intermediate values that are related to the same mask. In a general case, this leakage cannot be exploited directly because the two intermediate values often occur in different time intervals. Therefore, it is necessary to preprocess the power traces in order to obtain power consumption values that depend on both intermediate values.

There are three cases of preprocesing of power traces:

- In the first case, the targeted intermediate values occur in different clock cycles. In this case, the preprocessing has to combine two points within a power trace, it is a typicall example for software implementations of masking schemes.
- In the second case, the targeted intermediate values occur within one clock cycle thus preprocessing has to combine single points in the trace.
- The third case represents a situation where the targeted intermediate values occur within a clock cycle and the power consumption characteristics allow exploiting the leakage directly. The last two cases are typical for hardware implementations.

It is really interesting, that higher-order DPA attacks were already mentioned in Kocher's [58]: "Of particular importance are high-order DPA functions that combine multiple samples from within a trace." Naturally, several researchers have tried to implement attacks based on this very brief sketch and Messerges was the first researcher to successfully report on a second-order DPA attack in [100]. However, Oswald describes second-order DPA attacks from a practical point of view including the preprocessing step in work [110].

The attack described in [100], targets the exclusive-or (short: XOR) operation of a byte of the key and a byte of masked data. It is assumed that in the implementation under attack, the mask is generated and subsequently exclusive-ored with the data prior to the exclusive-or operation that involves the key byte:

$$t = 1 : m = rand() \text{ (generate mask-byte)}, \tag{2.42}$$

$$t = 2 : x = p \oplus m \text{ (XOR mask with plain text-byte)}, \tag{2.43}$$

$$t = 3 : y = x \oplus k \text{ (XOR masked plain text with key-byte)}. \tag{2.44}$$

In the attack, the point in the power trace $s_j[t = 1]$ that corresponds to the time when the mask is generated (eq. 2.42) is subtracted from the point in the power trace $s_j[t = 3]$ that corresponds to the time when the masked data is XORed with the key byte (eq. 2.43). The joint distribution of these two power samples allows to derive the keybyte bit by bit. For every bit in the plain text byte the adversary calculates the mean values $\bar{S_0} = \sum_j |s_j[t = 1] - s_j[t = 3]|$ (if the plain text bit is 0) and $\bar{S_1} = \sum_j |s_j[t = 1] - s_j[t = 3]|$ (if the plain text bit is 1). If $\bar{S_0} - \bar{S_1} > 1$ then the key bit is 1, otherwise it is zero. Example of the result for the first four bits of a second-order DPA attack is depicted in Fig. 2.14. In the attack, it is mandatory to use the absolute value of the differences, because otherwise the difference of means is 0 in both cases. In addition, it is necessary that the mean values of the power traces are roughly the same, otherwise the difference of means also does not lead to conclusive results.

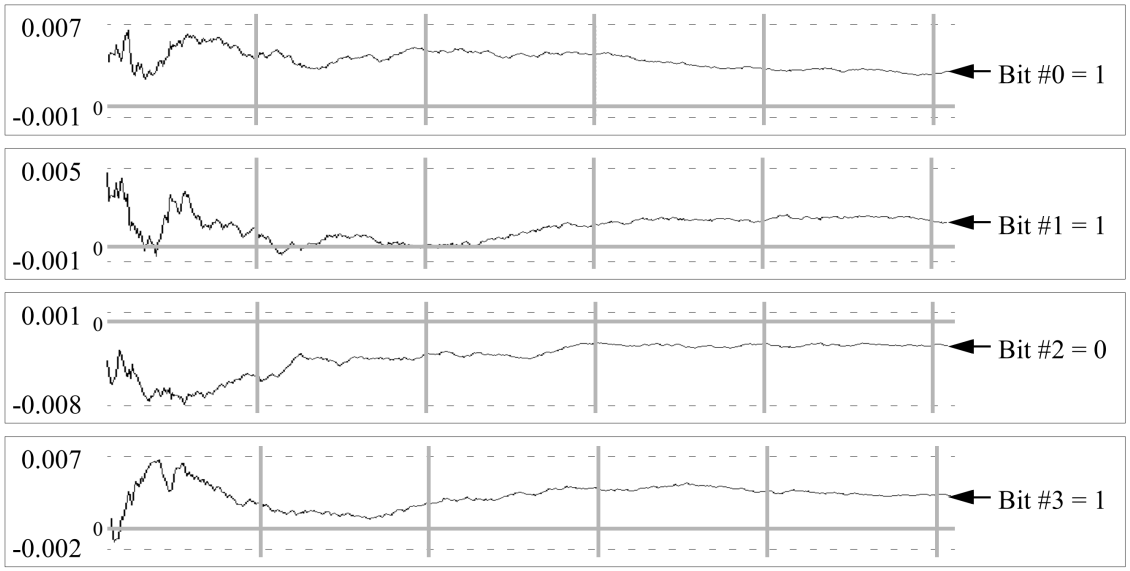


Fig. 2.14: Typical result of second-order DPA attack based Messerges approach [100].

In paper [110] Elizabeth Oswald introduced easy-to-implement second-order DPA (SODPA) attacks. The attacks are using the following assumption. Let a be a value $\in \{0,1\}^n$ and $C(a)$ denotes the power consumption of the value a. Then the power consumption C of the device at the time when a is processed is proportional to the Hamming-weight of the value $a : C(a) \approx HW(a)$. We assume that the instantaneous power consumption of the device under attack depends linearly on the HW of the processed data.

Observation - Let a and b are values $\in \{0,1\}^n$, let $\oplus$ denote the exclusive-or operation, and let $HW(x)$ denote the Hamming-weight of x. Then the following relation holds with probability one:

$$HW(a \oplus b) = |HW(a) - HW(b)| . \tag{2.45}$$

Consequently, we can correctly predict $|C(a) - C(b)|$ with $HW(a \oplus b)$ if $a, b \in \{0,1\}$. We can use this observation to mount second-order DPA attacks: In the first step, the adversary chooses a point in a power trace, subtracts it from the rest of the trace and takes the absolute value of the result. In the second step, the adversary tests for all keys whether the HW of the exclusive-or of the two intermediate values under attack correlates to the preprocessed power traces. Only for the correct key and for the correct point, a peak will occur in the power trace. From generally proposed attack we can mount it in the following way.

Elementary Example of SODPA

In this elementary example, we present the attack that is targeted on masked Key Addition of AES. In this example, the cryptographic module is a chip containing a hardware masked implementation of the AES algorithm [112, 146]. The masking method used was based on the combination of additive and multiplicative masking for masking the non-linear byte substitution operation (SubBytes). In general, the SubBytes operation is the most difficult part of the AES encryption algorithm to be masked. The reason is its non-linearity. The essence of the masking method used is in the conversion of the SubBytes inversion from the $GF(256)$ field (Galois Field) to the inversion in the $GF(4)$ field. The concrete specification of the hardware implementation of the non-linear byte substitution is not important to run the power analysis attacks. On the other hand, the knowledge of intermediate values processed by the cryptographic module corresponding to the plaintext is essential. Data processing is described by the scheme in Fig. 2.15. To run the attack, the power traces corresponding to the whole process of encryption and decryption are provided, together with the corresponding plaintext. An example of a power trace is depicted in Fig. 2.16.

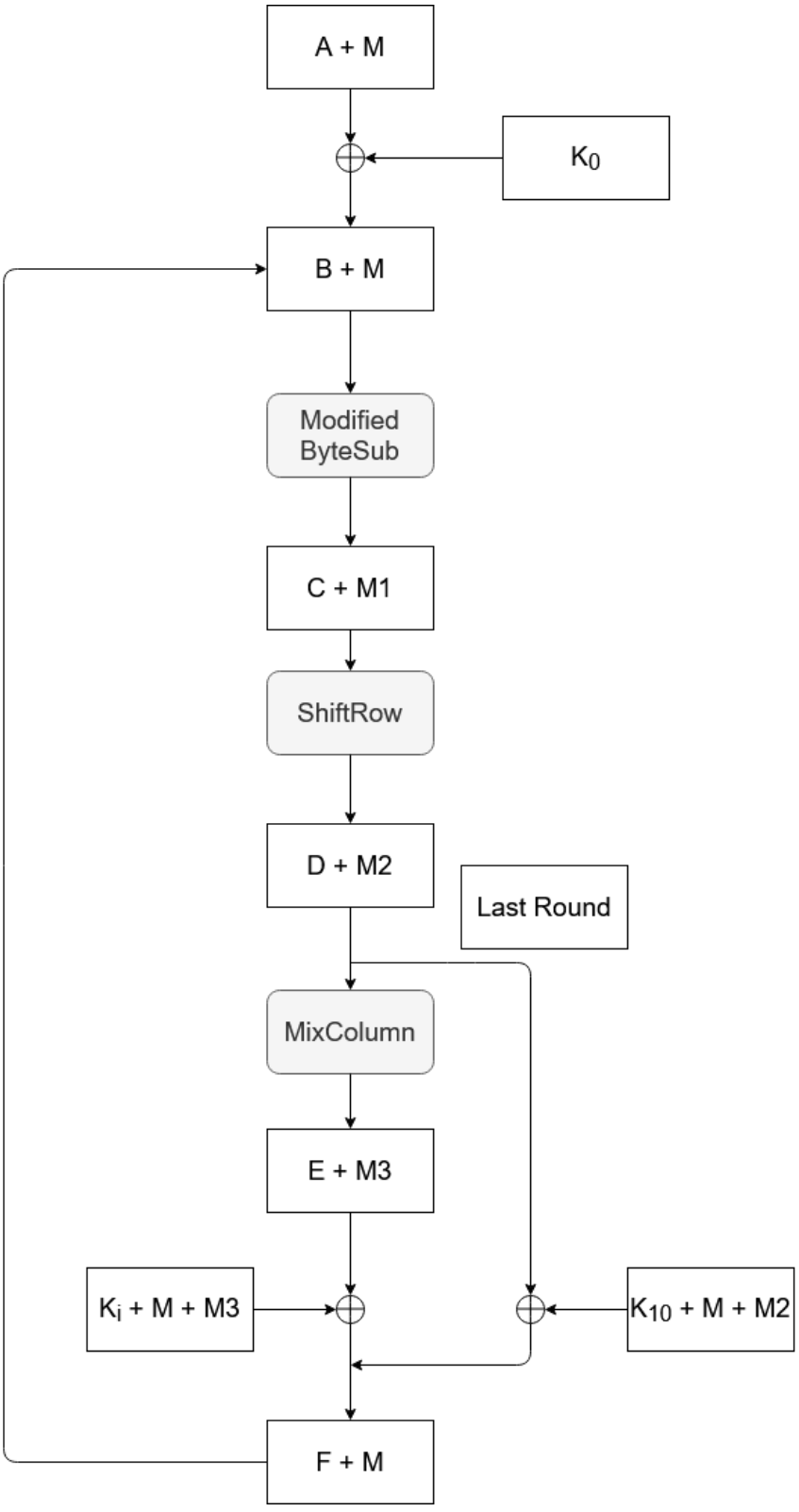


Fig. 2.15: Scheme of protected algorithm implemented [146].

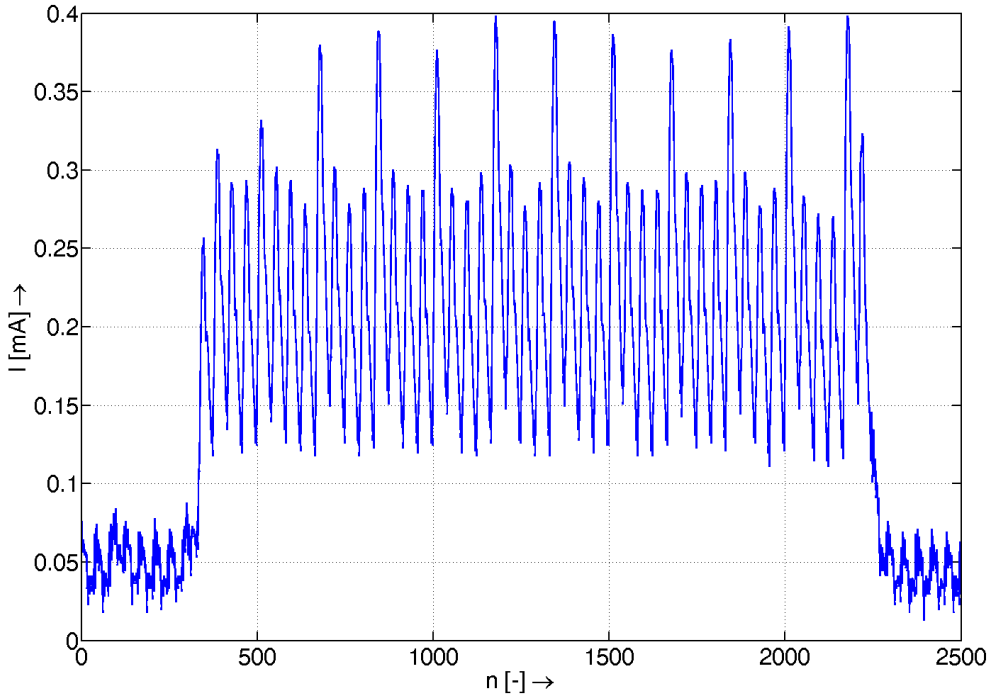


Fig. 2.16: Example of power trace hardware masked implementation.

Altogether, 150,000 power traces were provided to run the attack for the following encryption key: $K_{s1} = [42, 138, 236, 244, 69, 67, 231, 207, 141, 31, 115, 14, 106, 251, 199, 152]$. In the following text, the encryption key is denoted as $K_{s1} = \{k_1, k_2, \ldots, k_N\}$ for $0 \leq k_i \leq 255$, $1 \leq i \leq 16$ and $N = 16$ represents the key size. This scenario is similar to the one described by Messerges in [100]. In this attack, we assume that the plain text P is concealed with a random mask M : $P \oplus M$. During the key addition, the masked plain text is exclusive-ored with the key: $P \oplus M \oplus K$. The manipulation of M and the computation of $P \oplus M \oplus K$ occur some time during the (initial) phase of the algorithm.

For the attack, we assume that we have recorded the power trace of the initial phase of the algorithm. We use the value of the mask M and the value $P \oplus M \oplus K$ of the key addition in our attack. In the first step, we locate the sequence of key addition operations (to save computation time). We make an educated guess for the time frame when M and $P \oplus M \oplus K$ are computed. Based on the first analysis we choose a specific time interval. In the second step, we predict $|C(M) - C(P \oplus M \oplus K)|$ with $HW(P \oplus K)$ and perform a standard DPA attack. Note that the value of the mask is unknown. For the prediction we used one byte of the plain text and we guess one byte of the secret key utilizing the ZV model because the implementation is realized in hardware.

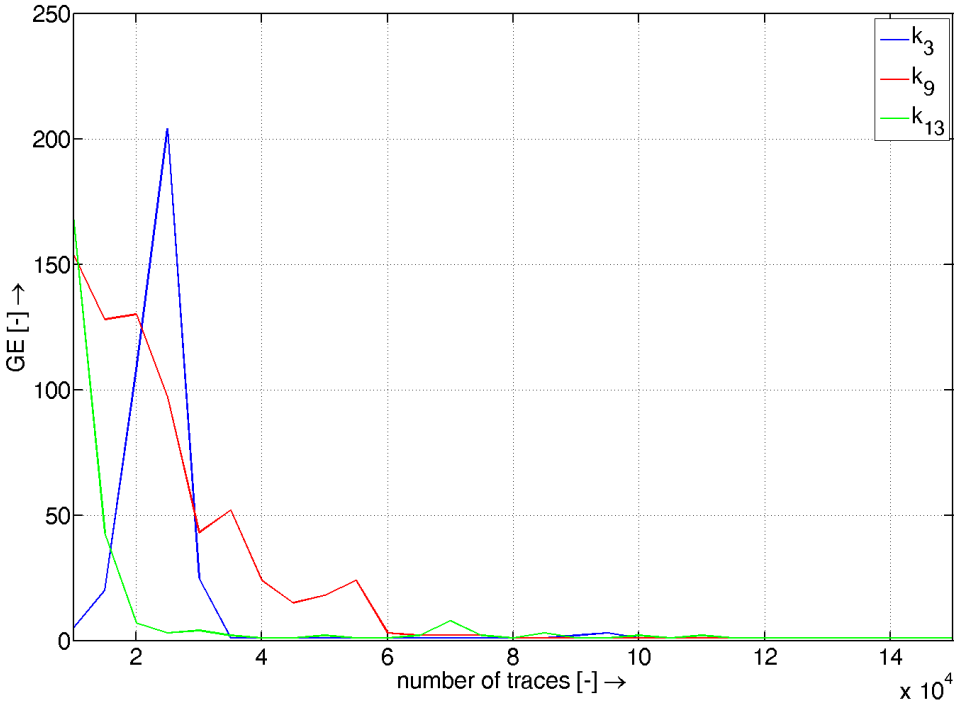


Fig. 2.17: Result of GE of SODPA for k_1, k_9 and k_13.

Using the SODPA attack based on the correlation coefficient, aimed at two intermediate results of mask loading and round key addition, using the ZV model and $150,000$ power traces, the key is estimated as:

$$K_e = [42, 138, 236, 244, 69, 67, 231, 207, 141, 31, 193, 14, 106, 251, 199, 202].$$

The attack reveals 14 bytes of the secret key and the wrongly estimated keys k_{11} and k_{16} were on positions 10 and 8 (marked with red color). The GE value is estimated to be 18 attempts which are required by the attacker to obtain the complete key in an ideal case. To provide a comparison to previous results, the GE value plot for key bytes k_2, k_8, k_{14} is shown in fig. 2.17.

The result of SODPA and the dependency of the correlation coefficient on the number of power traces for the second byte is shown in fig. 2.18 and 2.19 successively, the correct key estimate is marked red. From the correlations plot, the sufficient distance for the correct key estimate is clear. For most of correctly estimated keys, the distance was sufficient. Based on the obtained result, it is clear that the SODPA attacks are really easy to realize. However the attack needs much more computing resources and many more power traces to reveal the secret key. In this case, even the $150,000$ power traces were not enough to obtain the whole secret key.

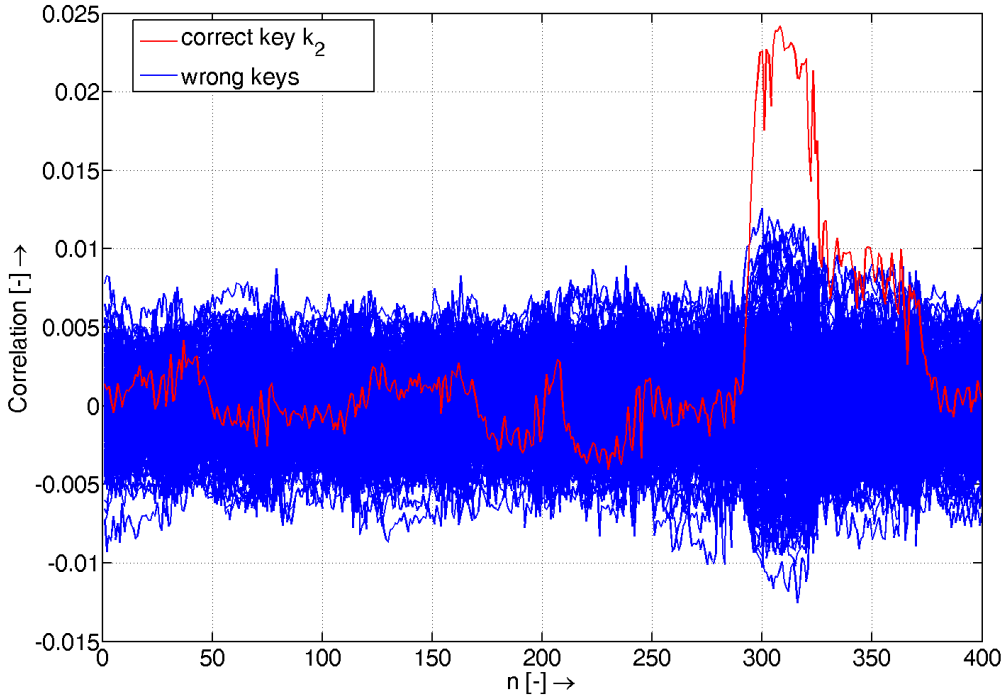


Fig. 2.18: Result of SODPA attack based on correlation coefficient.

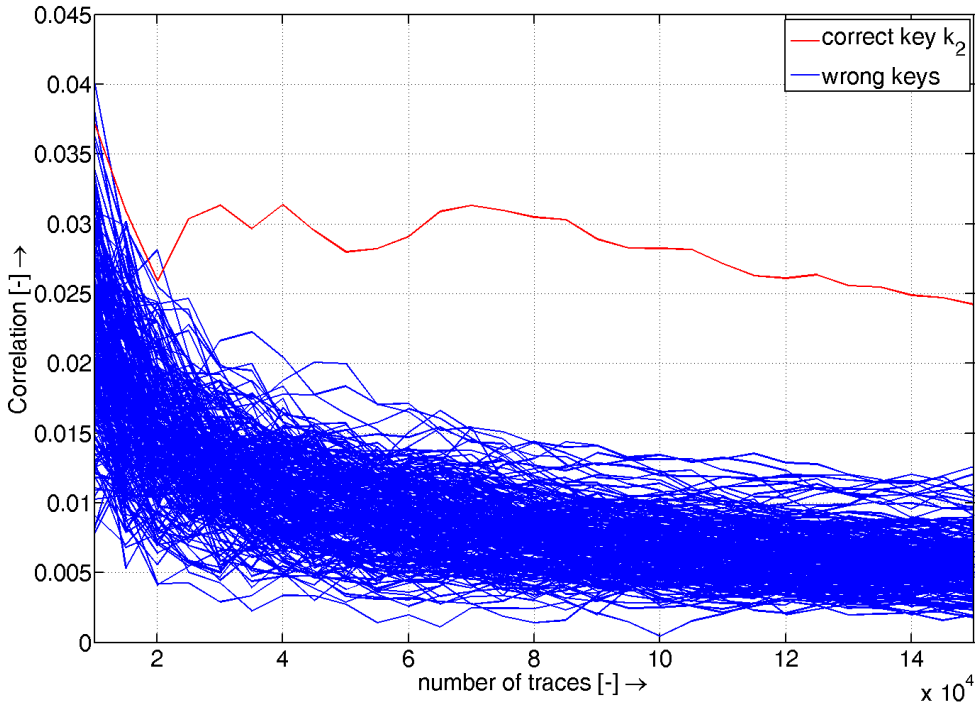


Fig. 2.19: Size of the correlation depending on the number of power traces.

3 Study of Protected Implementations

In this section, we describe masking and hiding countermeasure techniques including the power analysis from a practical point of view. More precisely, Boolean masking and shuffling of crucial operations of AES are attending in our scientific and educational text as well as a short current state description. As an elementary example, we bring in to play the DPA Contest because it is wordwide known and freely available (meant power traces including plain and cipher text). Therefore, the reader can verify himself the obtained results that is the best way to understand the explained issues.

The first version of the DPA Contest was created by a research group of the French technical university Telecom ParisTech already in 2008. The main goal was to provide an objective way of comparison of various power analysis attacks. Currently, the fourth version of DPA Contest is open[1]. Instinctively, an independent comparison of power analysis attacks is problematic due to several reasons. We attempt to categorize dominant causes:

- dissimilar measuring devices,
- non-identical implementations of the cryptographic algorithm,
- various cryptographic devices or various metrics of attack comparison.

DPA Contest is trying to minimize these factors by providing power traces of a cryptographic device, that are measured under predetermined conditions. Moreover, authors of DPA Contest provide software tools that are used for attack implementation and result evaluation. The main tool is called *AttackWrapper* that controls the implemented attack, reads power traces and allows communication with other tools. One can use any programming language C, C ++, Python, Perl, R-language or Matlab to implement the attack. Other important tool called *ComputeResults* is used to analyze the obtained results from *AttackWrapper*. In this way, independent researchers have an opportunity to develop their own attacks and subsequently they can make a comparison with other attacks.

This chapter contains various results from selected author's publications such as [151, 157, 93, 70, 83] that are focused on power analysis of protected implementations. The amended version of the text below is a part of the author's papers in journals with an impact factor, namely, Computers & Security [83], IET Information Security [70] and Radioengineering [93]. The rest of the chapter contains information from IEE conference papers [151, 157].

[1] http://www.dpacontest.org/v4/index.php

3.1 DPA Contest V4.1

In order to protect software implementations of AES, *masking* [20, 38] and *shuffling* [46] are the most studied and used techniques. *Shuffling* represents a simple *hiding* countermeasure which randomizes the operation order. In general, the *SubBytes* operation is the most difficult part of the AES algorithm to mask due to its non-linearity — we refer interested readers to the technical report [111] to delve into methods that deal with this aspect of AES countermeasures. On the other hand, various masking methods of the AES algorithm have been proposed [37, 112, 18, 110]. In any case, implementation of countermeasures brings overheads in terms of memory and time, hence causing that researchers start to look for lightweight approaches. One of such lightweight countermeasures is the Rotating Sbox Masking (RSM), which is a type of Low-Entropy Masking Scheme [106, 144, 12]. The main idea behind RSM consists in the usage of precomputed lookup tables [119] and, at the same time, reducing the overhead by carefully choosing the limited mask set [11]. This essentially allows to reuse S-boxes while reducing the computation of mask compensation since only 16 possible masks are applied. The set of chosen masks can be a public parameter; however, this set should be shifted by a random *offset* before each encryption. We refer interested readers to works [106, 12, 39] where more details of RSM and its security analysis are provided. RSM has been studied by researchers worldwide under the framework of DPA Contest V4.1 [40].

In DPA Contest V4.1 a smart card based on the chip Atmel ATMega-163 is used as a cryptographic device under attack. Masked algorithm AES-256 is implemented and for masking a RSM is utilized [106, 102]. The method provides simple implementation and low computational cost and, in addition, it should be resistant versus first order power analysis attacks. Note that the goal of attacks in DPA Contest V4.1 is to reveal the first 128 bits of the secret key stored, therefore we focus only on the first round of AES-256 in the following text. For all power traces measured, plain texts, secret key and the corresponding encrypted texts are provided by default. Power traces were measured using the following testbed: electromagnetic near-field probe Langer RF-U 5-2, digital scope Lecroy Waverunner 6100A, preamplifier Langer PA303 and Regulated Power Supply Agilent E3631A [40]. For DPA Contest V4.1, $30,000$ power traces[2] are available. An example of the power trace is shown in Fig. 3.1.

[2] At the beginning of the contest, $100,000$ power traces were available.

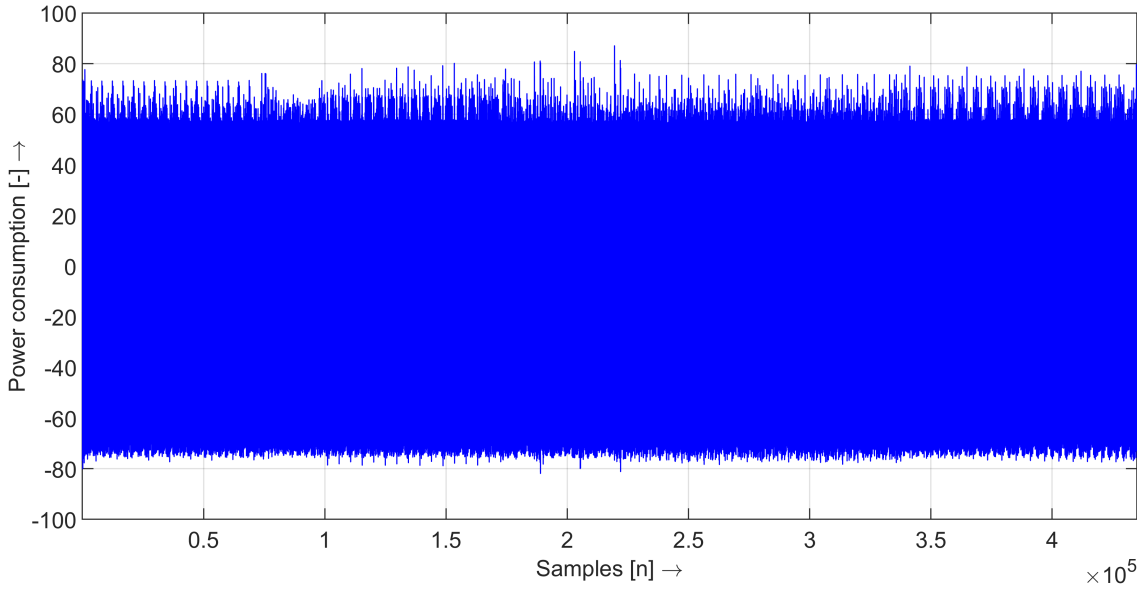


Fig. 3.1: Example of one power traces for DPA Contest V4.1.

3.1.1 Description of Countermeasures Implementation

The RSM utilized 16 masks denoted as $m = \{m_0, m_1, \ldots, m_{15}\}$, which are publicly available in order to mask the intermediate value. These masks are chosen in a way that the leaked information about the currently processed data is minimal [106]. At the beginning of each encryption process, a random value denoted as a secret offset is drawn from 0 to 15. This value is unknown to the attacker and the mask set is rotating according to this value. In the next step of the algorithm, for each i-th byte of plain text p_i and mask $m_{i+\text{offset}}$ the XOR (exclusive OR) operation is applied according to the following equation (`Plaintext Blinding`):

$$\begin{Bmatrix} p_0 \ldots p_{12} \\ p_1 \ldots p_{13} \\ p_2 \ldots p_{14} \\ p_3 \ldots p_{15} \end{Bmatrix} \oplus \begin{Bmatrix} m_{0+\text{offset}} \ldots m_{12+\text{offset}} \\ m_{1+\text{offset}} \ldots m_{13+\text{offset}} \\ m_{2+\text{offset}} \ldots m_{14+\text{offset}} \\ m_{3+\text{offset}} \ldots m_{15+\text{offset}} \end{Bmatrix} = \begin{Bmatrix} s_0 \ldots s_{12} \\ s_1 \ldots s_{13} \\ s_2 \ldots s_{14} \\ s_3 \ldots s_{15} \end{Bmatrix} = State. \tag{3.1}$$

where the value $i+\text{offset}$ is calculated modulo 16. Substitution `SubBytes` is replaced with sixteen masked $\texttt{MaskedSubBytes}_i(s_j) = \texttt{SubBytes}(s_j \oplus m_i) \oplus m_{i+1}$, where $i \in \{1, 2, \ldots, 16\}$. After the linear operation of the AES algorithm, additional operations

follow to ensure that the *State* entering into the next round is equal to:

$$\left(\texttt{MC} \circ \texttt{SR} \circ \texttt{SBox} \circ \begin{Bmatrix} s_0 \oplus m_{0+\text{offset}+r} \oplus O_0^r \dots \\ s_1 \oplus m_{0+\text{offset}+r} \oplus O_1^r \dots \\ s_2 \oplus m_{0+\text{offset}+r} \oplus O_2^r \dots \\ s_3 \oplus m_{0+\text{offset}+r} \oplus O_3^r \dots \end{Bmatrix} \right) \oplus \\ \oplus \begin{Bmatrix} m_{0+\text{offset}+r+1} \dots m_{12+\text{offset}+r+1} \\ m_{1+\text{offset}+r+1} \dots m_{13+\text{offset}+r+1} \\ m_{2+\text{offset}+r+1} \dots m_{14+\text{offset}+r+1} \\ m_{3+\text{offset}+r+1} \dots m_{15+\text{offset}+r+1} \end{Bmatrix} \tag{3.2}$$

where $r \in \{1, \dots, 9\}$ denotes the number of the round (AES-128) and O_i^r represents the $(i+1)$-th byte of the r-th subkey. `SR` and `MC` represent linear transformations `ShiftRow` and `MixColumns`. The masking scheme for the last encryption round ($r = 10$ for AES-128) ensures that the output equals to:

$$\left(\texttt{SR} \circ \texttt{SBox} \circ \begin{Bmatrix} s_0 \oplus m_{0+\text{offset}+r} \oplus O_0^r \dots \\ s_1 \oplus m_{0+\text{offset}+r} \oplus O_1^r \dots \\ s_2 \oplus m_{0+\text{offset}+r} \oplus O_2^r \dots \\ s_3 \oplus m_{0+\text{offset}+r} \oplus O_3^r \dots \end{Bmatrix} \right) \oplus \begin{Bmatrix} m_{0+\text{offset}+r+1} \dots m_{12+\text{offset}+r+1} \\ m_{1+\text{offset}+r+1} \dots m_{13+\text{offset}+r+1} \\ m_{2+\text{offset}+r+1} \dots m_{14+\text{offset}+r+1} \\ m_{3+\text{offset}+r+1} \dots m_{15+\text{offset}+r+1} \end{Bmatrix}. \tag{3.3}$$

The last mask is removed by the XOR operation. The principle of the masked RSM-AES algorithm is shown in Alg. 2, where a basic knowledge of AES algorithm is assumed (terms such as round, state, operation etc.) [1]. The original unprotected version of the AES can be obtained by removing the lines marked using blue color.

3.1.2 Power Analysis Realized

In this section, we investigate the security of V4.1 implementation in practice. We used the basic power side-channel techniques (described in Sec. 2) in the same way as a potential adversary would do, and tried to analyse the sensitive information and answer the question if the implementation is secure. Our analysis, focused on exploiting the first-order leakage, discovered some lacks. The crucial one showed that an adversary can prepare templates in order to reveal the mask and mount a standard DPA attack aimed at the S-box output in order to recover the whole secret key. We tested this observation on a public dataset, and implemented a successful attack that revealed the secret key. In the final step of our analysis, we submitted the attack to the DPA Contest to compare the results.

Algorithm 2 Pseudo code of RSM in DPA Contest V4.1.

Require: Plain text X; 16 bytes X_i, where $i \in \langle 0, 15\rangle$, Key schedule; 15 128-bit constants $\mathsf{RoundKey}[r]$, where $r \in \langle 0, 14\rangle$.

Ensure: Cipher text X; 16 bytes X_i, where $i \in \langle 0, 15\rangle$.

1. % Draw a random offset, uniformly in [0,15].
2. $X = X \oplus \mathsf{Mask}_{\mathsf{offset}}$ % Plain text blinding
3. % All rounds
4. **for** $r \in \langle 0, 12\rangle$ **do**
5. $\quad X = X \oplus \mathsf{RoundKey}[r]$
6. $\quad$ **for** $i \in \langle 0, 15\rangle$ **do**
7. $\quad\quad X_i = \mathsf{MaskedSubBytes}_{\mathsf{offset}+i+r}(X_i)$
8. $\quad$ **end for**
9. $\quad X = \mathsf{ShiftRows}(X)$
10. $\quad X = \mathsf{MixColumns}(X)$
11. $\quad X = X \oplus \mathsf{MaskCompensation}_{\mathsf{offset}+1+r}$
12. **end for**
13. % Last round
14. $X = X \oplus \mathsf{RoundKey}[13]$
15. **for** $i \in \langle 0, 15\rangle$ **do**
16. $\quad X_i = \mathsf{MaskedSubBytes}_{\mathsf{offset}+13+r}(X_i)$
17. **end for**
18. $X = \mathsf{ShiftRows}(X)$
19. $X = X \oplus \mathsf{RoundKey}[14]$
20. % Cipher text damasking
21. $X = X \oplus \mathsf{MaskCompensationLastRound}_{\mathsf{offset}+14}$

The power analysis was carried out in the following steps:

- Localization of interesting points for crucial operation based on correlation coefficients.
- Key observation of possible lacks.
- Attack proposal and implementation (experimental verification on public dataset).
- Attack submission and result comparison.

In order to locate points of interest, we utilized a differential power analysis attack depending on the offset value for important operations of the algorithm: `Plain text Blinding` and `Masked SubBytes`[3]. In other words, we chose the output of above mentioned operations as the intermediate value of the DPA attack based

[3]Other methods e.g. NICV could be performed to localize points of interest and the results would be identical.

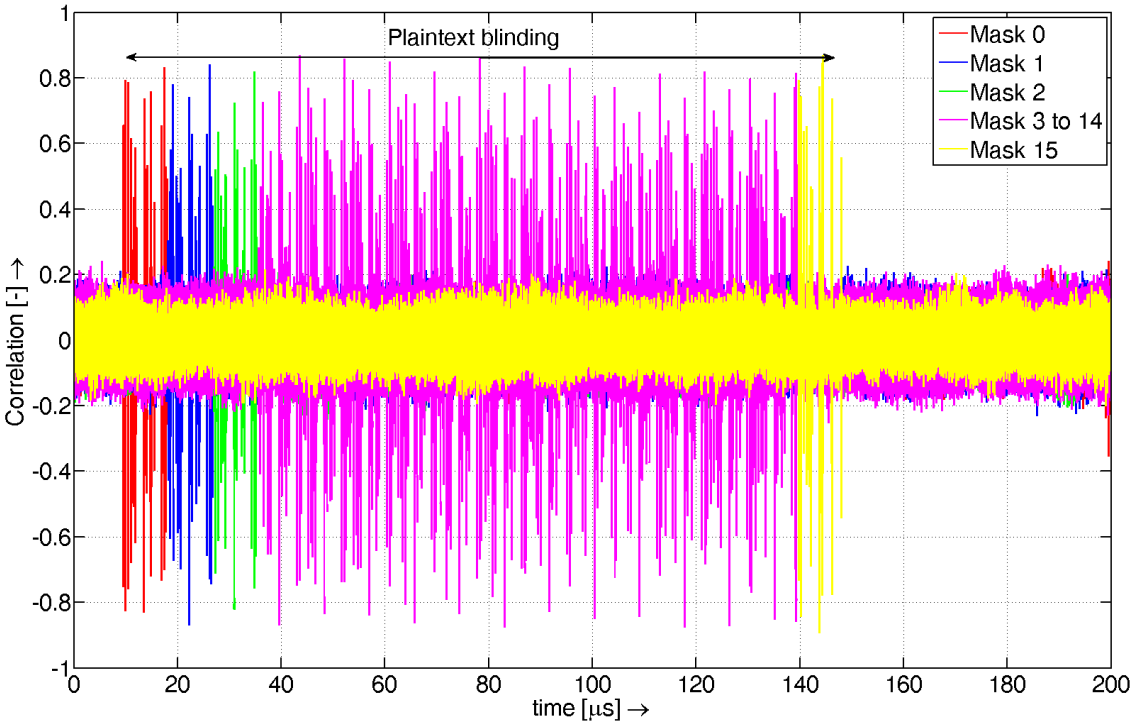


Fig. 3.2: Result of CPA for operation `Plaintext blinding`.

on correlation coefficient (in the following test denoted as CPA) [78, 80]. The result of the CPA analysis for `Plain text blinding` is shown in Fig. 3.2.

We can observe that this operation strongly leaks information about the mask values, in other words, in a power trace, a lot of points exist that carry information about the secret offset that must be kept secret. The algorithm for selecting the points of interest localized and stored the 3 highest correlated points for every mask value. Together 48 interesting points were chosen from the realized CPA. An interesting remark is that selected points of individual masks loading were distributed with constant distance of $4,342$ samples (e.g. Mask 0 $t = (5222, 6777, 8777)$, Mask 1 $t = (29564, 11119, 13119)$ and so on).

Based on the results obtained, we propose an attack that targets this operation. The main idea lies in a combination of profiling and non-profiling power analysis attacks. In the first step, the attack utilizes the templates based on MLP to reveal mask values (secret offset). In the following step, DPA attack is performed (based on correlation coefficient) that is aimed at `MaskedSybBytes`. Note that if the mask values are uncovered to the attacker, from the point of power analysis it is practically an unmasked AES implementation. The block diagram of the proposed attack is depicted in Fig. 3.3.

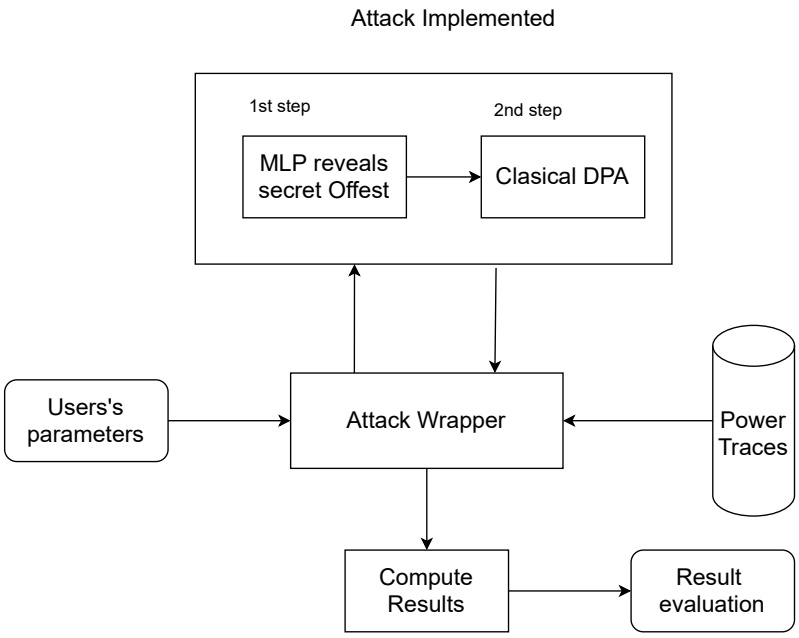


Fig. 3.3: Block diagram of the proposed attack.

An MLP template attack was created and trained in Matlab using the Netlab neural network toolbox [105]. We created a typical two-layer perception network and we used optimized learning based on the scaled conjugate gradient algorithm. A standard sigmoid was chosen as an activation function. The input layer contained 48 inputs corresponding with selected interesting points, hidden layer contained $1,000$ neurons and output layer had 16 neurons and we used 180 training cycles. The learning set and test set consisted of $1,500$ and $1,500$ of power traces successively. Success rate of 99.7% was verified on the test set. After the MLP approach was successfully created and tested, the second step of the attack was realized. The complete attack implementation was investigated using 30 power traces of the publicly available dataset for the first time. The obtained results calculated with tool *ComputeResults* are summarized in Tab. 3.1.

This program provides following characteristics about the attack implemented:

- GSR $> 80\%$: Minimum number of traces for the Global Success Rate (GSR) to be above 80%.
- Min PSR $> 80\%$: Minimum number of traces for the minimum Partial Success Rate (PSR) to be above 80%.
- Max PGE < 10: Number of traces for the maximum Partial Guessing Entropy (PGE) to be below 10.
- GSR stable $> 80\%$: Number of traces for the Global Success Rate to be stable above 80%.

Tab. 3.1: Attack results based on publicly available power traces.

GSR > 80%:	20
Min PSR > 80%:	20
Max PGE < 10:	17
GSR stable > 80%:	20
Min PSR stable > 80%:	20
Max PGE stable < 10:	17
GSR at trace 30:	1
Min PSR at trace 30:	1
Max PSR at trace 30:	1
Min PGE at trace 30:	1
Max PGE at trace 30:	1
Mean time / trace (ms):	560

- Min PSR stable > 80%: Number of traces for the minimum Partial Success Rate to be stable above 80%.
- Max PGE stable < 10: Number of traces for the maximum Partial Guessing Entropy to be stable below 10.
- Time/Trace: Mean time per trace on computer where attack is running (CPU i5-3470 at 3.20 GHz with 16 GB of RAM).

Plots of PGE for individual key byte are depicted in Fig. 3.4. From the plots, we can observe that almost every key byte was revealed successfully based only on 10 power traces with the exception of the key bytes 9 and 13. For these key bytes, about 20 power traces were necessary. Our analysis employing a public data set demonstrated that the attack was able to reveal the whole secret key from only 21 power traces. Evaluation of the attack based on private datasets is outlined in Tab. 3.2.

Tab. 3.2: Results obtained for DPA Contest evaluation.

Key found	Max PGE < 10	Key found (stable)	Max PGE stable < 10	Time/Trace (ms)
23	19	28	19	1,100

The main outcome represents the confirmation of the attack functionality and perfectibility, only 23 power traces were necessary to reveal the secret key. Our attack shows that the RSM masking scheme based on the rotation of masks can be easily broken in practice.

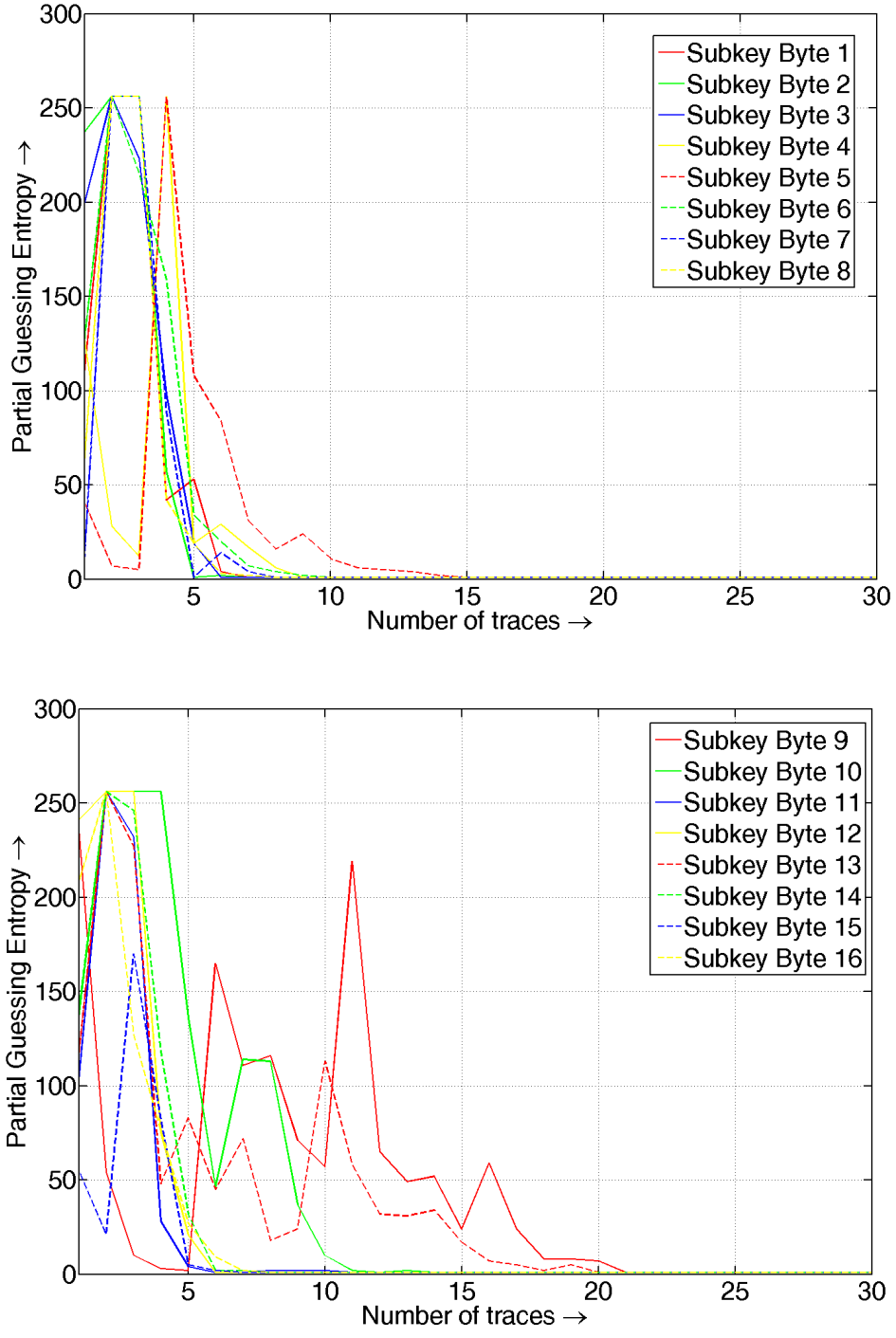


Fig. 3.4: PGE for secret key of the DPA contest V4.1.

3.2 DPA Contest V4.2

Participants of the contest have performed many attacks aimed at the original RSM implementation (DPA Contest V4.1). Different techniques were used such as MIA (Mutual Information Analysis) [144, 71], Collision on the S-box [64] or Recovering the *offset* value based on TA [71]. We refer to work [11] that provides a deep analysis of attacks executed during the DPA Contest V4.1. DPA Contest organizers improved the original RSM implementation by rewriting the code in assembly language, using an individual mask for each S-box and combining the masking and shuffling techniques. This improved implementation is denoted as V4.2 [11, 102]. The implementation was designed based on the knowledge of previous lacks, therefore this approach is devised to resist most of the proposed attacks to the original implementation.

Even though the DPA Contest V4.2 has been open since September 2014, only two attacks have been successfully performed so far [41][4]. Considering the worldwide importance of the DPA contest in the scope of cryptography, this fact is definitely perceived as a proof of the improvement in the implementation resilience of the V4.2 version. Certainly, many researchers have tried to mount attacks on the V4.2 without success. On August 19, 2015 four participants were written in the *Hall of fame* and only one of them implemented two successful attacks[5]. In comparison, in the previous version 31 participants were written in the *Hall of fame* with no unsuccessful attack.

In our work we investigated the security of V4.2 implementation in practice. We used the basic power side-channel techniques in the same way as a potential adversary would do, and tried to analyze the sensitive information. Our analysis, focused on exploiting the first-order leakage, discovered some lacks. The crucial one showed that an adversary can mount a standard DPA attack aimed at the S-box output in order to recover the whole secret key even when a shuffling technique is implemented. We tested this observation on a public dataset, and implemented a successful attack that revealed the secret key using only 35 power traces. In the final step of our analysis, we submitted the attack to the DPA Contest to compare results.

[4]This state of the contest was on July 2015 when the work was written and our attack was implemented and submitted. Note, that the situation can change because the contest is current.

[5]Participants who reported their involvement in the competition are listed in the *Hall of fame*, it dose not mean that they created a successful attack. Results of the attacks that were submitted are summarized in the web-page table.

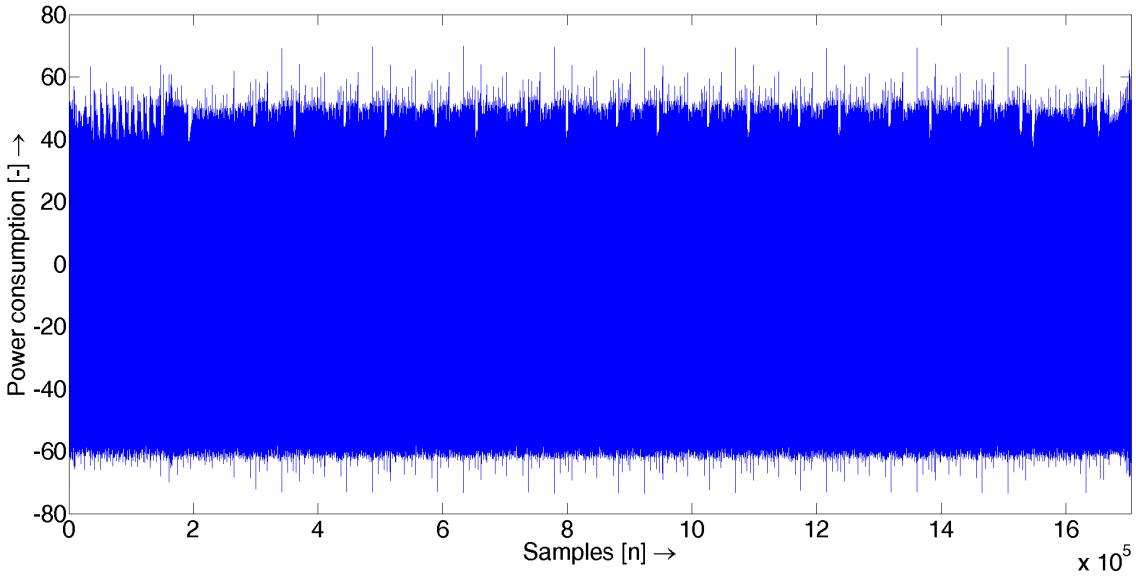


Fig. 3.5: Example of the power trace.

3.2.1 Description of Countermeasures Implementation

In the following text, we briefly introduce the RSM countermeasure V4.2 and the target platform. We provide some relevant observations. The final implementation of RSM V4.2 was written in assembly language and carefully checked to avoid most of the identified pitfalls in the previous version [11]. The 128-bit key version of AES was the selected one, therefore power traces contained a complete encryption process that were acquired using the LeCroy WaveRunner6100A oscilloscope and an EM probe. Figure 3.5 shows an example of a power trace where individual encryption rounds of AES are clearly visible. The target platform was an 8-bit AVR microcontroller Atmega163 embedded in a smartcard. RSM represents a boolean masking scheme that uses 16 public masks[6]:

$$\begin{aligned} M_{i\in[\![0,15]\!]} = \{&\text{0x03, 0x0c, 0x35, 0x3a, 0x50, 0x5f,}\\ &\text{0x66, 0x69, 0x96, 0x99, 0xa0, 0xaf,}\\ &\text{0xc5, 0xca, 0xf3, 0xfc}\}. \end{aligned}$$

These masks are rotating according to the secret value called $offset$ $[\![0, 15]\!]$. Altogether 16 masked S-boxes have to be pre-calculated based on mask values. The

[6]Mask set was updated on 20 July 2015, the original contained following masks: $M_{i\in[\![0,15]\!]} =$ {0x00, 0x0f, 0x36, 0x39, 0x53, 0x5c, 0x65, 0x6a,0x95, 0x9a,
0xa3, 0xac, 0xc6, 0xc9, 0xf0, 0xff}.

masked S-boxes fulfill the following property:

$$\begin{aligned} MaskedSubBytes_{offset[i]+r}(X_i) = \\ = SubBytes(X_i \oplus M_i) \oplus M_{i+1}, \end{aligned} \tag{3.4}$$

which essentially is:

$$\begin{aligned} MaskedSubBytes_{offset[i]+r}(K_i \oplus M_i \oplus X_i) = \\ = SubBytes(X_i \oplus K_i) \oplus M_{i+1}, \end{aligned} \tag{3.5}$$

where the $X[\![X_0, X_1, \ldots, X_{15}]\!]$ denotes 16-bytes plaintext, $K[\![K_0, K_1, \ldots, K_{15}]\!]$ is 16-bytes $RoundKey[r]$ and $r \in [\![0, 9]\!]$ denotes round of AES. To pass through the linear layer of AES algorithm, the mask bytes have to be compensated by exclusive-or (XOR), thanks to sixteen 128-bit precomputed constants. Mask compensation constants are equal to:

$$\begin{aligned} MaskCompensation[i] = \\ = ShiftRows(MixColumns(Mask[offset[i]+ \\ +(r+1)])) \oplus Mask[(offset[i] + (r+1)). \end{aligned} \tag{3.6}$$

For the last round the compensation is different because there is no $MixColumns$ operation, thus mask compensation constants for the last round are equal to:

$$\begin{aligned} MaskCompensationLastRound[i] = \\ = ShiftRows(Mask[offset[i] + 10]) \oplus \\ \oplus Mask[(offset[i] + 10)]. \end{aligned} \tag{3.7}$$

The encryption of the RSM V4.2 works as follows:

1. Draw 16 of 4-bit (uniformly random, unknown) *offset*[] for the key blinding.
2. Draw 2 shuffle values based on shuffle functions for the first and the last round denoted as Shuffle0 and Shuffle10: $[\![0, 15]\!] \to [\![0, 15]\!]$ bijective.
3. Calculate $KeyBlinding$ operation $RoundKey[0] \oplus Mask[offset[]]$.
4. Perform all rounds — Do nine times $r \in [\![0, 8]\!]$:
   (i) Calculate the $AddRoundKey$ operation for every byte $i \in [\![0, 15]\!]$ of state.
   (ii) Then
   $X_i = MaskedSubBytes_{offset[i]+r}(X_i)$; if $r = 0$ then $i \in Shuffle0([\![0, 15]\!])$, in other cases $i \in [\![0, 15]\!]$.
   (iii) Perform $ShiftRows$ and $MixColumns$.
   (iv) Calculate XOR between the state array and $MaskCompensation[]$ constant.

5. Perform the last round:
   (i) Calculate the $AddRoundKey$ operation.
   (ii) Then
   $X_i = MaskedSubBytes_{offset[i]+r}(X_i)$ where $i \in Shuffle10([\![0, 15]\!])$.
   (iii) Perform $ShiftRows$ and $AddRoundKey$ operations.
   (iv) Calculate XOR between the state array and the $MaskCompensationLastRound[]$ constant, which automatically unmasks the state.

Remark 1: From the description above, the main security requirement is deduced to be keeping the *offset* and *shuffle* values secret from the adversary. Once these values are revealed to the adversary, it is theoretically possible to mount successful DPA or TA attacks. The same *offset* value for a set of traces means that the same mask M_i value was used. From the point of view of a DPA attack, it is practically equivalent to have a totally unmasked implementation because Pearson correlation $\rho(x \oplus M_i, y)$ simplifies $\rho(x, y)$. A similar situation occurs for *shuffle* operation: when the value is revealed, the attacker can sort power traces according to this value that accounts for individual bytes of state. In this manner, the points of power consumption corresponding to the $MaskedSubBytes$ operation are in one column in matrix that contains sorted power traces, therefore standard "vertical" DPA attack based on Pearson correlation works. In other words, the $MaskedSubBytes$ operations occur at the same time because randomizing the operation order is suppressed. Based on these observations, we focused our analysis only on secret *offset* and *shuffle* revelation including the S-box operation.

3.2.2 Power Analysis Realized

Similarly as in DPA contest V4.1, the power analysis was carried out in the following steps:

- Localization of interesting points for crucial operation based on correlation coefficients.
- Key observation of possible lacks.
- Attack proposal and implementation (experimental verification on public dataset).
- Attack submission and result comparison.

At the beginning of our analysis, we performed classical differential power analysis based on correlation coefficient (denoted as CPA) in order to localize operations and interesting points (IP) that provided information about processed data. Please, take

into account that every input data is known to CPA in this case (secret key and plaintext, *offsets*, *shuffle*) [78]. Based on the previous analysis, we focused on *Mask loading*, *Shuffle loading* and *SubByte* operations. For the analysis, we used one dataset of DPA Contest V4.2 that contains $5,000$ power traces corresponding with the first secret key denoted as $k00$[7]$= [182, 88, 199, 29, 65, 215, 183, 52, 68, 150, 41, 171, 151]$ (decimal notation).

The CPA results obtained for every 16 bytes of *Mask loading* are depicted in Figure 3.6. In the figure we can observe that this operation takes place at the beginning of the encryption (from 1.7×10^5 to 1.9×10^5 samples), strongly leaking information about mask values. Note that the standard Hamming weight (HW) model was deployed to calculate the hypothetical power consumption. In the next section, we explain how we used the localized IPs to create and test the templates to disclose masks (individual *offset*).

Remark 2: There are usually 256 different exploitable leakages in the case of "standard" boolean masking but only 16 for RSM (considering one byte mask). In the previous version of RSM implementation, it was fairly easy to perform template attacks in order to recover the random *offset* (in this case the whole 128-bit mask) because for each plaintext byte there was a leakage depending on the operation $X_i \oplus M_i$, moreover the sequence of masks was known. Consequently, it was necessary to guess only one of the 16 possibilities and the adversary was able to reveal the whole 128-bit mask.

In the improved implementation, the *offset* is random for every plaintext byte, therefore the adversary has to guess 16^{16} individual *offsets* (masks). Moreover, in our research, we focus on IPs that provide information related to the HW of loaded masks, therefore we can assume that the useful information related to the secret *offset* is very small. It is possible because the HW of every 16 mask is equal to $4 \times 2, 8 \times 4$ and 4×6[8]. Such assumptions about the IP suitability to reveal the secret *offset* must be verified in practice.

The CPA results obtained for every 16 bytes of *Shuffle loading* are depicted in Figure 3.7. We can observe that this operation takes place from 2.0×10^5 to 2.8×10^5 samples, strongly leaking information about the *shuffle* value. As in the previous case, we used these IPs to create and test the templates that disclosed *shuffle*.

[7]Please, note that datasets were updated on 6 July 2015 and since then each of them contains $5,000$ power traces corresponding to one secret key. Old data sets containing only $1,000$ power traces are no longer available.

[8]In the origin mask set, the HW of 14 masks equal 4 besides two mask 0 and 255.

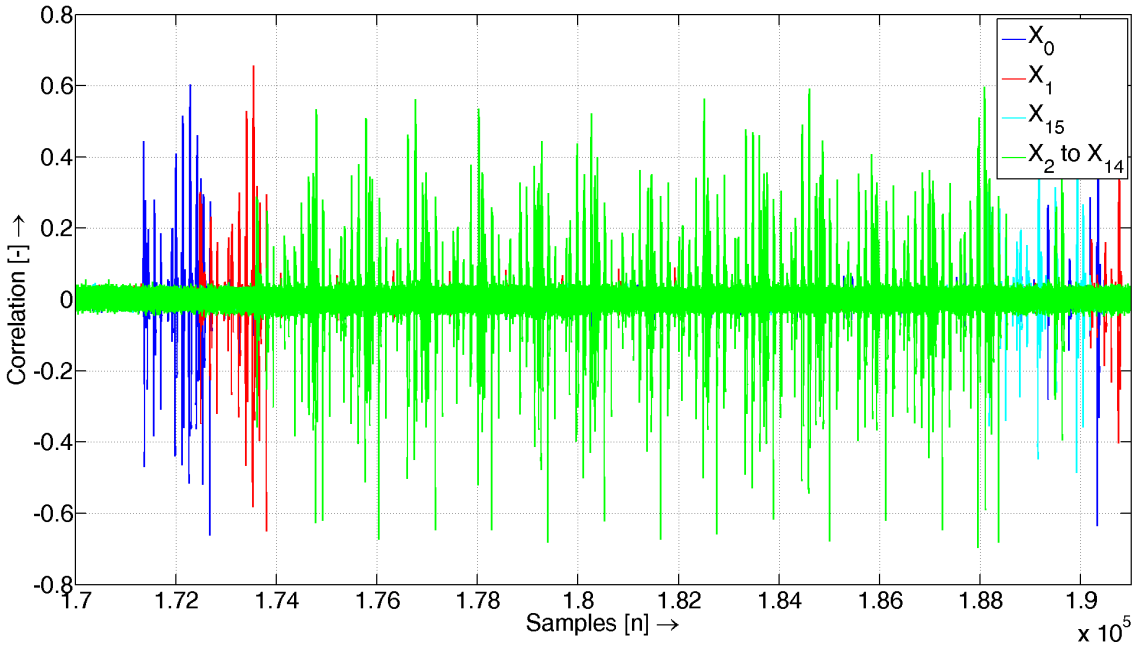


Fig. 3.6: Results of CPA for *Mask loading* operation.

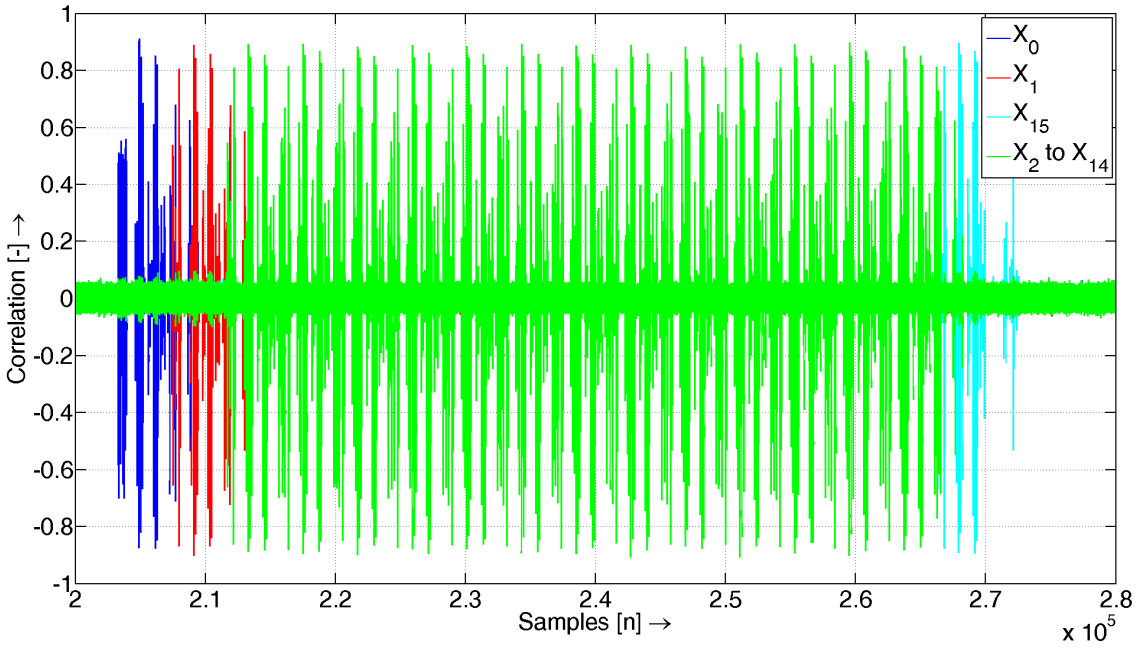


Fig. 3.7: Results of CPA for *Shuffle loading* operation.

The result of S-box localization corresponding with every state byte is displayed in Figure 3.8. Note that our CPA sorted power traces according to *shuffle0* (to fulfill the condition: the first *Shuffle* byte equals zero, the second equals one and so on) in the first step and the correlation was calculated in the second step. Thus, only approximately 300 power traces could be used from the first data set.

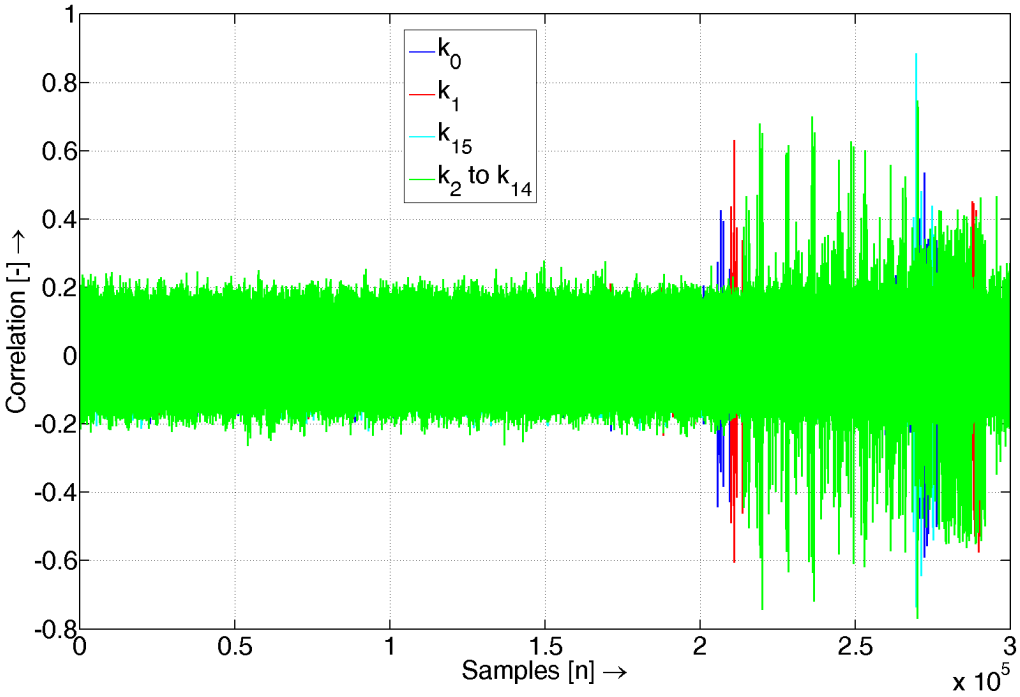


Fig. 3.8: Results of CPA for *SubByte* operation, *shuffle* values are known.

Figure 3.8 clearly shows that S-box operation takes place from 2.05×10^5 to 2.69×10^5 samples. It is a well known fact that the *SubBytes* operation is a perfect target for a standard DPA attack due to its attributes, therefore *masking* and *hiding* techniques are generally aimed at this operation. As described in previsions sections, this fact also applies to the DPA Contest implementation. However, it is apparent that the value of the first S-box byte is processed repeatedly (Figure 3.8). In fact, two time intervals take place from 2.05×10^5 to 2.09×10^5 samples and from 2.70×10^5 to 2.76×10^5 samples. The same situation occurs in the case of other S-box bytes and the second time interval where S-box values were proceed takes place from 2.70×10^5 to 2.92×10^5. We assumed that the second interval corresponded to the following operation *ShiftRows* and it was not influenced by shuffling because shuffling is applied only to the S-box operation in the first and last rounds. In other words, we assumed that the standard CPA aimed at S-box output worked even if the *shuffle* values were unknown. In order to confirm our assumption, we performed another CPA aimed at S-box output with the difference that our program did not consider *shuffle0*. Our algorithm simply calculated the CPA on the basis of the first 300 power traces from the first data set. Results obtained are depicted in Figure 3.9 and confirm our assumption. We observed no peaks corresponding with S-box operation, but the second interval that involved information about S-box outputs was almost unchanged (compare Figure 3.8 and Figure 3.9).

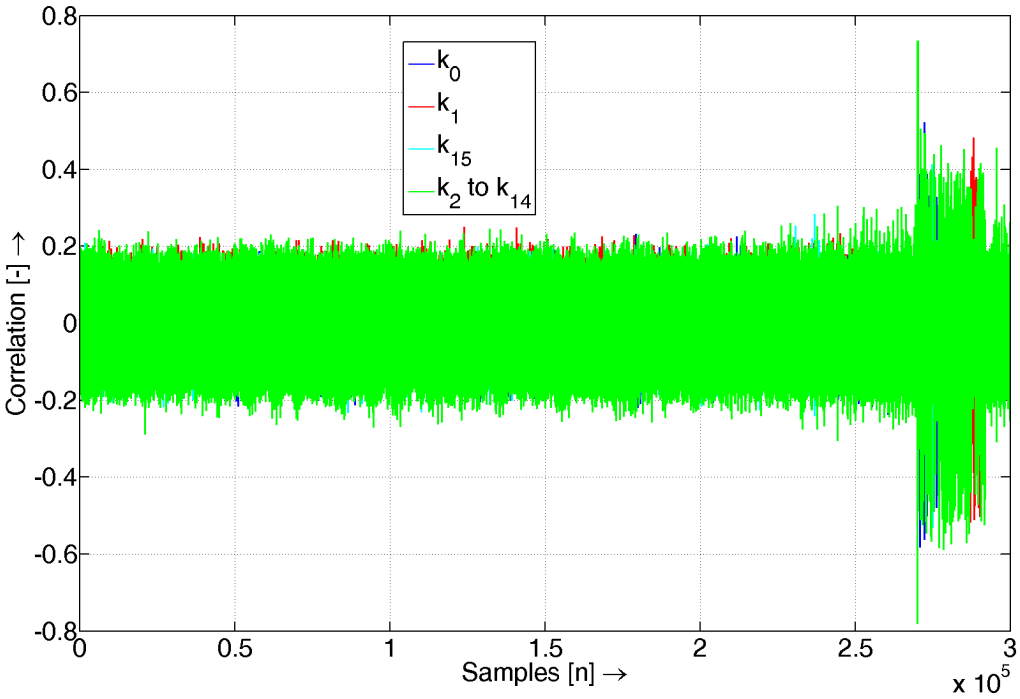


Fig. 3.9: Results of CPA for *SubByte* operation, *shuffle* values are unknown.

At the end of our analysis, we conducted experiments in order to confirm the possibility of mounting a standard CPA attack, and to compare attack results depending on the usage of *shuffle* and number of power traces. Similarly as in the previous case, we implemented two of the standard CPA attacks to reveal the first key byte based on 300 power traces (unknown key, known plaintext and *offset*). The first attack utilized *shuffle0* values and attacked the first time interval (peak from 2.066×10^5 samples). The second attack did not take *shuffle0* into account and was aimed at the second time interval (peak from 2.722×10^5 samples).

Results obtained are depicted in Figures 3.10, 3.11, 3.12 and 3.13. We can observe that the correct key guess can be revealed without any problem using both attacks. The most interesting fact is that the second attack implementation needed only 35 power traces to reveal the secret key byte and the adversary would need about 100 power traces to reveal the secret key in the first implementation. Therefore, an implementation of the attack that uses *shuffle* values actually brings no advantage for the adversary in practice. This fact represents an important lack (thread) of the implemented algorithm because the adversary can use the CPA attack aimed at the S-box without the knowledge of *shuffle* in order to reveal the secret key stored. In fact, the adversary does not attack the S-box itself but the following operation that is not shuffled. It means that the adversary can bypass the shuffling of the S-box in a fairly easy way in a real power analysis attack.

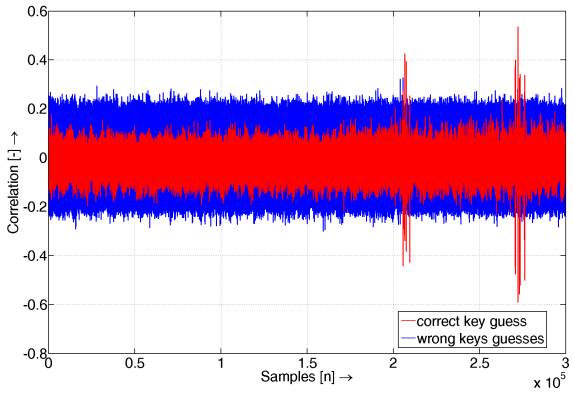


Fig. 3.10: Results of the first CPA attack, *shuffle* values were known.

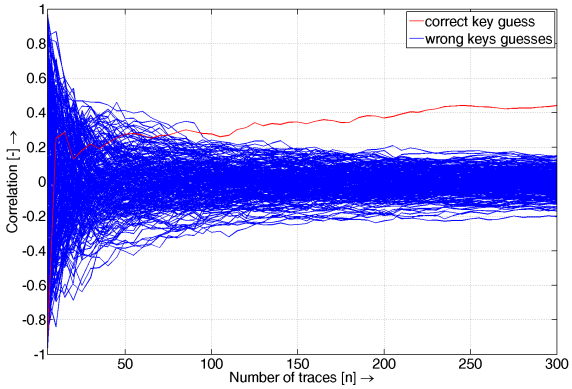


Fig. 3.11: Size of the correlation depending on the number of power traces for the first attack.

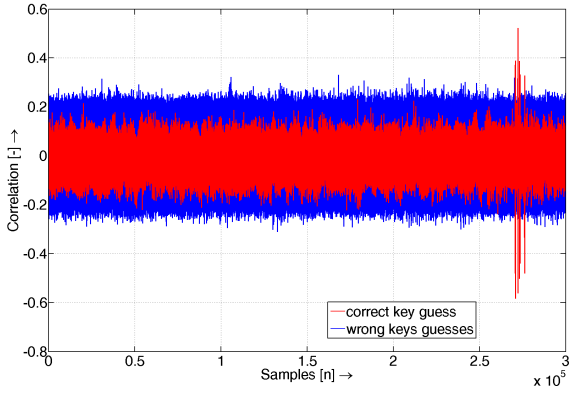


Fig. 3.12: Results of the second CPA attack, *shuffle* values were unknown.

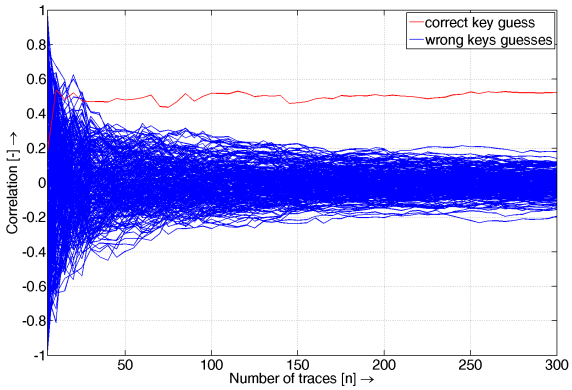


Fig. 3.13: Size of the correlation depending on the number of power traces for the second attack.

Based on the identified pitfall, we proposed and implemented a simple PA attack that consisted of two steps. In the first step, only secret *offset* values were revealed using templates. In the second step, a standard DPA based on correlation coefficient was applied. The DPA was aimed at the S-box output in the first round of AES. In fact, we deployed the outcomes from the previous sections and bypassed *shuffling*, therefore the CPA attack pointed to the following operation that worked with the same intermediate values as the S-box (in the second time interval). The block diagram of the attack proposed including all necessary components of DPA Contest is depicted in Figures 3.14. We implemented the attack in the MATLAB environment. The main tool provided by the DPA Contest is called *AttackWrapper*, it is an interface between implemented attacks and the power traces. Another im-

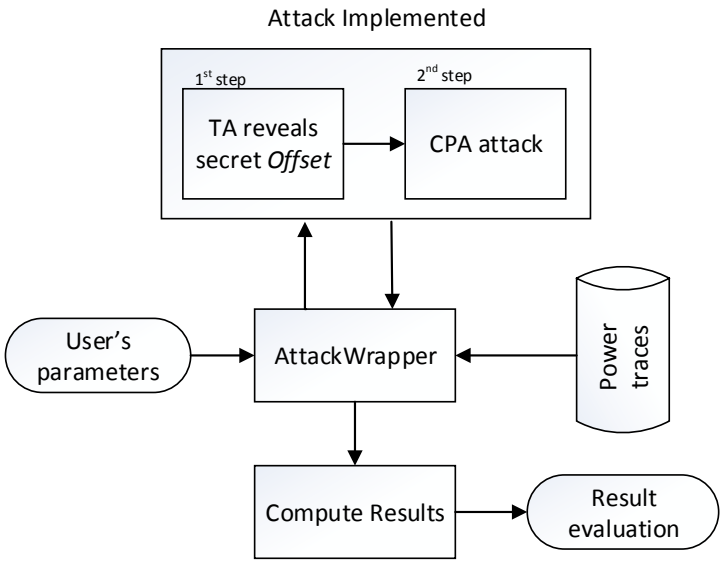


Fig. 3.14: Block diagram of the attack proposed.

portant tool called *ComputeResults* is utilized to analyze the obtained results after the application of the *AttackWrapper*. We built the standard **templates** and evaluated their effectiveness at the beginning of the attack implementation. Calculation of the probability density function was performed according the following Eq. 3.15 in attack phase:

$$p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_i,k_j}) = \frac{exp(-\frac{1}{2} \cdot (\mathbf{t} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{t} - \mathbf{m}))}{\sqrt{(2 \cdot \pi)^{NIP} \cdot det(\mathbf{C})}} \tag{3.8}$$

where $(\mathbf{m}, \mathbf{C})$ represents templates prepared in profiling phase based on multivariate normal distribution that is fully defined by a mean vector and a covariance matrix. The power trace measured from the target device is denoted as $\mathbf{t}$ and NIP is the number of interesting points. Generally, the adversary computes this probability for every template created. Probabilities measure how well the templates fit to the measured power trace $\mathbf{t}$. Intuitively, the highest probability indicate the correct template and because each template is associated with sensitive information (*offset* or *shuffle*), the adversary obtains desire information.

At first, we created templates to determine the secret *offset* for individual bytes (16 templates for 16 state bytes) based on interesting points that we localized. Each template was built based on 100 interesting points that were most correlated with mask loading operation. Generally, we used the first 4,000 power traces from DPA Contest in order to create templates and the following 1,000 power traces to evaluate their effectiveness (denoted as the test set). Moreover, we calculated *confusion*

Tab. 3.3: Confusion matrix of secret *offset* revelation for the first byte.

true → pred. ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	TA precis
0	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1,00
1	0	65	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0,92
2	0	0	64	0	0	0	0	0	0	0	0	0	0	0	0	0	1,00
3	0	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	1,00
4	1	6	0	0	62	0	0	0	0	0	0	0	0	0	0	0	0,90
5	0	0	0	0	0	71	0	0	0	0	0	0	0	0	0	0	1,00
6	3	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0,95
7	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	1,00
8	1	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0	0,98
9	2	0	0	0	0	0	0	0	0	66	0	0	0	0	0	0	0,97
10	1	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0,99
11	0	0	0	0	0	0	0	0	0	0	0	59	0	0	0	1	0,98
12	0	0	0	0	0	0	0	0	0	0	0	0	63	0	0	0	1,00
13	0	0	0	0	0	0	0	0	0	0	0	0	0	61	0	0	1,00
14	0	0	1	0	0	0	0	0	0	0	0	0	0	0	47	0	0,98
15	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	60	0,97
TA recall	0,88	0,92	0,98	1,00	0,91	0,99	1,00	1,00	1,00	1,00	1,00	0,98	1,00	1,00	1,00	0,98	**0,98**

matrices[9] to evaluate all guesses carried out by matching phase templates[10]. An example of the confusion matrix corresponding with the *offset* revelation of the first byte is shown in Table 3.3. The confusion matrix in Table 3.3 crosses predictions with true values obtained from the whole test set classification. Each column of the table corresponds to the correct values of the secret *offset* and each row corresponds to the predicted values. For example, zero *offset* value (the first column) was wrongly estimated once as *offset* 4, three times as *offset* 6 and so on. This *offset* was classified 64 times and only 8 guesses were wrong. On the other hand, there was no wrong estimate for the third *offset* value and all 56 occurrences were matched correctly.

For the following analysis, we denote:

(i) True positive (TP): the template attack predicted the *offset* $= X$ and the actual offset was X.

(ii) False positive (FP): the template attack predicted the *offset* $= X$ and the actual offset was Y.

[9]The interested reader can consult [132] to obtain additional explanations about performance measurements for classification, e.g. *confusion matrix*, *precision*, *recall*.

[10]It is usually more suitable to use *guessing entropy* as a metric to compare different key recovery side-channel attack implementations [135], but we focused on *offset* revelation not the secret key, therefore we used a confusion matrix, which is also often used during profiling PA attacks [34].

(iii) True negative (TN): the template attack predicted the *offset* $= Y$ and the actual offset was Y.

(iv) False negative (FN): the template attack predicted the *offset* $= Y$ and the actual offset was X

where $X \neq Y \wedge X, Y \in [\![0, 15]\!]$ represent *offset* value.

The precision rate is the ratio of correct guesses and all guesses made for individual templates (see each row):

$$\text{TA precis} = \frac{TP}{TP + FP}. \tag{3.9}$$

The recall rate of TA shows the ratio of correct guesses and all guesses made for the correct *offset* (column of matrix):

$$\text{TA recall} = \frac{TP}{TP + FN}. \tag{3.10}$$

Naturally, the accuracy rate indicates the ratio of all correct guesses and the number of all guesses made (in Table 3.3, the accuracy is marked bold):

$$\text{Accuracy rate} = \frac{TP + TN}{TP + FP + TN + FN}. \tag{3.11}$$

We present precision rate (Figures 3.15 a)) and recall rate (Figures 3.15 b)) of *offset* revelation using the boxplot. Input data is 16 columns and 16 rows of confusion matrices corresponding with individual state bytes (a total of 16 times, $1,000$ template matchings were calculated). On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. It can be observed that almost all *offset* values were classified with high precis and recall approaching 100% for every state byte. Moreover, the remaining group of *offsets* was also classified with high precis and recall, on average 97% (*offsets* $2, 6$ and 7). Average accuracy of templates was equal to 98.96%. Based on results obtained, we conclude that that the complete mask (individual *offset* values) can be guessed with high probability.

In the following step, we created templates to determine the *shuffle* for individual bytes based on interesting points that were localized in the previous section to finish our analysis. We created and evaluated templates in the same way as in the previous case (16 templates based on 100 interesting points for 16 state bytes). This templates are not involved in our attack proposed but the results can be useful for future research. Results of precision rate (Figures 3.16 a)) and recall rate (Figures 3.16 b)) of *shuffle* revelation are summarized in Figures 3.16. Similarly as in the case of secret *offset* revelation, some of the *shuffle* values were easier to identify with median precis and recall around 95% such as *shuffle* 0, 1, 3, $10, 12, 13, 14$ and 15.

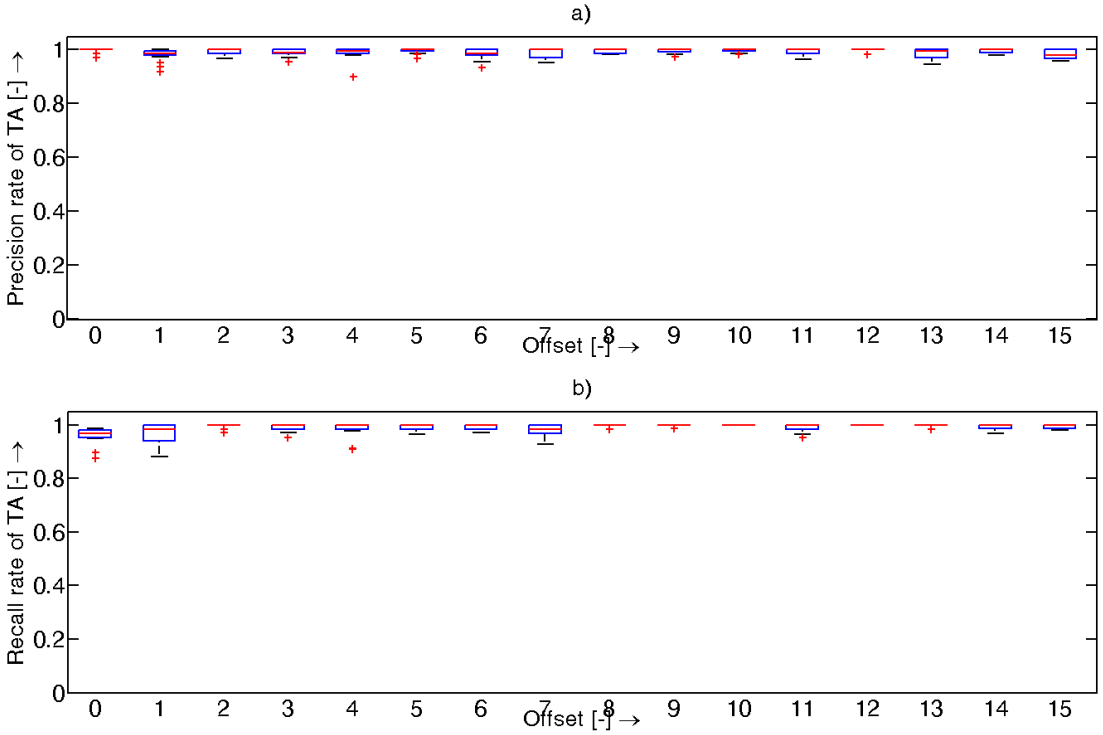


Fig. 3.15: Obtained results of the secret *offset* revelation.

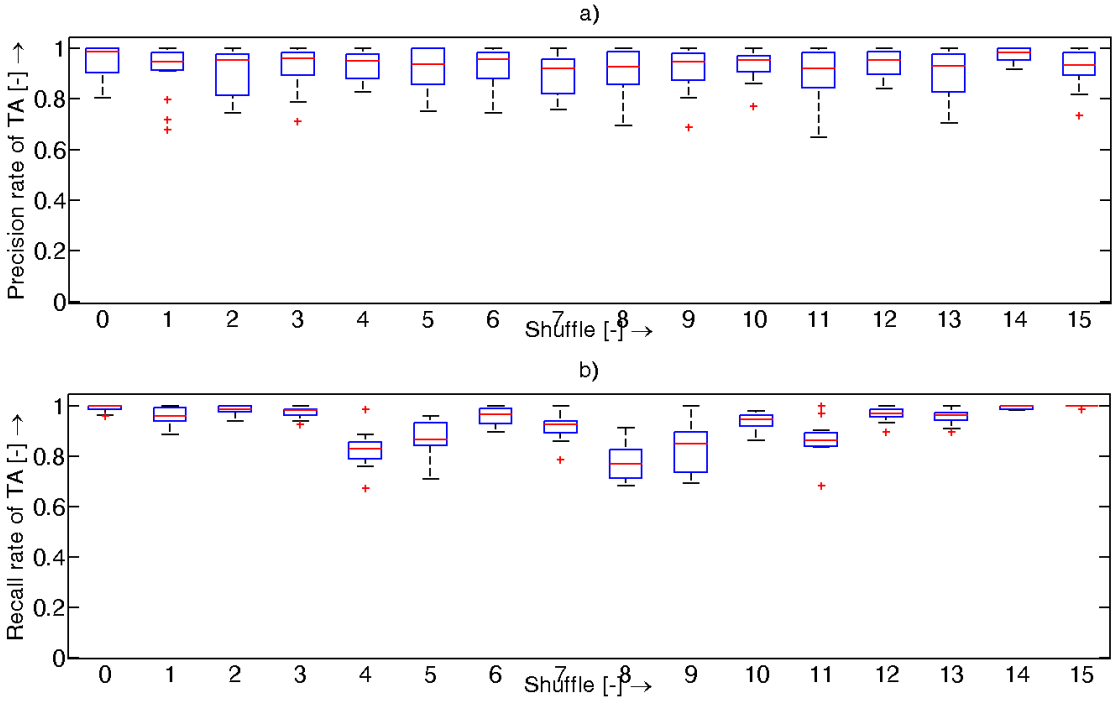


Fig. 3.16: Obtained results of the *shuffle* revelation.

The remaining *shuffle* values were classified with median precis and recall from 80% to 93%. Average accuracy of templates was equal to 92.41%. Based on the results, we conclude that guessing *shuffle* is more difficult than guessing *mask*, however it is possible in practice. In order to complete our analysis, Table 3.4 provides the accuracy rate for individual state bytes of *offset* and *shuffle* revelation, results confirmed previous assumptions.

Tab. 3.4: Accuracy rate for individual state bytes of offset and shuffle revelation.

State byte	0	1	2	3	4	5	6	7	8
Offset	0,98	0,99	0,98	0,99	1,00	0,99	0,99	0,99	1,00
Shuffle	0,95	0,93	0,95	0,96	0,85	0,87	0,92	0,87	0,88
State byte	9	10	11	12	13	14	15	ϕ	
Offset	0,99	0,98	0,99	1,00	0,99	0,98	1,00	0,9896	
Shuffle	0,90	0,96	0,91	0,96	0,93	0,98	0,96	0,9241	

Remark 3: Templates showed that crucial values to keep secret can be discovered by an adversary. Once these values are revealed to the adversary, it is possible to mount standard DPA aimed at the first S-box output. We have proved already that it is sufficient to guess mask values (secret *offsets*) in order to perform successful DPA. A possible critic to the conducted attacks is that wrongly estimated intermediate values resulting from poorly guessed *offset* could decrease the correlation coefficient. In our experiments, the average accuracy of offset revelation was considerably high (98.96%). Furthermore, the adversary can use some of the preprocessing methods to increase the template accuracy and, by extension, the attack efficacy.

Once the template creation was performed, we completed our test attack by implementing the **CPA attack** (the second step) and we evaluated the attack using 100 power traces from the second public dataset. This dataset was measured for the secret key $k01 = [239, 56, 194, 175, 88, 42, 126, 107, 20, 37, 93, 19, 158, 157, 190, 252]$ (decimal notation). Naturally, we deployed the second dataset because all previous analysis that included the template creation were performed using the first dataset (secret key $k00$). In order to prove the performance of the proposed attack, it is necessary to use different power traces corresponding with the different secret key. Results given by the tool *Compute Results* are summarized in Table 3.5.

Tab. 3.5: Result of attack implemented based on 100 power traces.

Min trace GSR > 80%:	35
Min trace Min PSR > 80%:	35
Min trace Max PGE < 10:	31
Min trace GSR stable > 80%:	35
Min trace Min PSR stable > 80%:	35
Min trace Max PGE stable < 10:	31
GSR at trace 100:	1
Min PSR at trace 100:	1
Max PSR at trace 100:	1
Min PGE at trace 100:	1
Max PGE at trace 100:	1
Mean time / trace (ms):	332

Metrics in the Table 3.5 are explained in the following list:

(i) Min. trace GSR > 80%: Minimum number of traces for the Global Success Rate (GSR) to be above 80%.

(ii) Min. trace Min. PSR > 80%: Minimum number of traces for the minimum Partial Success Rate (PSR) to be above 80%.

(iii) Min. trace Max. PGE < 10: Number of traces for the maximum Partial Guessing Entropy (PGE)to be below 10.

(iv) Min. trace GSR stable > 80%: Number of traces for the Global Success Rate to be stable above 80%.

(v) Min. trace Min. PSR stable > 80%: Number of traces for the minimum Partial Success Rate to be stable above 80%.

(vi) Min. trace Max. PGE stable < 10: Number of traces for the maximum Partial Guessing Entropy to be stable below 10.

(vii) Additional characteristics are analogical with the above listed characteristics for the maximum number of power traces.

(viii) Time/Trace: Mean time per trace on the computer where attack is running (CPU i5-3470 at 3.20 GHz with 16 GB of RAM).

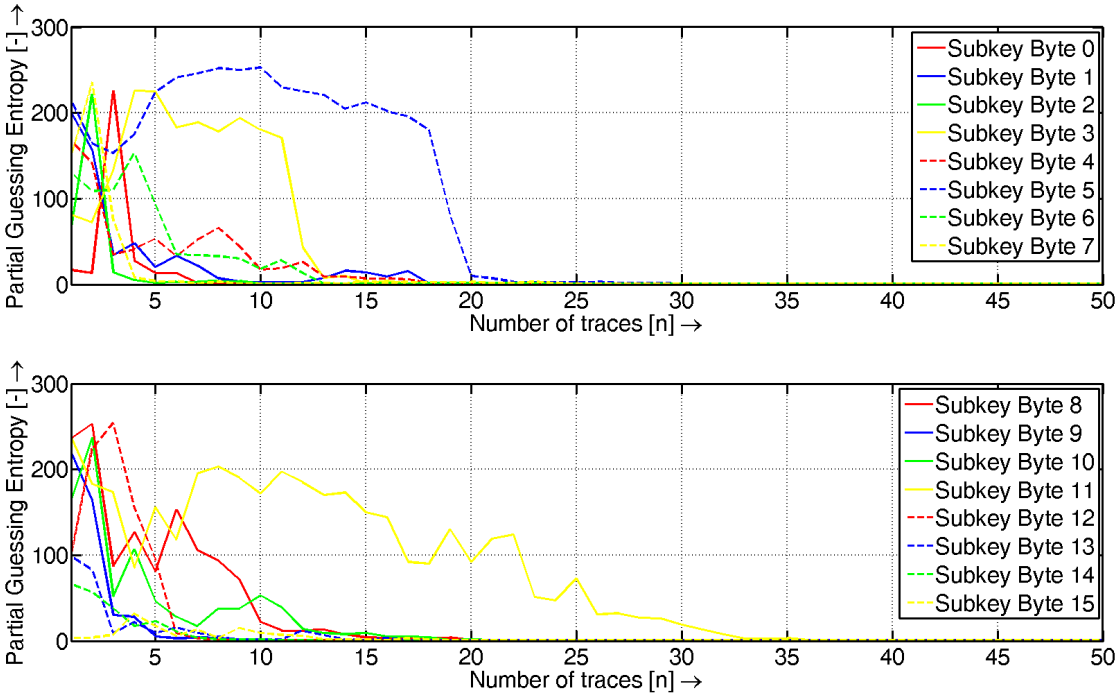


Fig. 3.17: Obtained results of secret key revelation.

The PGE for individual key bytes is plotted in Figures 3.17. Almost all key bytes were revealed successfully based on approximately 15 power traces with the exception of key bytes 5 and 11. In order to reveal these bytes, the attack needed 29 and 35 power traces successively. Hence, based on the second public dataset, our attack discovered the whole secret key by deploying a total of only 35 power traces. Results confirmed the efficacy of the proposed attack methodology and we gave proofs that the improved RSM masking scheme can be easily broken in practice. Our attack was submitted to the DPA Contest V4.2 on 19 July 2015. Evaluation based on $1,000$ power traces of each 16 private datasets was realized in September 2015. The attack was successful, we summarized obtained results of the evaluation in the following Table 3.6 where the individual metrics represent the average value that were obtained from 16 datasets tested. The PGE for individual key bytes is plotted in Figures 3.18.

We carried out the basic power analysis of the improved RSM implementation V 4.2. Our analysis discovered some potential lacks that an adversary could profit to disclose the stored secret key. The main lack lies in the fact that the adversary can attack the S-box output even when the *shuffling technique* is in use. We demonstrated that the implementation of the *shuffling technique* provides no protection in terms of vertical DPA attacks since the adversary does not attack the S-box itself

Tab. 3.6: Result of attack implemented based on 100 power traces.

Min trace GSR > 80%:	104
Min trace Min PSR > 80%:	99
Min trace Max PGE < 10:	107
Min trace GSR stable > 80%:	119
Min trace Min PSR stable > 80%:	119
Min trace Max PGE stable < 10:	107
GSR at trace 1,000:	1
Min PSR at trace 1,000:	1
Max PSR at trace 1,000:	1
Min PGE at trace 1,000:	1
Max PGE at trace 1,000:	1
Mean time / trace (ms):	2,300

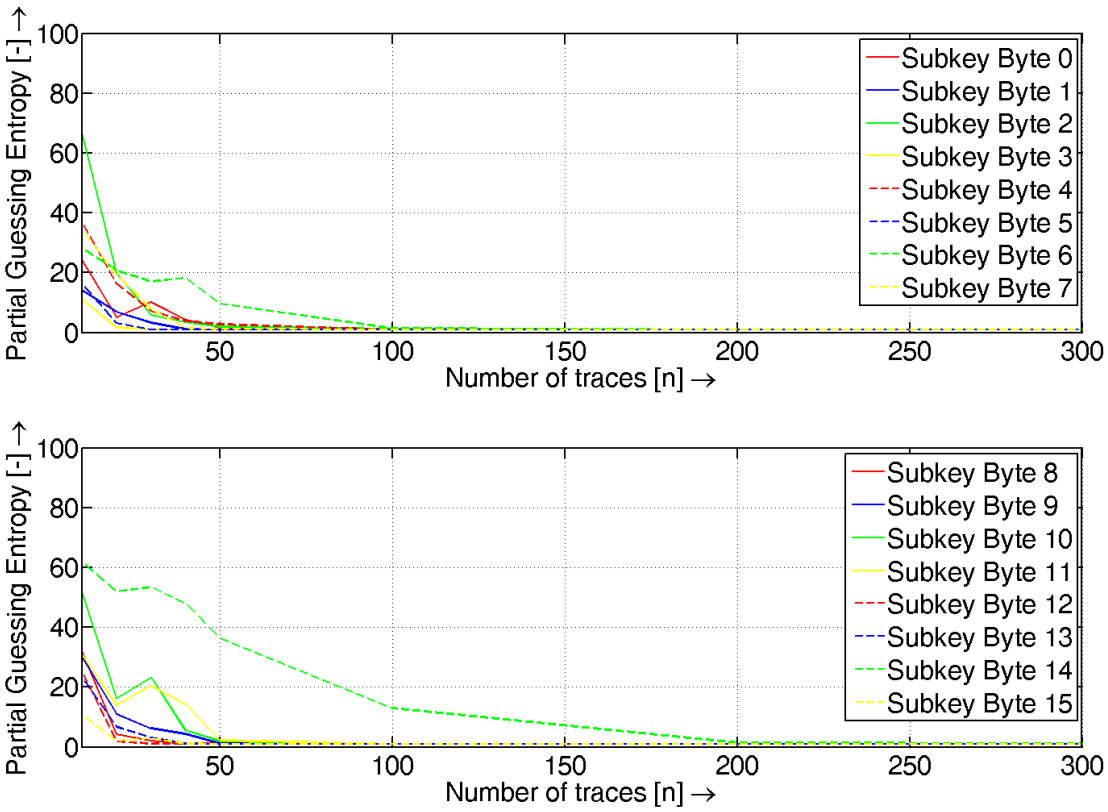


Fig. 3.18: Obtained results of DPA Contest evaluation.

but the following operation that is not shuffled. Based on this principle, we proposed and performed a successfully attack, which consisted of two steps: in the first step, the secret *offset* is discovered using templates; in the second step, the CPA attack is performed. Our attack disclosed the secret key on the basis of only 35 power traces.

3.3 Robustness of Profiling Attacks

In this section, we focus on profiled attacks introduced as the strongest leakage analysis in an information theoretic sense [22]. Several papers highlighted that the characteristics of leakages vary across the measured leakages [25, 30, 125, 140]. More precisely, real world datasets often suffer from errors or distortions in the measured leakages that may affect the efficiency of the adversary. The variability can be due to several factors such as human errors, instrument malfunction (due to the device ageing), variability across different devices or different acquisition campaigns. The impact of these issues on the success of an attack can be reduced with pre-processing techniques, but cannot be entirely removed [140]. In this section, we aim to verify which profiled attack (among conventional profiled attacks and profiled attacks based on machine learning) has the lowest sensitivity to modifications of the characteristics of leakages. For this purpose, we rely on several scenarios among which:

1. leakages associated to a wrong target value in the profiling set,
2. misaligned leakages in the profiling and/or attacking sets,
3. fluctuation of the signal-to-noise ratio in the profiling and/or attacking sets,
4. an increased mean of leakages (called DC offset) in the profiling or in the attacking sets.

All our results are based on distortions and errors applied on real datasets downloaded from the public DPA contests V4.1 [40] and V4.2 [41]. Seting of the profiling method is standard and it is described in section 2.1.

3.3.1 Description of Scenarios and Testbed

We consider a wide range of cases grouped in four scenarios that are listed in the following:

- **Scenario 1**: we increase the number of leakages from the profiling set associated to wrong target values. This scenario can be the illustration of a problem in the protocol used in order to build the dataset as already seen in the DPA Contest V4.1.
- **Scenario 2**: we increase the number of misaligned leakages (from the profiling set and/or from the attacking set). Each (temporal) misaligned leakage is randomly time-shifted from 1 to 6 points from the original leakage. This scenario can be due to a dysfunction in the power supply, an unstable clock, a lack of a good trigger signal or due to countermeasures such as frequency changing, voltage changing and random delay interrupts.
- **Scenario 3**: we increase the level of noise on leakages from the profiling set and from the attacking set. This scenario represents a context in which several

devices (executed near the measurement) influence the environmental noise.

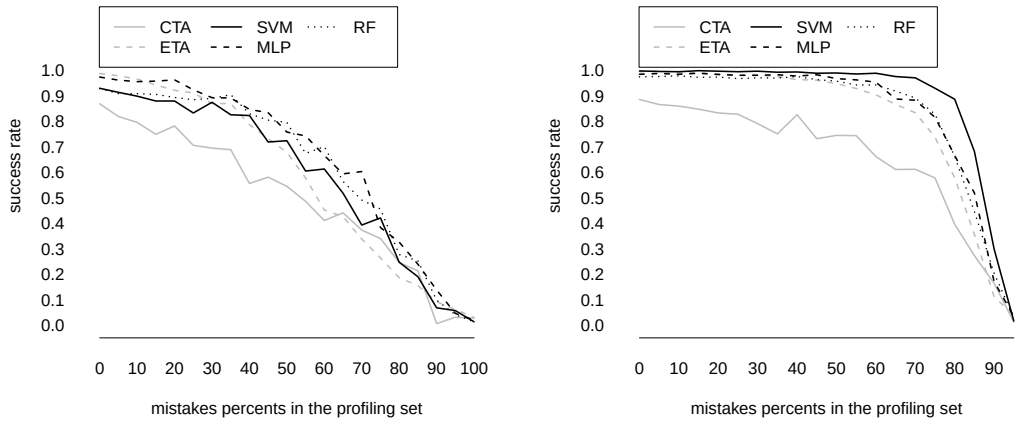
- **Scenario 4**: we increase the mean value of the leakages by adding a constant value (called the DC offset) in the profiling set and then in the attacking set. The DC offset can be the result of (1) a difference between the profiling device (used to build the profiling set) and the target device, or (2) a difference between the acquisition campaign during the profiling and attacking step.

Based on these scenarios, we test the robustness of five profiled attacks that were described in section 2.1. DPA Contest represents an international framework that allows researchers to compare their side-channel attacks under the same conditions. The contest version 4.1 provides leakages associated to the execution of an implementation of AES (128-bit key) protected with a low-entropy Boolean masking scheme called Rotating Sbox Masking (RSM). Regarding RSM, the algorithm was deeply described in sections 3.1 and 3.2, we refer interested readers to the work of Bhasin *et al.* [12] to get more information. Few months after the beginning of the DPA Contest V4.1, the organizers provided an improved implementation of RSM (denoted as version 4.2) to avoid most of the identified pitfalls in the previous version. In the following text, for the clarity, we essentially plot the results based on the DPA Contest V4.2. Nevertheless, we obtain the same conclusion based on the dataset provided by the DPA Contest V4.1. The DPA Contest team used a LeCroy WaveRunner6100A oscilloscope with an EM probe in order to acquire a set of leakages from an 8-bit AVR microcontroller Atmega163. Based on the acquired dataset, we aim to show the sensitivity of profiled attacks by targeting the secret offset of RSM (having an entropy of 4 bits). However, our experiments can be generalized to other sensitive information (e.g., the secret key) and other cryptographic primitives which represent an interesting future work.

In order to build our datasets based on the set of leakages provided by the DPA Contest, we select the features in the traces that (1) linearly correlate the most with the mask value[11], and (2) are distant each other from at least a certain number of samples (a number denoted as *surroundings* in the following). Note that in the following, we will express the signal-to-noise ratio in decibels (dB). In each scenario we vary the number of points per leakage (from 20 to 100 points per leakage) denoted n_s, the number of leakages in the profiling set (from 500 to 4000 leakages) denoted N_p, and the surroundings parameter (from 0 to 2). However we provide figures related to the most informative settings for a reason of simplification and space. We use an attacking set that contains 1000 power traces in order to evaluate the quality of attacks. We consider the first order *succes rate* as a metric of comparison[12].

[11]Note that an adversary targeting the offset or the mask value leads to the same result in our case: the (Pearson) correlation between them equals to one.

[12]Defined as the probability that the model returns the right mask value from one attack leakage.

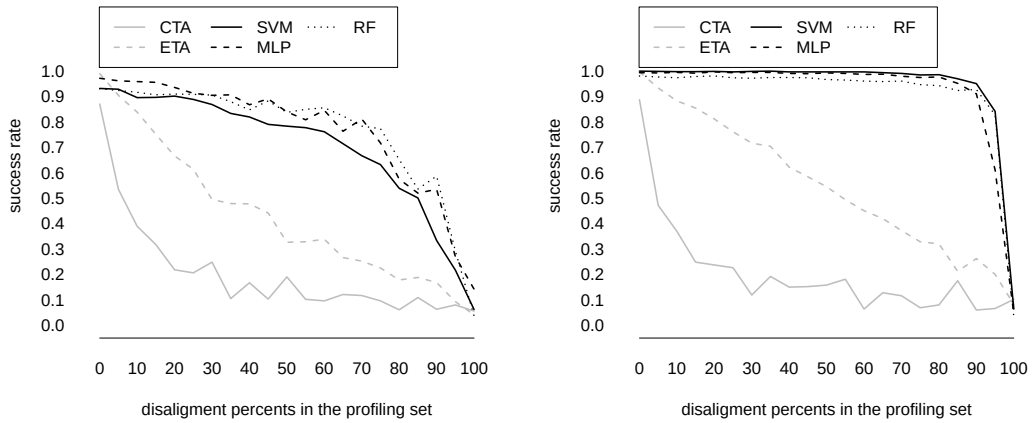


(a) $Np = 500$, $n_s = 50$, surroundings $= 0$ (b) $Np = 4000$, $n_s = 50$, surroundings $= 0$

Fig. 3.19: Probability to retrieve the target value as a function of the number of mistakes in profiling set (DPA Contest V4.2).

3.3.2 Scenario 1: experimental results for mistakes

Figure 3.19 shows the probability of each profiled attack to return the target value when varying the percentage of leakages in the profiling set associated to wrong target values. ETA are the method of choice when there is no mistake in the profiling set. CTA provide the worst results overall due to the high number of parameters to estimate leading to a high sensitivity to errors in the profiling set. It is worth to note that all the methods succeed to have a better success than a random model (i.e., a success rate higher than $\frac{1}{16}$) even with more than 80% of mistakes in the profiling set. More precisely, profiled attacks based on machine learning model outperform conventional profiled attacks in the majority of cases (and provide similar results in the other cases) when the percentage of errors is high (especially with the dataset of the DPA Contest V4.2). For example, based on Figure 3.19(b), with 80% of mistakes in the profiling set provided by the DPA Contest V4.2, SVM reach a success rate of 0.887 while the ETA achieve a success rate of 0.578. The rationale of this result is that (1) the increase of mistakes is equivalent to a reduction of the number of leakages in the profiling set leading to be in a high dimensionality context, and (2) it has been shown that machine learning based attacks outperform template attacks in a high dimensionality context [68, 73].



(a) $Np = 500$, $n_s = 50$, surroundings $= 0$ (b) $Np = 4000$, $n_s = 50$, surroundings $= 0$

Fig. 3.20: Probability to retrieve the target value as a function of the number of misalignments in profiling set (DPA Contest V4.2).

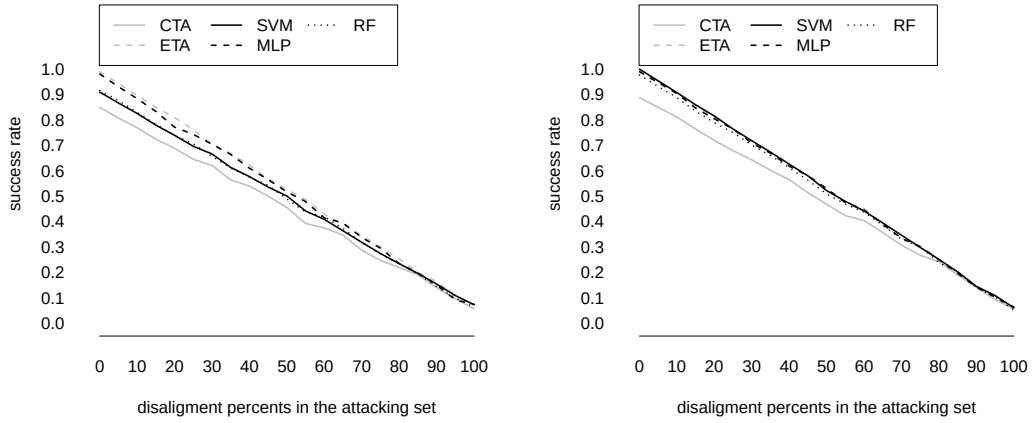
3.3.3 Scenario 2: experimental results for misalignments

Misaligned leakages are easier to exploit (compared to mistakes in the profiling set) since the signal related to target values still persist for several instants. Figure 3.20 shows the success rate of each model when varying the percentage of misalignments in the profiling set. ETA still provide the best results when the percentage of misalignments is low. On the contrary, CTA underperform all the profiled attacks. Note also that machine learning based attacks provide a higher success than ETA when increasing the percentage of misalignments. For example, based on the DPA Contest V4.2 and with 80% percentage of misalignments in the profiling set, Figure 3.20(b) shows that ETA have a success of 0.32 while SVM, MLP and RF reach a success rate higher than 0.95. However, an increase of the surroundings parameter or of the number of points per leakage allow to increase the success of ETA and, as a result, reduce the sensitivity of template attacks to misalignments in the profiling set.

Figure 3.21 shows the results of attacks when leakages from the attacking set are misaligned. The success rate of each model decreases with the percentage of misaligned leakages in the attacking set. Furthermore, the five models perform similarly.

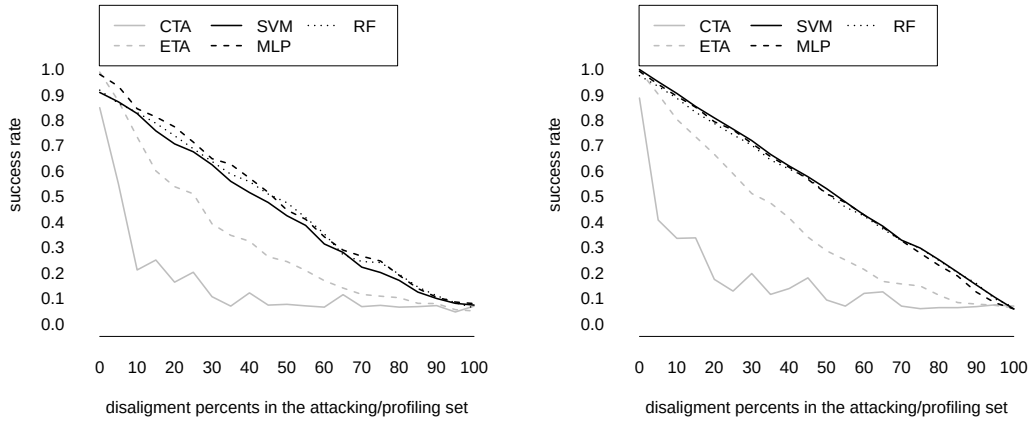
Figure 3.22 shows the results when we vary the percentage of misaligned leakages in the profiling and attacking sets. Three observations can be made:

1. CTA have the worst success rate,
2. ETA and machine learning models perform similarly on the DPA Contest V4.1,
3. machine learning models outperform ETA on the majority of cases based on the DPA Contest V4.2.



(a) $Np = 500$, $n_s = 50$, surroundings $= 0$ (b) $Np = 4000$, $n_s = 50$, surroundings $= 0$

Fig. 3.21: Probability to retrieve the target value as a function of the number of misalignments in attacking set (DPA Contest V4.2).



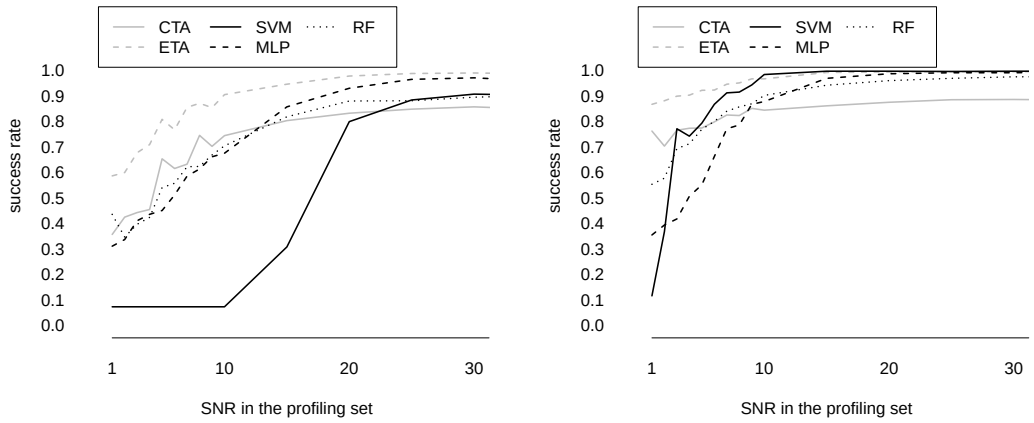
(a) $Np = 500$, $n_s = 50$, surroundings $= 0$ (b) $Np = 4000$, $n_s = 50$, surroundings $= 0$

Fig. 3.22: Probability to retrieve the target value as a function of the number of misalignments in profiling and attacking sets (DPA Contest V4.2).

Regarding the last observation, the success rate of models appears to be related to the sum of (1) the outcomes based on misalignments in the profiling set, and (2) the results based on misalignments in the attacking set.

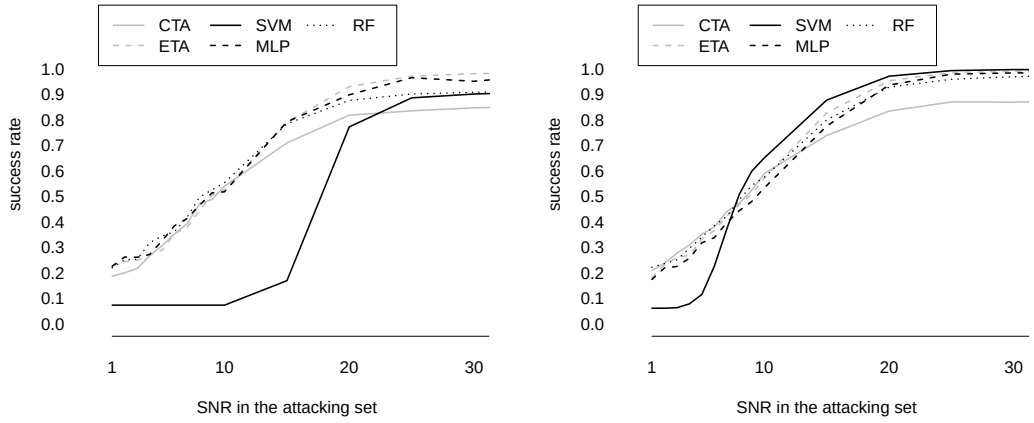
3.3.4 Scenario 3: experimental results for noise

Our third scenario focuses on an increase of the signal-to-noise ratio. Figure 3.23 plots the outcomes when varying the signal-to-noise ratio in the profiling set. ETA outperform all the models in low and high signal-to-noise ratio while CTA under-



(a) $Np = 500$, $n_s = 50$, surroundings $= 0$ (b) $Np = 4000$, $n_s = 50$, surroundings $= 0$

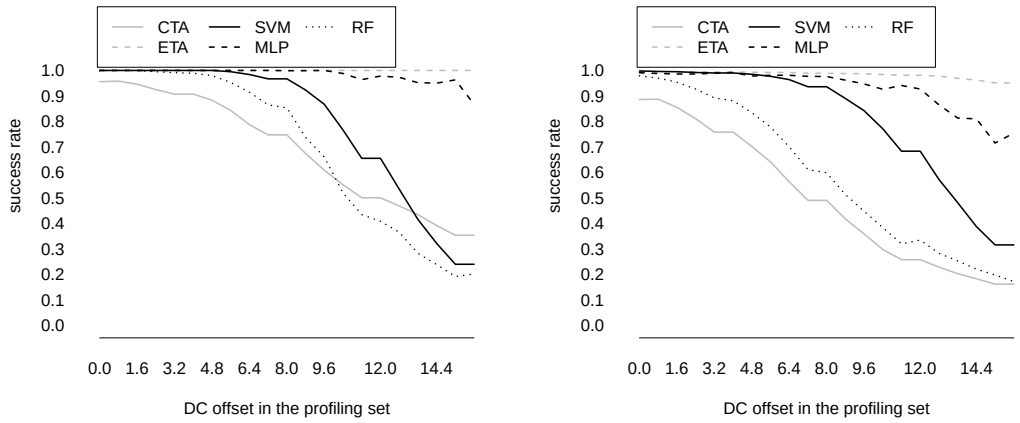
Fig. 3.23: Probability to retrieve the target value as a function of the SNR in profiling set (DPA Contest V4.2).



(a) $Np = 500$, $n_s = 50$, surroundings $= 0$ (b) $Np = 4000$, $n_s = 50$, surroundings $= 0$

Fig. 3.24: Probability to retrieve the target value as a function of the SNR in attacking set (DPA Contest V4.2).

perform all the models in a high signal-to-noise ratio. Figure 3.24 shows the results when varying the signal-to-noise ratio in the attacking set. In a low level of noise, ETA outperform or have similar results than machine learning based attacks. In a high level of noise, the models have similar results except SVM that provide the worst result overall.



(a) $Np = 500$, $n_s = 50$, surroundings $= 0$ (b) $Np = 4000$, $n_s = 50$, surroundings $= 0$

Fig. 3.25: Probability to retrieve the target value as a function of the DC offset applied on the leakages from the profiling set (DPA Contest V4.2).

3.3.5 Scenario 4: experimental results for DC offset

The last scenario analyzes a variation between the profiling and the attacking sets due to a DC offset (i.e., a drift of the global mean of leakages). In a context with a low DC offset applied to the DPA Contest V4.1, ETA provide the best results. However, an increase of the DC offset in the profiling set leads machine learning models (especially a model based on MLP) to outperform ETA.

The results of ETA change when considering the DPA Contest V4.2. Figure 3.25 shows the success of attacks when increasing the value of the DC offset in the profiling set using leakages from the DPA Contest V4.2. We obtain similar results when varying the DC offset in the attacking set. ETA reach the best success compared to other models. Note that this result can be due to the fact that the amplitude of the leakages from the DPA Contest V4.2 differs from leakages provided by the DPA Contest V4.1. In other words, these results highlight that ETA outperform machine learning based attacks when the DC offset is low.

3.3.6 Summary

Obtained results are consistent with the no free lunch theorem explaining that the best model for all scenarios does not exist [141]. Nevertheless, the good results of machine learning algorithms compared to (efficient) template attacks can be explained with the bias-variance theorem recently introduced in the side-channel literature [67].

The bias-variance framework decomposes the error rate (that is inversely proportional to the success rate) of an attack in three weighted terms among which the bias and the variance terms. The values of the variance and the bias relate to the attack complexity: a strategy with a high variance means a high sensitivity to the profiling set while an attack with a high bias indicates a high systematic error compared to the best attack independently of the size of the profiling set.

The bias-variance decomposition shows that (i) CTA have a high variance (i.e., a high sensitivity to the profiling set) due to a high complexity (related to the number of parameters to estimate), (ii) ETA reduce the complexity of template attacks by reducing the number of estimated parameters, and (iii) machine learning models can vary the variance according to a meta-parameter. For example, SVM compensate the increase of the model complexity due to the increase of the number of points per leakage by reducing the variance term through the modification of the meta-parameter γ. As a result, the learning models can handle a larger error in the profiling set (that increases the complexity to learn) while keeping a lower variance term compared to template attacks.

Obtained results underline that efficient template attacks represent the best models when (1) there is no (or a low) variability in the profiling set and in the attacking set, and (2) the level of noise varies between leakages. Overall, classical template attacks provide the lowest success to retrieve the target value. However, profiled attacks based on machine learning gain interest for evaluators of cryptographic devices (1) when the number of mistakes (i.e., the number of leakages incorrectly associated to a target value) in the profiling set increases, (2) when the leakages are misaligned in the profiling and/or attacking sets, and (3) when the leakages from the profiling set and from the attacking set differ from a high DC offset. In summary, our results are of practical importance for evaluators using tools to analyze the leakages of devices.

3.4 *k*-Nearest Neighbors in Power Analysis

Machine learning (ML) as a scientific discipline explores the construction of algorithms that can improve their performance based on previous experiences or trainings [4]. Most of the machine learning problems deal with the classification of various input data. In general, machine learning approaches can be classified as supervised [61] and unsupervised learning [43]. Intuitively, in supervised learning, the machine is presented with a set of training data with the label and the goal is to determine the general function that associates the data with the label. In unsupervised learning, the machine is presented with a set of unlabeled data, and the machine tries to determine the hidden structure of the data.

From the description above, one can clearly see an analogy between machine learning approaches and power analysis attacks. More specifically, profiling attacks are a supervised learning problem, where ML techniques are used for a model creation of the target device. Generally, the model created is based on the multivariate normal distribution in the power analysis. This fact is based on the following simple analysis of a power trace: one can analyze power consumption measured by looking at a single point of a power trace (we look at the power consumption of a cryptographic device at a fixed moment of time). For this point, we can determine the probability distribution that is dependent on the processed data. Generally, it is difficult to make a statement about the data dependency of the power consumption of cryptographic devices. However, for most cryptographic devices, it is valid to approximate the distribution of the data dependency of the power consumption by a normal distribution. Moreover, power consumption of cryptographic devices is mostly proportional to the Hamming weight or the Hamming distance of data processed. In these cases, the distribution is composed of nine normal distributions with different means and the standard deviation is approximately the same. In order to consider the correlation between more points in the power trace, it is necessary to model a power trace measured as a multivariate normal distribution which constitutes a generalization of the normal distribution to higher dimensions. We refer to the book [78] where the authors deal with the complete analysis of statistical characteristics of power traces. On the other hand, non-profiling attacks can be seen as an unsupervised learning. Instead of statistical methods, one can apply ML in order to find the desired structures in the data. In this text, we do not take into account this application of ML.

In recent years, the cryptographic community has explored new approaches in power analysis based on machine learning. In the field of power analysis, the possibility of using neural networks was first published in [122]. Naturally, this work was followed by other authors, e.g. [63], who dealt with the classification of individual power prints. These works are mostly oriented towards reverse engineering based on power print classification. Yang et al. [143] proposed MLP in order to create a power consumption model of a cryptographic device in CPA. Lerman et al. [66, 68] compared a template attack with a binary machine learning approach, based on non-parametric methods.

Hospodar et al. [52, 53] analyzed the SVM on a software implementation of a block cipher. Heuser et al. [49] created the general description of the SVM attack and compared this approach with the template attack. In 2013, Bartkewitz [8] applied a multi-class machine learning model which improves the attack success rate with respect to the binary approach. Moreover, they used (linear) SVM as a preprocessing tool for feature selection, similar as Brank [101]. Recently, Lerman et

al. [69] proposed a machine learning approach that takes into account the temporal dependencies between power values. This method improves the success rate of an attack in a low signal-to-noise ratio with respect to classification methods. Another SVM-based attack was presented in [44] where the authors used SVM to recover the secret key (bit by bit) by exploiting the leakage in the key permutation round.

Lerman et al. [71] presented a machine learning attack against a masking countermeasure, using the dataset of the DPA Contest V4. The method of power analysis based on a multi-layer perceptron was first presented in [92]. In this work, the authors used a neural network directly for the classification of the AES secret key. In [81], this MLP approach was optimized by using the preprocessing of the power traces measured. Lerman et al. [72] introduced a semi-supervised a Template Attack, that combines supervised and unsupervised learning. The method was confirmed by the experiments on an 8-bit microcontroller and by a comparison with a template attack. The authors proposed an unsupervised learning approach for PA in [23] aimed on DES algorithm. Heyszlet et al. [50] introduced an unsupervised cluster classification algorithm k-means to attack cryptographic exponentiation of a public key cryptographic system and recover secret exponents without any prior profiling. Note that the algorithm k-means should not be confused with the k-nearest neighbor algorithm. Zhanget et al. [145] proposed a DPA attack based on the correlation coefficient using Genetic Algorithm. Perin et al. [116] presented the attack based on a clustering algorithm that attacks the randomized exponentiation of the RSA algorithm. In work [6], the k-NN algorithm was briefly mentioned as a possible mutual information estimator. At the end of 2014, Dirmanto presented a small but concise overview of machine learning approaches in power analysis [54]. The survey paper summarizes the main theoretical aspects [54]. During the CHES 2015 conference, Whitnall presented an unsupervised clustering algorithm (K-means clustering) in order to recover a nominal power model [140]. The model was used in a key recovery attack, with minimal requirements in the profiling phase and moreover the approach was effective and robust across an extensive set of distortions. Work [47] contains a survey of machine learning approaches in power analysis. The main milestones in power analysis attacks based on the ML approach are depicted in Fig. 3.26.

Recently, the researchers focus on deep learning techniques that are usable in power analysis attacks. Convolutional neural networks were presented in [17, 76]. The work [10] presented a survey of deep learning approaches in power analysis attacks. How to select a suitable algorithm was presented by Lerman [74]. Semi-supervised SCA based on collaborative learning was proposed in [75] (SVM clasifier was utilized). Mahmoud et al. [77] presented a hybrid power side-channel and modeling attack on strong Physical Unclonable Functions. Simpler models such

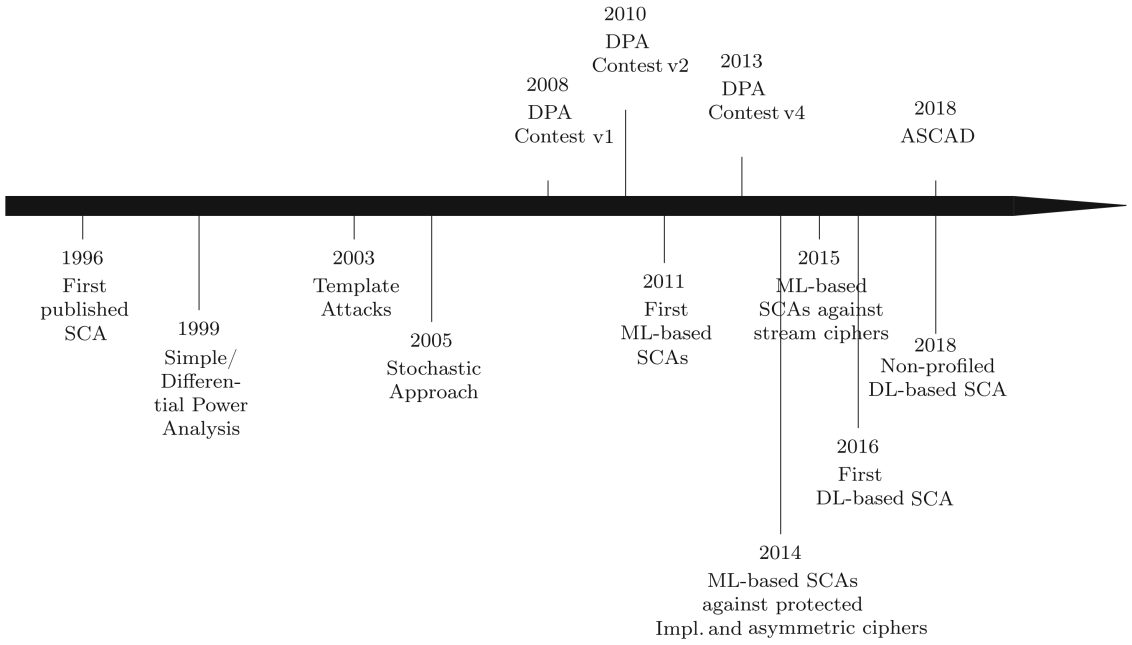


Fig. 3.26: Chronology order of statistical techniques in power analysis attacks [47].

as Bayes classification that lead also to good results were presented in [117]. The hierarchical classification where the goal is to explore the natural clustering of the leakage in order to arrange the class variables was described in [118]. It is clear, that feature selection is crucial for an attack success based on ML. There is only a single paper that systematically compares the effectiveness of proposed techniques from the side-channel domain [48]. Papers [126, 127] present attacks against an FPGA implementation based on various neural networks. Shortly after the publication of the ASCAD data set, Timon exploited the database for non-profiled deep learning [137].

In previous works, individual machine learning (ML) approaches are compared mostly with a template attack or a stochastic attack (SA) [52, 53]. ML approaches have not been compared yet. The work [71] can be mentioned as an exception, where SVM and RF are compared with the TA and the SA. In this section, we try to make an extensive comparison of machine learning algorithms in PA. We focus only on the usage of the individual ML algorithms in profiling attacks where ML techniques are used for a model creation of the target device. We do not consider other possible applications such as structure searching, preprocessing or feature selection. For our research, we implemented a verification program that always chooses the optimal settings of the individual ML models in order to obtain the best classification accuracy. Our research was based on three datasets, the first dataset containing the power traces of an unprotected AES implementation

where we classify one byte of the secret key. The second and third datasets were independently prepared from public datasets of power traces corresponding to the masked AES implementation (DPA Contest V4 [40]) where we classify the secret *offset*. We decided to use the first order success rate as a metric of the comparison because our two datasets were focused on mask classification and Guessing entropy is not suitable in this case[13]. Furthermore, we compared all ML approaches with a template based attack. In this research, we wanted to answer particularly these questions:

- Which ML algorithm is the most suitable for profiling PA attacks?
- Are there any generally appropriate settings of the ML algorithms that can be used by the potential attacker for PA attacks?
- How big is the influence of the number of power traces and interesting points on the classification results of individual ML algorithms?

Nowadays, the method using the SVM is considered to be the most effective machine learning algorithm in the power analysis. In many concrete attacks, in which an adversary has only a limited number of power traces available, the SVM is better in comparison with the classical template attack or the stochastic attack. Based on the results obtained, we propose a power analysis method based on the k-NN algorithm as the most effective method. Even there is no "intelligence", the algorithm shows great application potential in PA, because the usage of the algorithm provides some advantages for the attacker in comparison with the other machine learning approaches and the classic power analysis attack. Moreover, we describe the general scheme of this method in profiling power analysis attacks.

In the previous section, we have already provided relevant references that deal with the well-known ML approaches in power analysis attack (such as SVM, MLP and RF). Therefore, the following text focuses only on the description of the approach proposed based on the k-NN algorithm. We provide the general scheme of this method in profiling power analysis attacks.

Preliminaries: a learning set $\mathbf{Y}$ (sometimes denoted as a training set) and a test set $\mathbf{X}$ with n and m instances represent power traces measured in the context of the power analysis. Each instance $\mathbf{y}_i$ where $i = 1, \ldots, n$ and $\mathbf{x}_j$ where $j = 1, \ldots, m$ in the learning and training sets contains one assignment (a class label that determines which class the concrete instant belongs to) and several attributes $\mathbf{y}_i = y_1, \ldots, y_N$, $\mathbf{x}_j = x_1, \ldots, x_N$ (features and observed variables respectively). These attributes represent the interesting points of power traces in time (samples). The learning set

[13]It is usually more suitable to use *Guessing Entropy* as a metric to compare different key recovery side-channel attack implementations [135], but we focused on the *offset* revelation using two datasets and on the secret key recovery using one dataset. Therefore, we used a confusion matrix and a success rate, which are also often used during profiling PA attacks [34].

is used in the profiling phase of the profiled attack and the test set is used during the attack phase. In the profiling power analysis attack, the label represents the desired byte value of the secret key i.e. together 256 possible variants (0 to 255). From the perspective of ML, we can see this problem as a multiclass classification where ML classifies the instance into 256 possible classes. The second method that is often used is to transfer this problem into the multi-label classification. The multi-label classification represents the problem of finding a model that maps inputs $\mathbf{x_j}$ to binary output vectors $\mathbf{y_i}$. There are two main methods for tackling the multi-label classification problem: problem transformation methods and algorithm adaptation methods [138, 123, 44]. Problem transformation methods transform the multi-label problem into a set of binary classification (two classes 0 or 1) that can be realized by single-class classifiers (such as binary relevance or label powerset). Algorithm adaptation methods adapt the algorithms to directly perform a multi-label classification (Multilabel Neural Networks). In machine learning, the k-Nearest Neighbors algorithm is a non-parametric method used for classification and belongs to the simplest machine learning algorithms [3]. The training phase of the algorithm consists only of storing the learning set into the memory. In the classification phase, k is a user-defined constant (typically small), and a point of the test set is classified by assigning the label which is the most frequent among the k training samples nearest to that classified point. If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

A typical example of k-NN classification is shown in Fig. 3.27. The test sample denoted as a gray square should be classified either to the first class of the white stars or to the second class of the black circles. If the classification process takes into account the three nearest points $k = 3$, the test sample is assigned to the second class, because there are 2 stars and only 1 circle inside the selected area (solid line inner circle). If $k = 5$, the test sample is naturally assigned to the first class because 3 black circles and 2 stars are in the selected area (dashed line outer circle). Commonly used distance metrics for continuous variables are defined as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}, \text{ Euclidean,} \tag{3.12}$$

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N}|x_i - y_i|, \text{ Manhattan,} \tag{3.13}$$

where i represents the number of attributes in the learning set. The overlap metric and Hamming distance are other possible metrics for discrete variables. The best choice of k, that strongly depends on the learning set, is very important. Generally, a large k reduces the effect of the noise on the classification but makes the boundaries between the classes less distinct. A suitable k can be selected by various heuristic

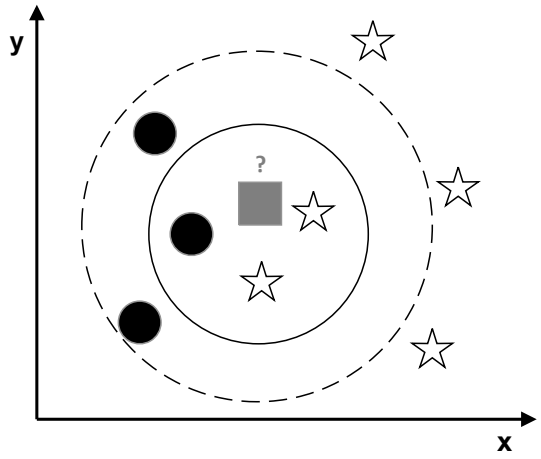


Fig. 3.27: Example of k-NN classification.

techniques, for example the hyperparameter optimization [31]. The following text describes the power analysis method based on k-NN.

Profiling Phase - In the attack based on k-NN, we assume that we can characterize the profiling device by labeling measured data. One can implement a certain part of the cryptographic algorithm and execute the sequence of instructions on a profiling device with different data d_i and different key values k_j, and record the power consumption. After measuring n power traces, we create the matrix $\mathbf{Y}_n$ that contains power traces corresponding to the pair (d_i, k_j). According to the key value, we add a label to the matrix $\mathbf{Y}_n$. In case of byte classification, the label can be expressed by four columns where every row represents a class using the binary expression 00000000 to 11111111 (every possible byte value from 0 to 255). The matrix $\mathbf{Y}_n$ represents a learning set which is stored in the memory.

Attack Phase - During the attack phase, the adversary uses the stored learning set together with the measured power trace from the target device (denoted as $\mathbf{t} = [x_1, \ldots, x_N]$) to determine the secret key value. Let's assume that for our k-NN algorithm we chose the following parameters: $k = 5$ and the Euclidean distance. The classification takes three steps:

- at the beginning, the algorithm calculates Euclidean distances of all stored training vectors $\mathbf{y_n}$ to vector $\mathbf{t}$:

$$d(\mathbf{x}, \mathbf{y_n}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}. \tag{3.14}$$

- In the second step, 5 closest training points are found according to the distances calculated.
- In the last step, the class is selected based on the majority vote.

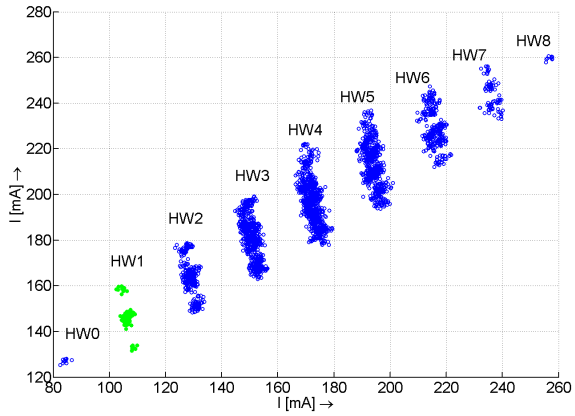


Fig. 3.28: Scatter plot of two interesting points that leak HW.

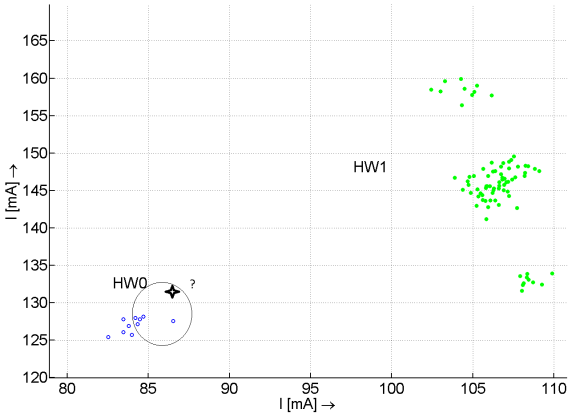


Fig. 3.29: Detail of the scatter plot for HW0 and HW1.

The result of this classification is the most probable class based on training set $\mathbf{Y}$. Since each training instance $\mathbf{y_n}$ is associated with a secret key value, the adversary obtains the information about the secret key stored in the target device.

In the following text, we provide a simple example of the attack based on k-NN using the real data of power consumption (we used only two dimensions i.e. two interesting points were selected) in order to demonstrate the suitability and simplicity of the approach proposed. Let's assume that the adversary wants to determine a Hamming Weight (HW) of processing data or a secret key value. During the profiling phase, the adversary measures $2,560$ power traces, 10 for every byte value (it means that 10 power traces correspond to the HW $= 0$, 80 power traces correspond to the HW $= 1$, and so on). In power traces, the adversary chooses two interesting points that leak information about HW. The profiling phase is finished by storing the points into the memory of a computer. Fig. 3.28 shows scatter plot of two chosen interesting points. The division of the two-dimensional space into nine groups according to the HW is clearly visible. We can approximate this data dependency of the power consumption by a two variable normal distribution (we refer interested readers to consult the statistical analysis in book [78]). In these cases, the distribution is composed of nine normal distributions. In the attack phase, the adversary measures the power trace from the target device and puts the same interesting points to the k-NN algorithm. Fig. 3.29 shows the process of classification for an unknown point that is marked with a black star and for parameter $k = 5$. All five nearest neighbors points belong to the distribution marked with blue color that corresponds to the HW$= 0$. The adversary obtains the desired information that the HW of processed data is 0. Similarly, the adversary can continue with additional power traces in order to reveal the desired sensitive information.

It is obvious, if we focus on two points of the power consumption at fixed time in our example, and realize measurements of power consumption repetitively for constant data then the points measured will appear more or less in the same (group). A similar situation occurs for different data processed by a cryptographic device, therefore the points are in clusters. A widely known fact is that the k-NN algorithm is one of many algorithms that are robust, simple and suitable for classification problems. Using this simple example, we wanted to demonstrate the classification problem during the power analysis attacks and a simple application of ML algorithms.

3.4.1 Description of Scenarios and Testbed

The following text summarizes the most important facts about the experimental setup and the verification program (implemented attacks). We created three datasets in order to test the chosen machine learning approaches. Based on the state of the art, we chose SVM, MLP, k-NN, DT (Decision trees), RF and LDA (Linear Discriminant Analysis).

The first dataset (DS1) is focused only on the first byte classification of the secret key. The dataset is prepared from power traces of the unprotected AES-128 implementation in our testbed. The cryptographic module was represented by the PIC 8-bit micro controller, and for the power consumption measurement we used the CT-6 current probe and the Tektronix DPO-4032 digital oscilloscope. We used standard operating conditions with 5 V power supply. Stored power traces had 100 000 samples and covered the `AddRoundKey` and `SubBytes` operations in the initialization phase of the algorithm. The implementation was realized in the assembly language and the executed instructions of the examined operation were exactly the same for every key byte (identical power prints). Therefore, it was possible to use the place, where the first byte is processed, in order to create a model with which it was possible to determine the whole secret key byte by byte. We verified this assumption experimentally and it is naturally conditioned by the excellent synchronization of measured power traces. Finally, we chose 5 interesting points based on the standard CPA method (we used the well known CPA method to localize interesting points in our whole research). Every of our chosen interesting points leaked information about the Hamming weight of the processed data. We chose points that had a distance at least one clock cycle from each other. This restriction for having IPs not too close avoids the numerical problems when inverting the covariance matrix during the template-based attack.

An example of power traces selected is depicted in Fig. 3.30. Overall characteristics of IPs selected are depicted in Fig. 3.31 using a box plot. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles,

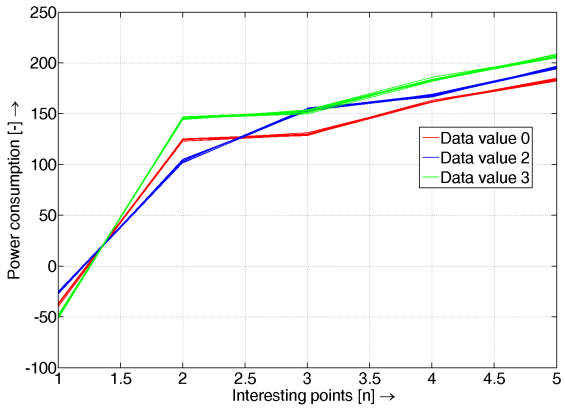


Fig. 3.30: Example of IPs for DS1.

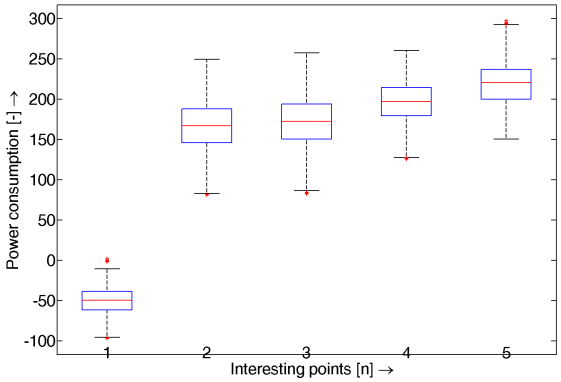


Fig. 3.31: Characteristics of DS1.

the whiskers extend to the most extreme data points not considering outliers, and the outliers are plotted individually. It can be observed that the first dataset does not include almost any noise. Consequently, our first dataset represents a matrix 2560×13 where the last 8 values are labels. Each label is expressed by four columns where every row represents a class using the binary expression from 00000000 to 11111111.

The second dataset (DS2) is focused on the mask classification and consists of 1 000 power traces. DS2 is prepared from electromagnetic traces that are freely available on the website of the DPA Contest V4 [40]. The masked block-cipher AES-256 in encryption mode without any mode of operation is implemented on the target cryptographic device Atmel ATMega-163-based smart card. The implemented masking scheme is a variant of the Rotating Sbox Masking [106, 102]. According to the authors, this masking scheme keeps performance and complexity close to the unprotected scheme and is resistant to several side-channel attacks. Sixteen masks that are incorporated in the computation of the algorithm are public information. The *offset* value, which is drawn randomly at the beginning of the computation, is a secret value. Mask values are rotating according to the *offset* value [106, 102]. Each stored trace has 435 002 samples associated to the same secret key and corresponds to the first and to the beginning of the second round of the AES algorithm. For DS2, we chose only the points that are most correlated with the secret *offset* value. We realized classical CPA for operation `Plaintext blinding` that is dependent on the *offset* value in order to locate the interesting points. We chose the 3 highest correlated points for every mask value, together 48 interesting points were selected. In other words, DS2 represents a matrix $1\,000 \times 52$ where the last four values are labels. In our case, the label value corresponds with the *offset* value 0 to 15. An example of power traces selected is depicted in Fig. 3.32. The overall characteristics of the interesting points selected are depicted in Fig. 3.33 using a box plot.

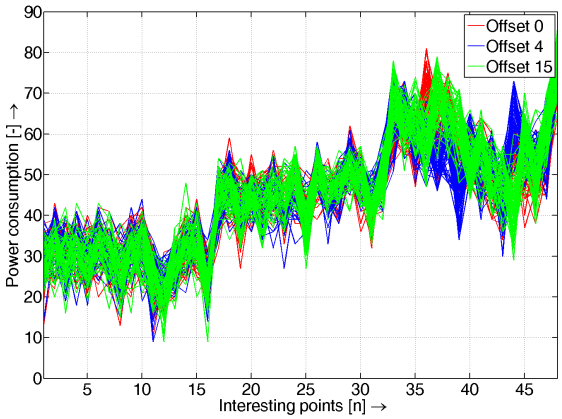


Fig. 3.32: Example of IPs for DS2.

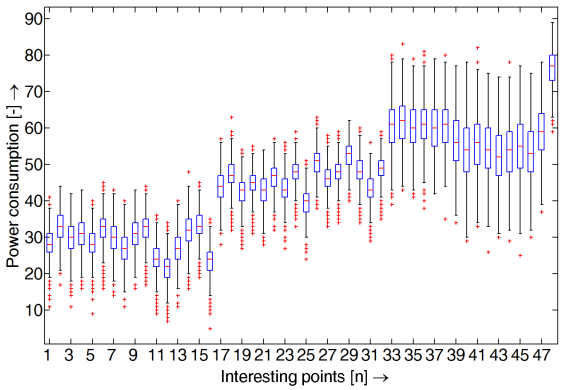


Fig. 3.33: Characteristics of DS2.

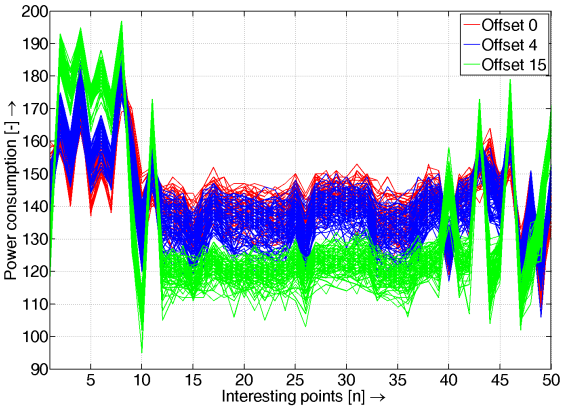


Fig. 3.34: Example of IPs for DS3.

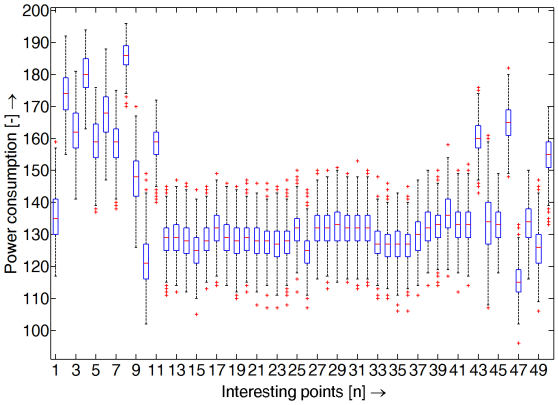


Fig. 3.35: Characteristics of DS3.

The third dataset (DS3) was created by Liran Lerman during preparation of the attack in the DPA Contest V4 [71]. This DS is focused on the mask classification and we used the first 1 000 traces of 1 500 available. The author chose 50 interesting points according to the computed Pearson correlation between each instance of 1 500 traces and the *offset* value. In other words, our DS3 represents a matrix $1\,000 \times 54$ where the last four values are labels. Again, the label value corresponds with the *offset* value 0 to 15 (sixteen possible variants). We refer to the work [71] for more information about the original dataset. An example of power traces selected is depicted in Fig. 3.34. The overall characteristics of interesting points selected are depicted in Fig. 3.35. A well-known fact is that noise always poses a problem during the power consumption measurement. Every stored power trace from DS1 was calculated as an average power trace from ten power traces measured using the digital oscilloscope to reduce electronic noise. We refer to website [40] to consult the level of noise in DS2 and DS3.

3.4.2 Implemented Program

Figure 3.36 shows the main principle of the verification algorithm implemented[14]. The main part of the implemented program is the block denoted as **Optimize Parameters**. This block finds the optimal values of the selected parameters for the tested machine learning algorithms. In other words, it executes each model for all combinations of user selected values of the parameters and then delivers the optimal parameter values as a result. Selected specific parameters are described in more details in the next section. The second important block of the program is the **Cross-validation**. Cross-validation (CV) is a standard statistical method to estimate the generalization error of a predictive model. In l-fold cross-validation, a training set is divided into l equal-sized subsets. Then a model is trained using the other $(l-1)$ subsets and its performance is evaluated on the current subset. This procedure is repeated for each subset. In other words, each subset is used for testing exactly once. The result of the cross-validation is the average of the performances obtained from l rounds. In verification program, we used the typical 10-fold cross-validation. We repeated the CV four or eight times in the **Loop**, because we created four or eight models for individual bit classification depending on the dataset. In other words, in our program we chose the multi-label classification where a model maps inputs $\mathbf{x_j}$ to binary outputs vectors $\mathbf{y_i}$ using single-class classifiers. The verification program returned two output values: the best parameters of the learning models and the obtained accuracy using these parameters. The accuracy was described using the typical confusion matrix.

The original implemented program contained a block called **Forward selection** that selected individual attributes of DSs. In each round, this block can add attributes and the performance is estimated using the inner operators, e.g. a cross-validation. This configuration allows us to get the best results of machine learning algorithms depending on individual parameters setting and attributes selected. In this way, we tested the influence of the number and the combination of selected attributes on the classification results. It is obvious, that time required to solve the program implemented strongly depends on the number of selected parameters of individual ML approaches. One can test an infinite number of parameters in theory, but it has no sense to do so in practice. For example, it is really unnecessary to test the k-NN algorithm for $k > 11$, because the results are worse and the advantages of the algorithm are reduced (based on long years practical experience with ML approaches). For these reasons, we chose only a limited number of parameters that are relevant and important for testing individual ML algorithms. Selection of parameters was realized based on our experience and knowledge with ML.

[14]We use Rapid Miner for implementation [51].

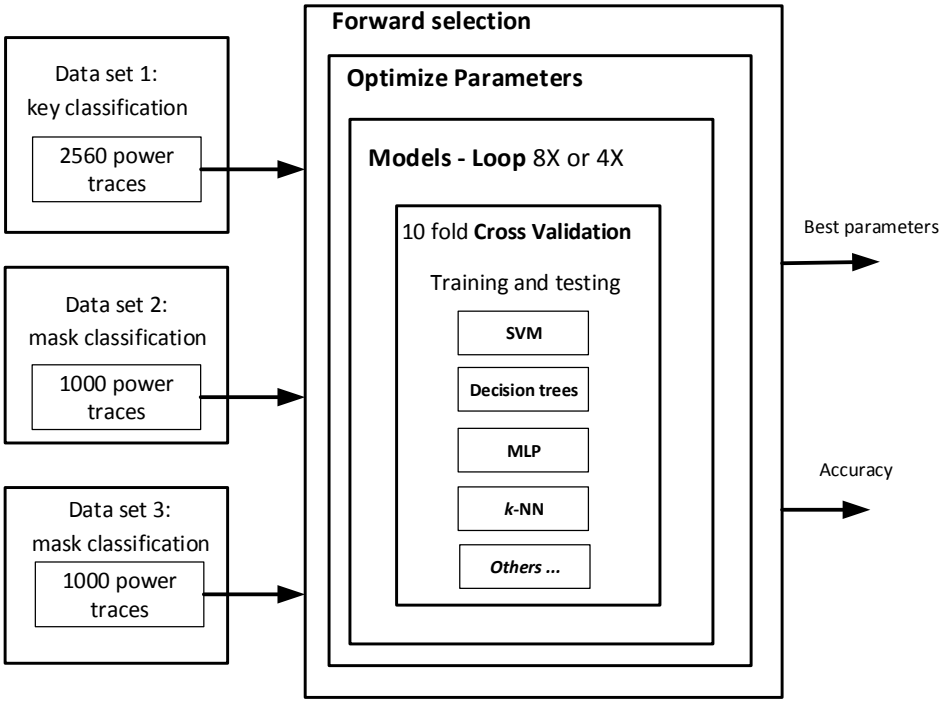


Fig. 3.36: Block scheme of our testing program.

In order to test the **SVM** approach, we chose the complexity parameter from 0 to 50, epsilon from 0.1 to 1 and types of kernel radial, linear, polynomial. Together it made 3,333 combinations. We selected three parameters: the depth from 1 to 100, the confidence from $1.0e^{-7}$ to 0.5 and the criterion set to gain the ratio, information gain, gini index and accuracy (484 combinations) to test **decision trees (DT)**. The **MLP** approach was tested by the following parameters: one hidden layer, type of the activation function, number of training cycles from 1 to 1 000, learning rate, neural network momentum both from 0 to 1 and normalization true or false (5 324 combinations). During the testing of the ***k*-NN** algorithm, we selected different types of metrics: Euclidean, Camberra, Manhattan and Chebychev distance, Correlation, Cosine, Dice, MaxProduct, Overlap and Jaccard similarity, parameter $k = 1, 3, ..., 11$ and a weighted vote (true or false). Together, only 132 combinations were tested. Testing **Random forest (RF)** model involved these parameters: the depth from 1 to 100, the confidence $1.0e^{-7}$ to 0.5, criterion set to gain ratio, information gain, gini index and accuracy, and the last parameter was the number of trees from 1 to 500 (5 324 combinations). As the last tested machine learning algorithm, we involved the linear discriminant analysis **LDA** in default setting.

In order to complete our comparison, we implemented the classical **Template attack**. We were interested in the comparison of effective template attack based on pooled covariance matrix [24], therefore we calculated the pool covariance matrix as an average value of all covariance matrices and we calculated the probability density function (equation (3.15)) with this matrix. Implementations of template attacks were done according to the equation (3.15):

$$p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_i,k_j}) = \frac{exp(-\frac{1}{2} \cdot (\mathbf{t} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{t} - \mathbf{m}))}{\sqrt{(2 \cdot \pi)^{NP} \cdot det(\mathbf{C})}} \tag{3.15}$$

where $(\mathbf{m}, \mathbf{C})$ represents templates prepared in the profiling phase based on the multivariate normal distribution that is fully defined by a mean vector and a covariance matrix. The measured power trace from the target device is denoted as $\mathbf{t}$ and NI is the number of interesting points. In the following text, a classical template and a template attack based on the pooled covariance matrix are denoted as T_{cls} and T_{pool} respectively. In the first experiment, we did not include a reduced template attack, because if the adversary does not consider the covariance matrix, he loses information about the relationship between the interesting points. All template attack implementations were realized in the Matlab environment.

3.4.3 Results Evaluation

The implemented program provided two outputs: accuracy and best parameters selected for individual ML algorithms. In order to calculate the accuracy, a typical confusion matrix was used. Interested readers can consult [132] to obtain additional explanation about performance measurement of classification, e.g. *confusion matrix*, *precision*, *recall*. Examples of confusion matrices of DS1 classification for SVM-rbf and k-NN algorithms are shown in Tab. 3.7 and 3.8. In the confusion matrix, accuracy is the arithmetic mean of the accuracy obtained from the 8×10 cross-validation for individual models. The σ value represents their standard deviation. The value denoted as *mikro* is actually the accuracy computed from the confusion matrix. In other words, it is the success rate calculated for all 20 480 experiments carried out on DS1. It is not possible to present every obtained confusion matrices in this paper, therefore we present the results based on the *mikro* value of the success rate.

In our **first experiment**, we verified that the influence of the block **Forward selection** on the resulting success rate is very low. It is clear, if the selection of interesting points is done in a correct way, the algorithm chooses always the maximum of attributes. This conclusion is natural and not surprising, because selection of the interesting points from power traces is crucially important during

Tab. 3.7: Example of confusion matrix for SVM-rbf based on DS1, parameters selected: rbf kernel, C=50, epsilon 0.46.

Accuracy:	96.15%	σ= 4.45%
mikro:	**94.38%**	
bit value:	0	1
pred. 0:	9640	551
pred. 1:	600	9689

Tab. 3.8: Example of confusion matrix for k-NN based on DS1, parameters selected: Euclidian distance, weighted vote false and k=5.

Accuracy:	99.58%	σ= 0.63%
mikro:	**99.20%**	
bit value:	0	1
pred. 0:	10161 (TN)	84 (FN)
pred. 1:	79 (FP)	10156 (TP)

the profiled attacks, and we used the well know and verified CPA method in order to localize interesting points during the dataset preparation. Based on the results obtained, we skipped this block and always chose the maximum of attributes in the following experiments.

In the **second experiment**, we were searching for the best success rate corresponding to the parameters of the selected machine learning algorithm on our three datasets. In this way, we got the best possible success rates for machine learning algorithms and we could compare machine learning algorithms according to the highest value. Tab. 3.9 summarizes the success rate obtained in percentage. The penultimate rows provide the average values of the success rate calculated from three values obtained and the last rows provide the differences between the average value and the maximal obtained average value. From the results, one can confirm that differences between optimized ML algorithms are negligible. Note, that the SVM with the rbf kernel had the best success rate of all SVM kernels for all datasets. The algorithm k-NN classified the DS1 with the highest success rate from all MLs tested (DS1 has the lowest value of noise). The SVM-rbf was the best ML classifier of the DS2 and DS3. Generally, the template attack based on the pooled covariance matrix[15] was the best in average success rate based on all tested datasets, but if we focus on the ML, the SVM-rbf together with k-NN were the best with almost the same success rate. The difference was only 0.45% and we can consider the differ-

[15]Results of template-based attack are informative, because template attacks were aimed at the whole byte classification.

Tab. 3.9: The highest possible success rate of tested ML algorithms [%].

	k-NN	SVM linear	SVM rbf	SVM poly	DT
DS1	99.20	85.57	94.38	86.93	94.10
DS2	97.65	92.65	99.02	97.92	83.00
DS3	88.05	89.50	92.95	90.12	79.85
ϕ	94.97	89.24	95.45	91.66	85.65
Δ	0.64	6.37	0.16	3.95	9.96
	RF	MLP	LDA	T_{cls}	T_{pool}
DS1	90.24	93.52	85.17	98.44	98.83
DS2	85.48	98.92	93.08	95.60	99.60
DS3	74.95	92.20	89.45	47.00	88.40
ϕ	83.56	94.88	89.23	80.35	95.61
Δ	12.05	0.73	6.38	15.26	0.00

ence negligible taking into account the number of experiments (together 28 480 of individual bit classification). MLP was the third best algorithm with only 0.57% difference from SVM-rbf. We can conclude that the SVM-rbf, MLP and k-NN are the most suitable candidates for profiling power analysis attacks.

The following text summarizes the parameters selected for individual machine learning algorithms during the second experiment.

Parameters selected for DS1:

- MLP: training cycle 475, momentum 0.0, learning rate 0.4, normalization true.
- SVM-linear: C=0.5, epsilon 0.001.
- SVM-rbf: C=50, epsilon 0.46.
- SVM-poly: C=2.5, epsilon 0.46.
- k-NN: $k = 5$, weighted vote false, Euclidian distance.
- DT: maximal depth 39, confidence $1e^{-7}$, no pruning true, criterion gini index.
- RF: number of trees 300, maximal depth 29, criterion gini index.

Parameters selected for DS2:

- MLP: training cycle 720, momentum 0.0, learning rate 0.3, normalization true.
- SVM-linear: C=17.5, epsilon 0.82.
- SVM-rbf: C=45.0, epsilon 0.28.
- SVM-poly: C=12.5, epsilon 0.28.
- k-NN: $k = 5$, weighted vote false, numerical measure OverlapSimilarity, kernel type multiquadric.
- DT: maximal depth 31, confidence 0.4, no pruning true, criterion gini index.
- RF: number of trees 500, maximal depth 70, criterion gini index.

Parameters selected for DS3:

- MLP: training cycle 640, momentum 0.6, learning rate 0.3, normalization true.
- SVM-linear: C=7.5, epsilon 0.91.
- SVM-rbf: C=45.0, epsilon 0.28.
- SVM-poly: C=4.0, epsilon 0.37.
- k-NN: $k = 5$, weighted vote false, numerical measure OverlapSimilarity, kernel type multiquadric.
- DT: maximal depth 71, confidence 0.05, no pruning false, criterion gini index.
- RF: number of trees 500, maximal depth 29, criterion gini index.

We can recognize some similarities of the parameters selected. Definitely, good choices for an attacker can be either MLP with one hidden layer and normalization true, SVM-rbf with parameters C = 45 and epsilon 0.28 or we suggest the k-NN with $k = 5$ and Euclidian distance.

Practically, we can optimize every ML algorithm (using individual parameter settings) to get almost identical classification results. The biggest difference between the tested algorithms lies in the required time that is needed to find the best setting of the concrete ML algorithm. For example, finding the best parameters of the SVM algorithm with poly kernel takes approximately eight days using parameters selected and the DS1. The difference is enormous in comparison with 6 minutes and 35 seconds that was necessary for k-NN optimization for the same DS1. In order to demonstrate this feature, Tab. 3.10 summarizes the time consumption of one executed 10-cross validation for implemented ML algorithms.

Tab. 3.10: Time consumption of 10-cross validation [s].

	k-NN	SVM linear	SVM rbf	SVM poly	DT
DS1	0.2	185	320	2 075	18
DS2	0.3	45	4	4	20
DS3	0.3	10	12	315	26
	RF	MLP	LDA	T_{cls}	T_{pool}
DS1	5 750	320	1	530	530
DS2	890	275	1	30	30
DS3	750	90	1	27	27

The time required to calculate one 10-cross validation for k-NN was less than 1s and 320s for SVM-rbf. Naturally, the attacker has to calculate so many numbers of CV as is the number of the tested parameters, therefore the time needed is directly proportional to the number of the tested parameters and the learning time. In our case, we tested only 132 combinations of the parameters for k-NN, but 1 111

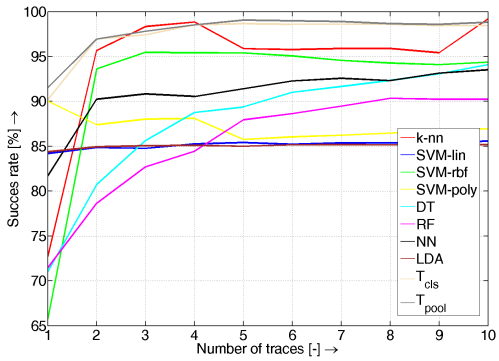


Fig. 3.37: Classification results DS1.

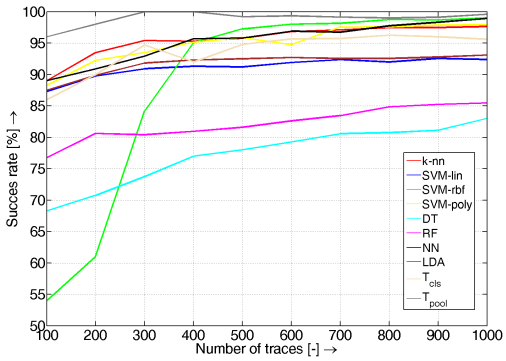


Fig. 3.38: Classification results DS2.

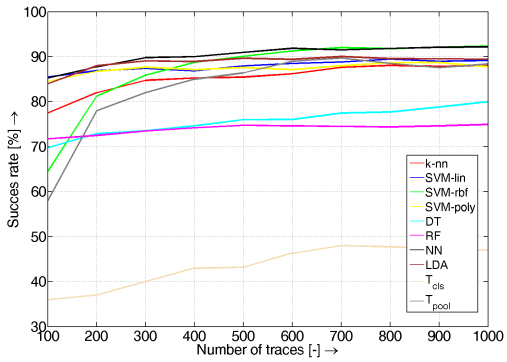


Fig. 3.39: Classification results DS3.

combinations for SVM with poly kernel. The algorithm k-NN is really easy, therefore it is not necessary to test many parameters and the algorithm does not include a learning phase. These are the main reasons why in terms of time consumption, the k-NN algorithm provides the best performance in our implementation[16].

The **fourth experiment** examines the classification success rate based on the number of power traces. For this purpose, we prepared new datasets from the original three DSs that differed in the number of power traces. From the first DS1, we created 10 datasets each containing one more power trace that corresponds to each key value. In other words, datasets created have from 256 to 2 560 power traces with step 256. From DS2 and DS3, we created again 10 datasets containing 100 to 1 000 power traces with step 100. Data prepared were classified successively using the implemented program. The obtained results are depicted in Fig. 3.37, Fig. 3.38 and Fig. 3.39.

[16]Note, that our test was performed on datasets containing 1 000 and 2 560 power traces.

Displayed graphs confirm the theoretical assumption of the increasing success rate with the increasing number of power traces in the profiling phase. The success rate is precipitously increasing until the maximal value and after that the value of the success rate stays almost constant. An interesting fact is that the number of power traces required to achieve the maximum value was comparable for all ML algorithms (especially for the best three, SVM-rbf, MLP and k-NN). Generally, about 500 power traces of DS1, 300 power traces of DS2 and 200 power traces of DS3 were necessary to achieve the maximal value. From a comparison of Fig. 3.38 and Fig. 3.39, we can see the influence of choosing the interesting points because these DSs were prepared based on identical power traces and aimed at the secret *offset* classification (in other words, datasets differ only in the method of selecting interesting points). The shift of the maximum success rate values around 10% is obvious. During the DS3 classification, the classic template attack provides really low values of calculated probabilities, therefore the first order success rate was worse when compared with other approaches.

In our **fifth experiment**, we investigate the success rate of the mask revelation depending on the number of interesting points and the number of power traces. Moreover, we investigate the influence of multiclass classification. In comparison with our previous experiments, we modified the program in such a way that ML classified 256 classes (whole byte classification). In other words, ML and TA classified secret *offset* directly (not successively bit by bit). For this purpose, we prepared datasets based on DS2 that differed in the number of power traces and interesting points. We prepared learning sets that contain 100, 250, 500 and $1,000$ of power traces successively and a test set that contains $1\,500$ instances in order to test profiling attacks. Figures 3.40, 3.41, 3.42 and 3.43 report the success rate to predict the right *offset* value as a function of the number of interesting points selected for the best profiled attacks: SVM-rbf, NN, k-NN, T_{cls} and T_{pool}. The experiments described in [92, 81] implemented MLP approach in Matlab using Netlab. In order to extend our research on testing different implementation, we involved also the MLP approach implemented in Netlab [105] (denoted MLP_Matlab) and we included to this experiment a reduced template attack denoted as T_{red}. The reduced template attack is calculated according to the equation (3.15) but the covariance matrix is equal to the identity matrix (reduced templates contain only the mean vector). One can extract the following observations. First, as expected, the higher the number of traces in the learning set, the higher the accuracy. For example, maximal success rate achieved was 70% and 99% for learning sets containing 100 and $1\,000$ power traces respectively. Second, the number of the selected points in each trace influences the success rate: the higher the number of interesting points, the higher the success rate of all attack implementations.

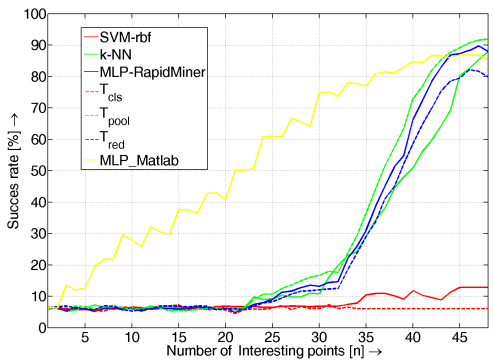


Fig. 3.40: Success rate of the secret *offset* revelation based on 100 power traces of DS2.

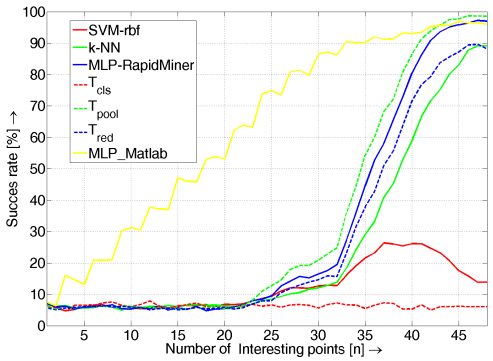


Fig. 3.41: Success rate of the secret *offset* revelation based on 250 power traces of DS2.

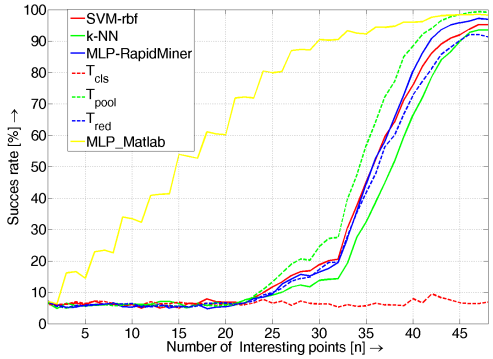


Fig. 3.42: Success rate of the secret *offset* revelation based on 500 power traces of DS2.

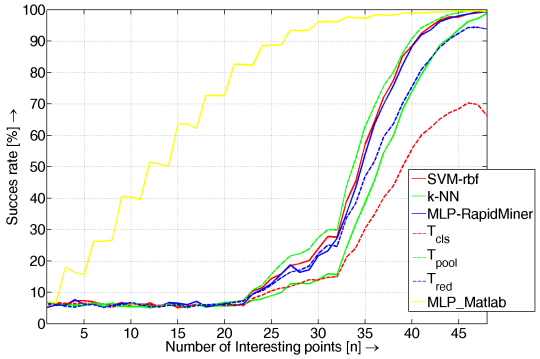


Fig. 3.43: Success rate of the secret *offset* revelation based on 1000 power traces of DS2.

The main finding is that the rise in the success rate of the attacks based on ML occurs much earlier than for every TA attack. We can observe success rates of 72% for MLP and 7% for TAs for 20 interesting points and 1 000 power traces. It is remarkable that if the learning set is small (in our experiment less than 1 000 power traces), the classic template attack is practically inapplicable. It provides the success rate somewhere around 7%. This is caused by the numerical problems that are connected with the covariance matrix. These numerical problems occur during the inversion which needs to be done in equation (3.15). In our case, the values calculated were very small and that lead to bad classification results. The obtained results confirmed that generally the ML approach is much more effective profiling power analysis attack in terms of a small number of power traces and interesting points.

It is pretty surprising that the MLP_Matlab approach is better in comparison with the second implementation. It is caused by more precise settings. The fact is that the template attack based on the pooled covariance matrix and the ML approaches (NN and SVM) are practically the same for larger learning sets. The obtained success rates were 99.9% and 99.6% for T_{pool} and MLP respectively. Furthermore, the results obtained confirmed that k-NN is more similar to the classic template attack. The success rate lies between T_{cls} and T_{pool}. This approach is much more efficient than the classic template attack for smaller datasets, on the other hand, T_{pool} is better, because it considers the relation between the interesting points selected. In practice, the k-NN approach corresponds with the reduced template attack, that does not take the covariance matrix into account. Naturally, we realized the same experiment for DS3, and we evaluated the results obtained. Not surprisingly, the results were practically the same, therefore we did not include them in the text.

In our **last experiment**, we performed ROC (Receiver Operator Characteristic) analysis for the chosen profiled attacks based on machine learning algorithms. This method is commonly used in medical decision making, and in ML in order to illustrate the performance of a binary classifier as its discrimination threshold is varied. Therefore, ROC graphs are useful to organize and, select classifiers and to visualize their performance. The creation of the ROC curve is described in the following text. The results of accuracy for individual models are calculated based on the confusion matrices (see section 3.4.3 and example in Table 3.8 for k-NN). The numbers along the major diagonal represent the correct decisions, and the numbers off diagonal represent the errors, the confusion between the various classes. In other words, the columns of the table correspond to the correct values of the class (in our case bit value 1 or 0) and the rows correspond to the predicted values.

For the following analysis, we denote:

- True positive (TP): the model predicted bit value 1 and the actual bit value was 1.
- False positive (FP): the model predicted 1 and the actual value was 0.
- True negative (TN): the model predicted 0 and the actual value was 0.
- False negative (FN): the model predicted 0 and the actual value was 1.

The *sensitivity* of a classifier is estimated as:

$$sensitivity = \frac{TP}{TP + FN}. \tag{3.16}$$

The *specificity* of a classifier is estimated as:

$$specificity = \frac{TN}{TN + FP}. \tag{3.17}$$

Generally, ROC graphs are two-dimensional graphs where *sensitivity* is plotted on the Y axis and $1 - specificity$ is plotted on the X axis, therefore they depict

relative tradeoffs between benefits (TP) and costs (FP). It is well known that the best possible prediction model would yield the point $(0, 1)$ in the upper left corner of the ROC space. This point is also called the "perfect classification", because it represents 100% sensitivity (no FN) and 100% specificity (no FP). The perfect classier produces a curve that runs vertically upwards from the origin $(0, 0)$ up to the point $(0, 1)$ and from this point horizontally to the right. A completely random guess (considering binary classification with the success rate of 50%) would give a line along a diagonal from the origin (0,0) to the top right corner (1,1).

Based on previous results we involved MLP, SVM-rbf, k-NN, DT and RF into the ROC analysis. We implemented ML using optimal parameters that were discovered using the second experiment (please consult in Section 3.4.3 - part of *the second experiment*). We calculated ROC based on whole datasets prepared and 10-fold cross validation. The results of ROC analysis corresponding with each bit classification of DS1 are depicted in Fig. 3.44.

The semitransparent areas indicate the standard deviation that results over the different cycles of 10-fold cross validation. Solid lines indicate the average results of the cross validation performed. It can be observed that some of the bits of the secret key can be perfectly distinguished. This group includes the first bit, the third bit and the eighth bit. In these cases, each model provides an almost perfect classification. Moreover, the remaining group of bits was also classified with high performance, but the differences between ML models were more significant. It is remarkable that the ROC curve plot of $k-$NN was the closest to the perfect classifier for each remaining bit and the second-best model was the SVM-rbf. The model based on $k-$NN provided a perfect classification even though other models did not (classification of bits 4, 5, 6 and 7). On the other hand, it was interesting that according to the ROC comparison, MLP was the worst for DS1. Naturally, the observation corresponds with the results of the second experiment (please consult this observation in Tab. 3.9). Based on our experiments and experience with MLP in profiled power analysis attacks, we concluded that it is caused by a small number of interesting points[17].

The results of ROC analysis corresponding with each bit classification of DS2 are depicted in Fig. 3.45 and show that each 4 bits of the secret *offset* were classified with a great performance. Based on ROC plots of k-NN, SVM-rbf and MLP, it can be observed that these models are really close to the perfect classification. The difference between these models is really negligible for DS2. ROC curves corresponding with each bit classification of DS3 are depicted in Fig. 3.46. As in the

[17]Since 2010, we have implemented mainly the MLP approach instead of the standard template attack in our profiling power analysis attacks that have been conducted in research and DPA Contests.

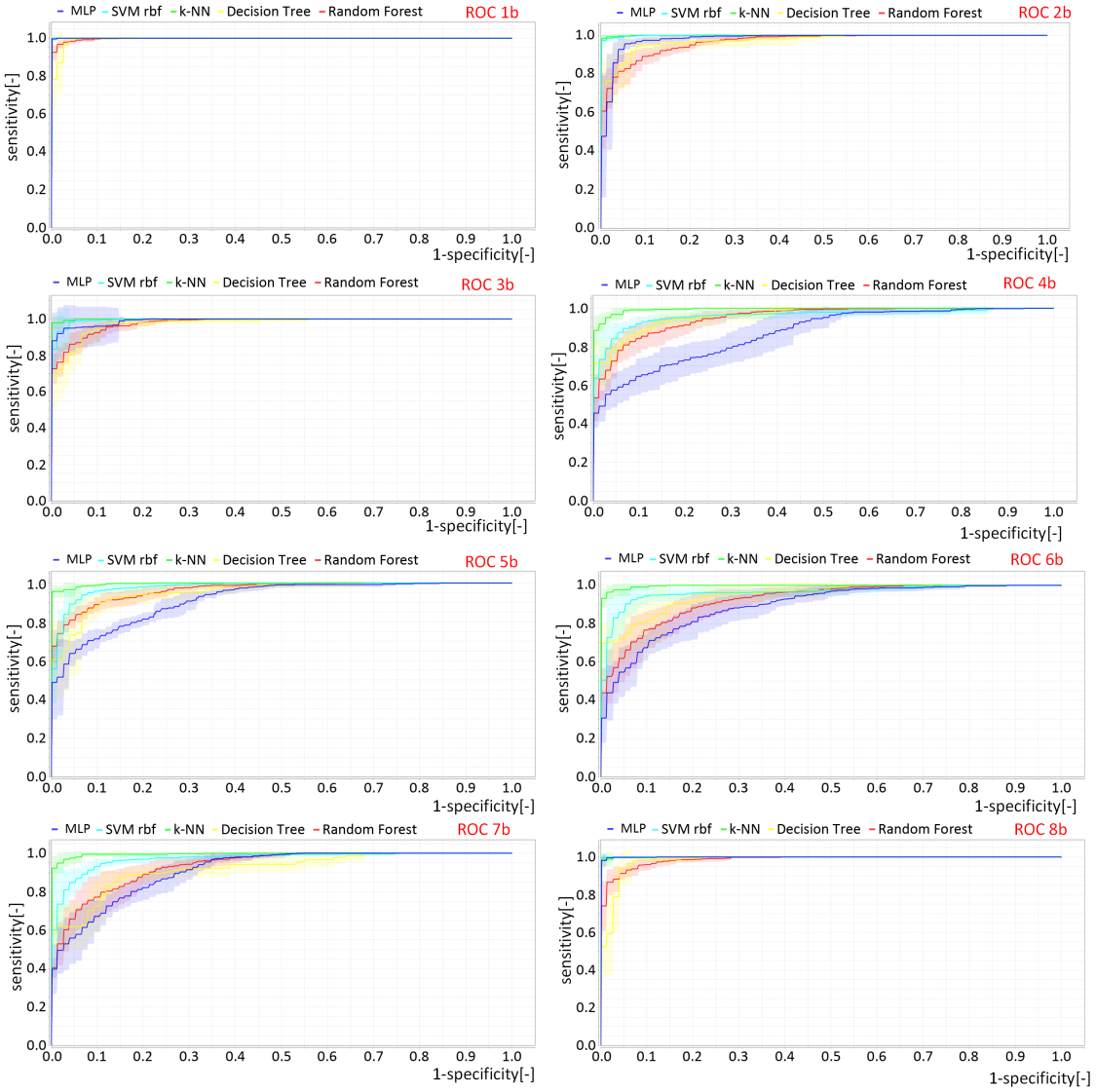


Fig. 3.44: ROC analysis for individual bits of DS1.

previous case, k-NN, SVM-rbf and MLP provided the best plots in ROC space. Moreover, the influence of the most crucial step of profiling PA, that lies in selecting the interesting points, is demonstrated by comparing Fig. 3.45 and Fig. 3.46, because the datasets DS2 and DS3 were prepared from identical raw power traces. In Fig. 3.46, the plots of ROC curves are more distant from (0,1), therefore the selection was performed less precisely and the model created provided lower performance. We conclude that if the selection of interesting points is properly performed, every possible distinguisher will provide more or less similar results regardless of the underlying technique deployed (either classic templates or machine learning after optimization).

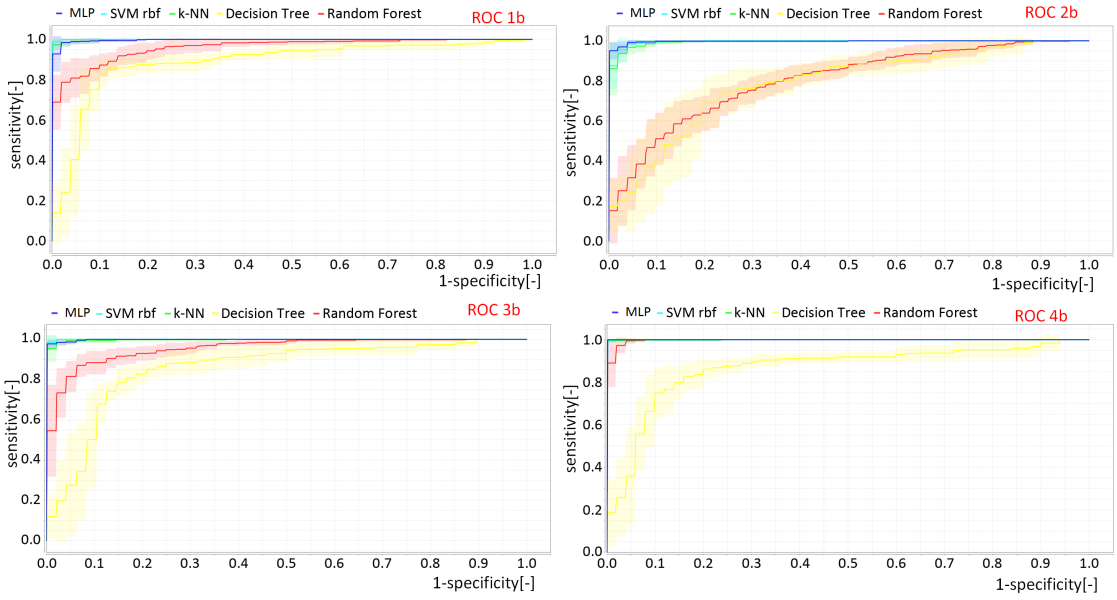


Fig. 3.45: ROC analysis for individual bits of DS2.

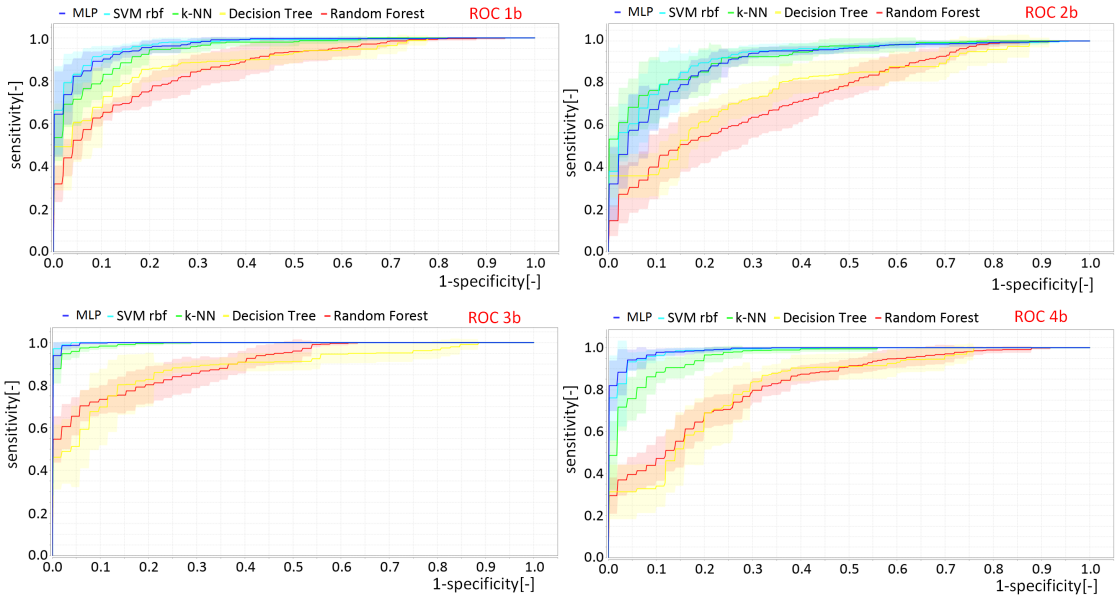


Fig. 3.46: ROC analysis for individual bits of DS3.

3.4.4 Summary

In this research section, we provided an extensive comparison of widely used machine learning algorithms in power analysis such as SVM, decision tree, and MLP including the new approach based on k-NN. We implemented a verification program that chose optimal settings of the parameters of individual machine learning algo-

rithms in order to obtain the best classification accuracy. Based on the obtained results, we can consider SVM-rbf, MLP and k-NN as the most suitable candidates for profiling power analysis attacks (in terms of classification accuracy). Generally, we can optimize every ML algorithm using parameter settings to get almost identical classification results. On the other hand, optimization of individual ML algorithms can be time consuming (possible differences can be enormous based on the selected parameters and algorithms, for example 6 minutes and 35 seconds was needed for k-NN optimization and 8 days for SVM-poly optimization).

Moreover, we investigated a success rate of the masks revelation depending on the number of interesting points and the number of power traces. As expected, the higher the number of traces and points in the learning set, the higher the accuracy of every power analysis attack implemented. The main finding was that the sharp rise in success rate of the ML attacks (MLP and SVM) occurs much earlier than for every TA attack. We can conclude that the ML approach is a much more effective profiling power analysis attack for a small number of power traces and interesting points. In other words, it is better to use profiling power analysis attacks based on the MLP if the adversary has only a limited number of measured power traces than to realize attacks based on templates.

From all realized experiments, we see a really good potential of the k-NN algorithm. The approach proposed based on the simplest k-NN algorithm can provide important advantages to the attacker compared with other profiling attacks. We summarize these observations in the following points:

- the basic principle of the method is very simple, the profiling phase constitutes only storing of measured data in the memory (the attacker has to realize this in any case),
- it is not necessary to prepare (calculate) templates, the attacker can save time and memory,
- the k-NN approach is implemented by default in many program environments, therefore it is not a problem for the attack implementation,
- the success rate is comparable with the template attack, the k-NN approach corresponds with the reduced TA that does not consider the covariance matrix,
- the attacker can use more interesting points compared with TA where it is limited due to memory limitations resulting from the covariance matrix,
- the k-NN does not include a learning phase compared with other MLs, therefore the attacker can work more efficiently (fast response to all changes related to the power traces measured, number of interesting points, size etc.).

4 Protected Hardware Implementation

The simplest way to increase the noise of operations is to perform several independent operations in parallel (the background of hiding techniques is described in section 2.3.1). The important fact is to utilize independent operations and a hardware platform for cryptographic algorithms with a wide data-path. This section presents a hardware implementation proposal where four parallel AES cores and a 512 bit data-path are utilized in order to protect the implementation. This section is a scientific contribution of the thesis and the amended version of the text below is a part of the author's paper at the ASHES'18 ACM conference [154].

We present the architecture and implementation of our encryption system designed for 200 Gbps FPGA (Field Programmable Gate Array) network cards utilizing the IPsec (IP security) protocol. To our knowledge, our hardware encryption system is the first that is able to encrypt network traffic at the full link speed of 200 Gbps using a proven algorithm in a secure mode of operation, on a network device that is already available on the market. Our implementation is based on the AES (Advanced Encryption Standard) encryption algorithm and the GCM (Galois Counter Mode) mode of operation, therefore it provides both encryption and authentication of transferred data. The design is modular and the AES can be easily substituted or extended by other ciphers. We present the full description of the architecture of our scheme, the VHDL (VHSIC Hardware Description Language) simulation results and the results of the practical implementation on the NFB-200G2QL network cards based on the Xilinx Virtex UltraScale+ chip. We also present the integration of the encryption core with the IPsec subsystem so that the resulting implementation is interoperable with other systems.

The demands on communication systems are tremendous and continuously growing as new online services are being introduced. The speed of communication networks rose from recent hundreds of Mbps to hundreds of Gbps. The fastest modern network adapters allow communication at the speed of 200 Gbps over a single network interface. As an example, the network card NFB-200G2QL [136] offers 2×100 Gbps with full duplex technology for end users. The very recent appearance of smart infrastructures, such as smart grids, smart cities and IoT networks, even changed the paradigms of communication. While during the last decade the developers focused on novel solutions for high-speed backbone networks with only a few endpoints, the situation is different today.

We face the problem of connecting millions of simple devices that all communicate over the network, sometimes with only a single central node. That is the case for sensor networks, cloud infrastructures and smart grid infrastructures, to name a few examples. As a result, it is not sufficient to provide only fast network devices

but also devices that can handle millions of simultaneous individual connections. To fulfill the ever-growing demands on both speed and flexibility, hardware-accelerated solutions are usually used. Typically, fast network cards based on FPGA chips are employed. By using hardware acceleration, a throughput of hundreds of Gbps can be achieved on a single device [136].

In recent years, cybersecurity also started to play a crucial role. To assure the security of communication networks, cryptographic mechanisms that are able to provide data confidentiality, integrity and authenticity need to be implemented. In most systems, mechanisms based on symmetric cryptography, in particular the AES encryption algorithm standardized by NIST (National Institute of Standards and Technology) [1], are used thanks to their proven security and high efficiency. However, encryption algorithms are usually too complex for high-speed implementations, so the performance of the whole communication chain is usually limited by the encryption subsystem. To avoid overloading the central CPUs of the systems and speed up the communication systems, the security functions are more and more offloaded to FPGA network cards.

In this paper, we describe the full architecture of the FPGA encryption subsystem that is able to encrypt data at the full line speed of 200 Gbps. We use the standard AES encryption algorithm in the GCM block cipher mode of operation [29] that is recommended by NIST. By using the GCM mode, we are able to assure both the confidentiality and authenticity of transferred data. Our system is modular, the choice of the encryption algorithm is open and other algorithms than AES can be used. In addition to the pipelined high-speed encryption core, we also present the architecture of the supporting modules, in particular those allowing the integration with the IPsec-based implementations. IPsec is the most common standard for creating VPN tunnels in high-speed networks, so its support makes our solution interoperable with other products. Furthermore, we enhance our implementation by adding the support of tamper-proof devices used for the storage and handling of sensitive cryptographic material, particularly digital certificates and pre-shared keys (PSK).

To prove the practical impact of our design, we present the first results obtained from the implementation on the new NFB-200G2QL FPGA network cards based on the Xilinx Virtex UltraScale+ chip [142]. We present the results of the VHDL simulation, the frequencies required to achieve 200 Gbps, the hardware utilization and other implementation aspects.

4.1 State of the Art

The idea of using hardware for accelerating encryption is not entirely new. There are several implementations of block ciphers already available on FPGA cards. Lemsitzer et al. [65] described the implementation of AES in the GCM mode on FPGA Virtex-4 cards and achieved 15.3 Gbps throughput back in 2007. Soliman et al. [133] used parallelisation techniques to speed up the AES on a Virtex-5-based card and achieved almost 74 Gbps. The authors in article [121] reach a throughput of 73.737 Gbps based on a clock frequency of 576.07 MHz. More recent papers from 2014 [32] and 2017 [33] describe architectures based on Virtex-5 and Spartan chips providing speeds up to 86 and 113 Gbps respectively. Some papers, such as [60], focus more on the efficiency and lightweight design of the implementation than its performance. However, the above mentioned results remain only theoretical because the frequency is higher than 300 MHz, which is not practically achievable on FPGAs currently available on the market. Smekal et al. [131] describe an encryption scheme and its implementation on Virtex-7 cards that is able to encrypt 5.1 Gbps traffic on a frequency of 100 MHz. Article [45] focuses on AES-GCM, but there is no real HW implementation. The paper presented in [16] describes an architecture for highly parallel implementations of AES-GCM with a throughput of 482 Gbps using a single Xilinx Virtex Ultrascale. Nevertheless, the firmware and real implementation are missing. The article in [82] presented a 100 Gbps AES-GCM encryption system working at a frequency of 100 MHz but due to a flaw in the basic architecture it is not possible to realize the implementation on a FPGA card[1]. Vliegen et al. [139] implemented side-channel-protected AES-GCM system on Virtex-7. The architecture provides a throughput of 15.24 Gbps at a clock frequency of 119 MHz, however, the work is more focused on the implementation of the Boolean masking countermeasure scheme and does not take into account the application in IPsec. In many other cases, the authors describe the implementation of the AES algorithm, not taking into account that the cipher must work in a mode of operation to be practically secure.

An overview of existing FPGA implementations and their implementation properties is given in Table 4.1. In particular, it is important to compare our implementation to those solutions that take into account a proper mode of operation and the integration of the algorithm in a larger system, consisting of firmware that supports e.g. IPsec.

[1]Parallel pipelined architecture proposed in the paper is not practically functional due the calculation of the Galois field multiplication needs input data every clock cycle.

Tab. 4.1: Comparison of current work.

Reference	AES	GCM	IPsec	Frequency [*Mhz*]	Throughput [*Gbps*]	Firmware	Year
Lemsitzer et al.[65]	✓	✓	✗	120	15.3	✓	2007
Nguyen et al. [108]	✓	✗	✓	112	3.15	✗	2018
Korona et al. [59]	✓	✗	✓	100.65	10.23	✗	2017
Soliman et al. [133]	✓	✗	✗	557	74	✗	2011
Farashahi et al. [32]	✓	✗	✗	671	86	✗	2014
Farooq et al. [33]	✓	✗	✗	886.64	113.5	✗	2017
Koteshwara et al. [60]	✓	✓	✗	50	0.417	✓	2017
Shou et al. [121]	✓	✗	✗	576.07	73.7	✗	2009
Smekal et al. [131]	✓	✗	✗	100	5.1	✓	2016
Vliegen et al. [139]	✓	✓	✗	119	15.24	✗	2017
Buhrow et al. [16]	✓	✓	✗	314	482	✗	2015
Henzen et al. [45]	✓	✓	✗	233	119.3	✗	2010
Martinasek et al. [82]	✓	✓	✗	200	102.4	✗	2017
Our work	✓	✓	✓	200	200	✓	2018

4.2 Contribution

We provide the full description of the architecture and the implementation results of a system that differs from all the existing systems mainly in the following aspects:

- **Speed**: our system is able to encrypt or decrypt *200 Gbps* traffic at full line speed, at an operating frequency of *200 MHz* (i.e. a realistic operating frequency for state-of-the-use FPGAs). Compared to the state-of-the-art implementations, our proposal is almost twice faster and, more importantly, practically implementable on off-the-shelf hardware.
- **Security**: we describe not only the implementation of the AES core, but also of all the necessary supporting mechanisms, such as the authenticated key agreement and expansion, the *GCM* mode of operation and the *IPsec* integration. These mechanisms are necessary for a practical deployment.
- **Flexibility**: our architecture is *modular*, thus replacing the encryption algorithm is fast and easy.
- **Usability**: our results are not only theoretical, they have very practical impacts. To prove the usability in the real world, we show the implementation of our encryption subsystem on the off-the-shelf 200 Gbps NFB-200G2QL FPGA

network card from Netcope Technologies [136]. This makes our solution the first demonstrable 200 Gbps AES-GCM implementation on a commercial network card, ready for being used in a realistic network setting.

4.3 Preliminaries and System Architecture

In most real-world applications, end users utilize high-speed encryption systems as encryption and decryption gateways. For this reason, our system is compliant with the IPsec protocol in order to achieve easy deployment in practice. Before the basic architecture is introduced, we briefly define the fundamental terms and concepts that are used in the IPsec protocol.

IPsec ensures data encryption, authentication and integrity of each IP packet in a communication session by utilizing the ESP (Encapsulating Security Payload) [57] and AH (Authentication Headers) [56] modes. ESP and AH rely on various symmetric cryptographic primitives and keys to provide security services. The end station can use different ciphers and keys for different data connection sessions. In IPsec terminology, the choice of ciphers and keys is defined as a Security Association (SeAs) and each station handles a special database where this information is stored. In fact, SeAs require two databases: a security policy database (SPD) and a security association database (SAD). The SPD stores the security requirements and policy requisites that should be met. The SAD contains the individual parameters of each active SeA. The initial entry in the database is often inserted by the Internet Key Exchange (IKE) protocol [130]. At the beginning of the communication between two entities, the IKE protocol selects the most secure cryptographic algorithm supported, authenticates the entities during the next step and establishes the set of secret keys for authentication and encryption services. Authentication can be realized by employing digital certificates (X.509) or pre-shared keys (PSK). The outcome of the IKE protocol is a SeA record that includes supported cryptographic algorithms and established keys (also referred to as master keys MKs) from which all the necessary keys are derived.

The architecture of our system is depicted in Fig. 4.1 and consists of two main subsystems, the *Authentication Subsystem* and the *FPGA Subsystem*. At the beginning of the communication between user A and B, an initial record in the SAD is created by a software application and a smart card[2]. For this purpose, the IKE version 2 (IKEv2) protocol is implemented that provides authentication based on

[2]The use of a smart card is optional and recommended for critical applications. The security of the smart card (e.g., to side-channel attacks) is out of the scope of this paper.

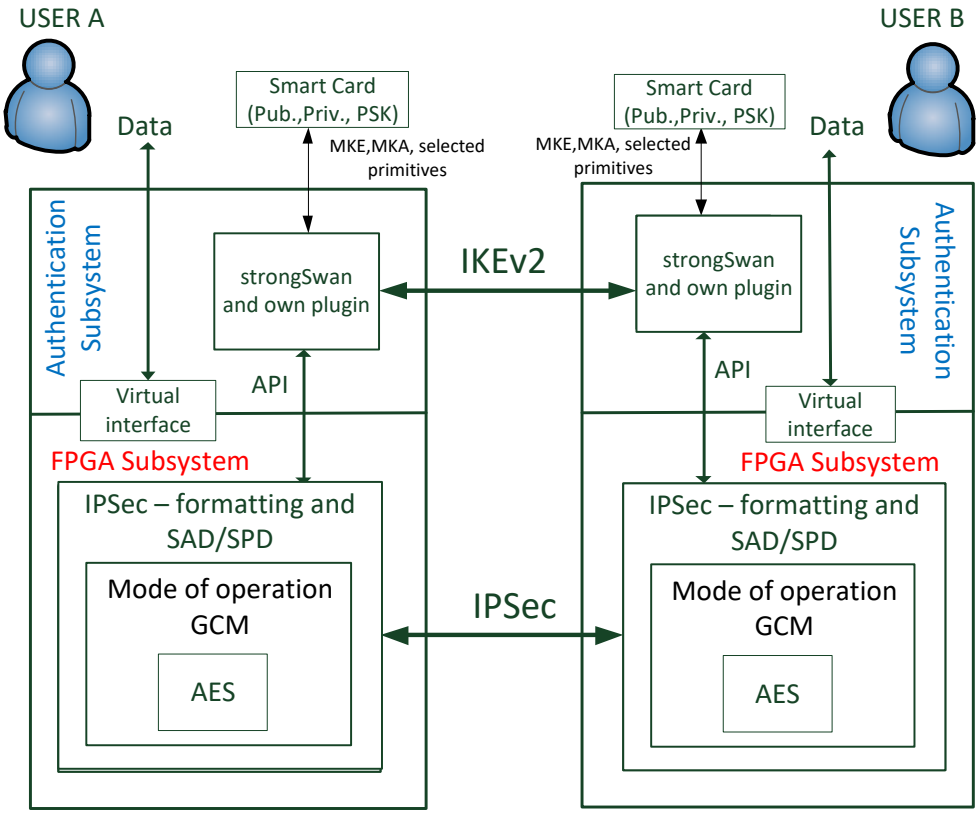


Fig. 4.1: Basic architecture of the proposed system.

digital certificates or pre-shared keys. An important aspect of the final system security is to keep private keys and the pre-shared keys secret. In our system proposal, this crucial requirement is accomplished by employing a smart card as a SAM (Security Access Module). The smart card never discloses the sensitive material to an external application, compared to standard operating systems, in which the sensitive material is mostly stored in configuration files in plain. That poses a risk which prevents systems based on a standard OS from being used, e.g., in critical infrastructures of governments. The authentication using a smart card is realized only at the beginning of the communication and takes less than 20 ms. After this action, the system makes use only of the FPGA Subsystem performing IPsec formatting, encryption/decryption and rekeying operations. For this reason, the usage of the smart card is not an obstacle in reaching the desired speed of 200 Gbps. At the end of the authentication phase, the generated data including the master encryption key denoted as MKE (Master Key Encryption), a master key for authentication denoted as MKA (Master Key Authentication) and the specification of cryptographic primitives are forwarded through an API (Application Programming Interface) to the FPGA Subsystem. The result of the authentication phase is the record in the SAD.

In our implementation, the encryption process is realized by the AES-GCM algorithm. The FPGA subsystem receives the user's data through a virtual interface (VI). Data are formatted according to the IPsec specification and encrypted. The FPGA subsystem consists of three individual components that are described in more detail in Sect. 4.5. The receiving entity operates analogically. The data stream is decrypted in FPGA and forwarded to VI.

4.4 Authentication Subsystem Implementation

In the first step of our research, we analysed the available implementations of IPsec and IKE. We selected strongSwan as the appropriate library for the implementation that fulfills all the requirements: open-source software, support of the IKEv2 protocol and the support of PKCS#11. The main purpose of the authentication subsystem is to provide secure mutual authentication and key establishment between two communication parties using smart cards (i.e. to create the initial record in SAD).

The authentication subsystem supports two types of authentication methods in which smart cards play the role of the SAM:

- **Public Key Infrastructure (PKI)-based smart card authentication**: smart cards supporting PKCS#11 (e.g., CardOS smart cards) store digital certificates and PKCS#15 file structures and provide authentication and key establishment based on certificates and PKI. The software application (in this case, the strongSwan IKEv2 implementation) loads a digital certificate from the smart card via a `pkcs11` plugin using the `OpenSC` and `pcsc-lite` libraries. Certificates (X.509) are used for mutual authentication and key establishment. The certificates and private keys are generated by external software, such as `openssl`, and are securely installed into the smart cards using the pkcs15-init utility. The private keys can be also generated directly on the smart card device by the pkcs15-init utility. Nevertheless, due to the recently discovered attacks [107] on several smart cards, we recommend the external key generation. All smart cards that support OpenSC and pkcs15-init utilities should be protected by PIN.
- **Pre shared keys (PSK)-based smart card authentication**: because the PSK is stored in a plain text in the default implementation of strongSwan, we developed our own implementation written in the Java programming language for Java cards. Our Java card applet runs on Java cards with Java Card OS JCAPI 2.2.2 or higher. The Java Card application uses the PSK to verify authentication parameters exchanged by IKE_SeA_INIT and IKE_AUTH phases. Based on the IKE parameters, the software application written in

the C programming language (in this case, the modified strongSwan library with our plug-in utilizing library `pcsc-lite-devel` for communication with programmable smart cards via APDU) establishes an encryption key (MKE), authentication key (MKA) and selects cryptographic primitives.

The communication between the smart card and the application is provided via Application Protocol Data Unit (APDU) messages. The smart card is physically connected to the Universal Serial Bus (USB) reader by the ISO/IEC 7816 contact interface.

4.5 FPGA Subsystem Implementation

For the final implementation, we selected the NFB-200G2QL FPGA network card that is a product of Netcope Technologies. The board is based on the FPGA chip Xilinx Virtex UltraScale+ with 2× QSFP28 cages supporting 2×100G Ethernet ports and offering three modules of QDRIIIe SRAM (Static Random Access Memory) memory. The network card is connected to a host computer via 2× Peripheral Component Interconnect Express (PCIe) bus generation 3.0 x16 for high-speed data transfer. The theoretical throughput of one PCIe bus is 128 Gbps. The NFB-200G2QL card handles network data at a speed of 200 Gbps over two PCIe buses for various Ethernet frame lengths[3]. Therefore, the PCIe buses do not represent the bottleneck in the system. The Virtex UltraScale+ chip provides 1 728 000 system logic cells, contains 788 160 Look-Up-Tables (LUTs) and 1 576 320 Flip-Flop (FF) registers. The actual appearance of the network card is depicted in Fig. 4.2.

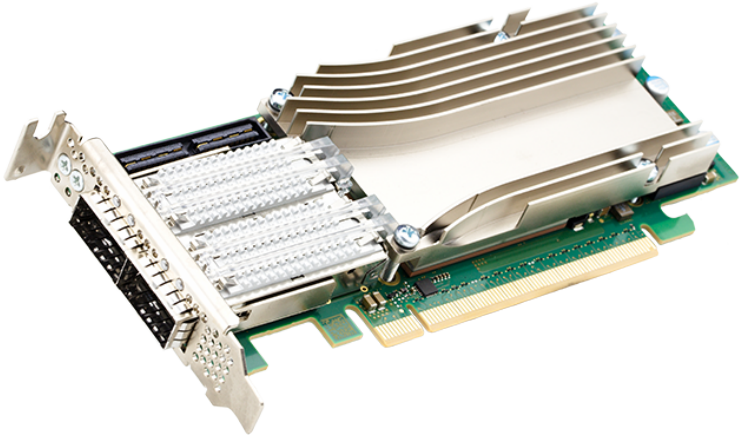


Fig. 4.2: NFB-200G2QL FPGA network card [136].

[3]Interested readers can obtain more detailed information in technical documentation https://www.netcope.com/getattachment/91efe95a-644f-4cd0-9a2f-6443d532910c/Improving-DPDK-Performance.aspx

In order to implement the FPGA subsystem, we used the Netcope Development Kit (NDK) as a platform for a high-speed application development. NDK provides a set of integrated tools and interfaces that makes the development of individual applications much easier. The integral part of the NDK are drivers, libraries and kernel modules that provide communication between the hardware part running on the FPGA card and a user software application running on a host computer (in our case, the authentication subsystem based on the strongSwan and a smart card). We implemented three individual components in the application core of the NDK:

- The first component is labeled **IPsec**. This component implements packet formatting and IPsec databases (SAD and SPD).
- The second component is **GCM**. That is the implementation of the operation mode according to the NIST recommendation 800-38D [29].
- The last implemented component is **AES** which consists of all functions of the Advanced Encryption Standard supporting key lengths of 128 and 256 bits.

The block diagram of the NDK framework including our components is depicted in Fig. 4.3.

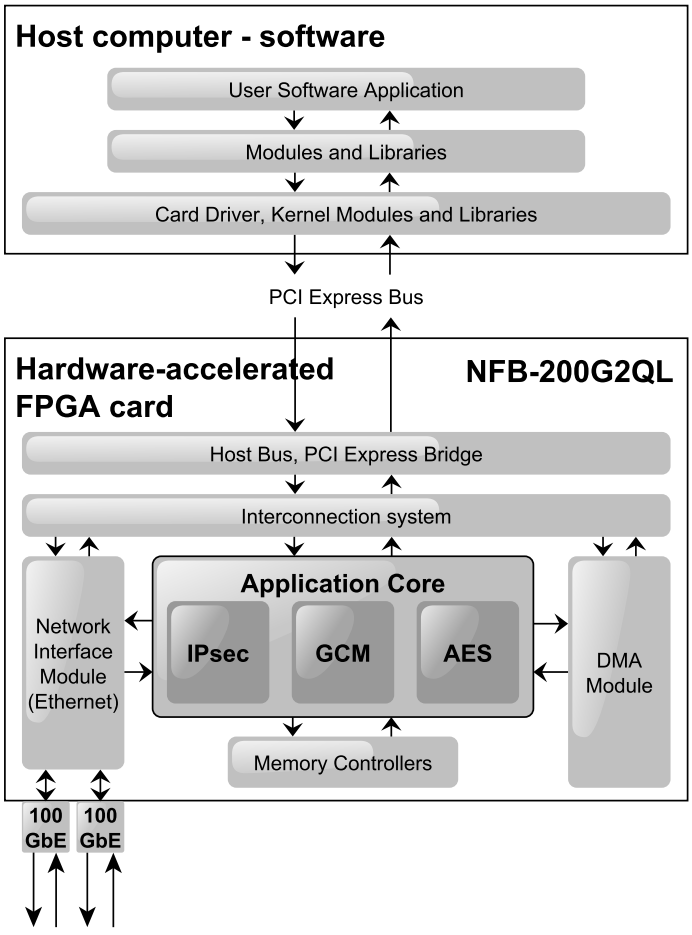


Fig. 4.3: Scheme of implemented components.

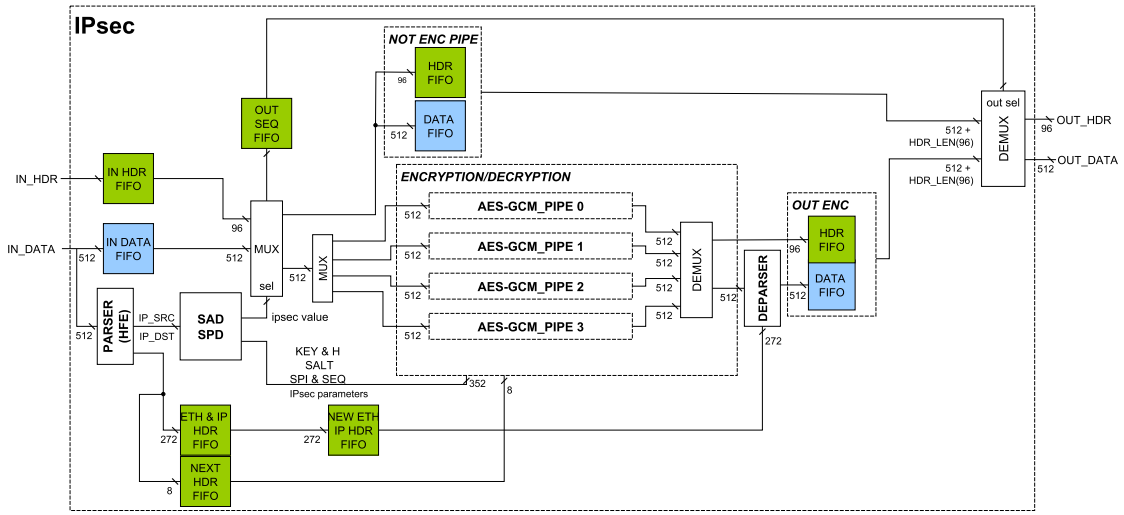


Fig. 4.4: Block scheme of the IPsec component (used 2x in our implementation).

The final implementation of hardware components was realized using the VHDL language. The functionality was verified using the Xilinx's Vivado simulator tool in version 2017.4. The implementation of the AES and GCM components has been fully described in our past paper [82] including the verification using the test vectors [98], the results obtained confirmed the correctness of the implementation. Therefore this section focuses mainly on the description of the IPsec component.

The block diagram of our IPsec hardware implementation is depicted in Fig. 4.4. The main components of packet processing are two components labeled as *PARSER* and *DEPARSER* that provide parsing and deparsing of the network data stream according to the IPsec specification. The function of these components is based on *HFE* (Header Field Extractor), which is a function of the NDK [120]. In the first step, the PARSER obtains Ethernet and IP headers and these values are stored in the FIFO (First In First Out) shift register (labeled as ETH & IP HDR FIFO and NEXT_HDR FIFO). The PARSER forwards the source and destination IP addresses to the SAD/SPD database where a decision is made on the encryption/decryption process. The implementation of the databases is based on Cuckoo hashing [114] and is realized directly on-chip utilizing BRAM (Block Random Access Memory). Access to the database takes one clock cycle and the limitation is ten thousand records in each table. If no match is found for IP addresses in the database, the network traffic is forwarded to *NOT ENC PIPE* and passes through the FPGA network card without further processing. If a match is found, the traffic is forwarded to the *Encryption/Decryption* block. This component consists of four *AES-GCM_pipe* components that provide mainly data alignment, padding and encryption or decryption.

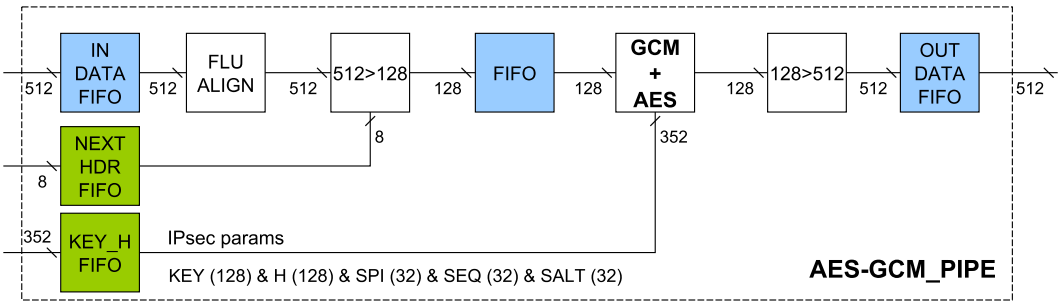


Fig. 4.5: The scheme of the application core of the encryption system.

The block diagram of this component is depicted in Fig. 4.5. The encrypted or decrypted output data are connected to demultiplexer and the *DEPARSER* assembles new headers together with the encrypted payloads. In the last step, the IPsec packet is forwarded to the output port. Fig. 4.6 presents the data flow diagram of our testbed. In fact, we have to instantiate two IPsec components to achieve the required speed of 200 Gbps. Since the IPsec component works in one direction, achieved speed of 200 Gbps is unidirectional. Due to resource utilization of the components of the NDK, it is not possible to instantiate more than two IPsec components on used FPGA chip. As a consequence, either 200 Gbps of unidirectional encryption (or decryption) or 100 Gbps of encryption and 100 Gbps of decryption can be performed, depending on the mode of instantiated IPsec components.

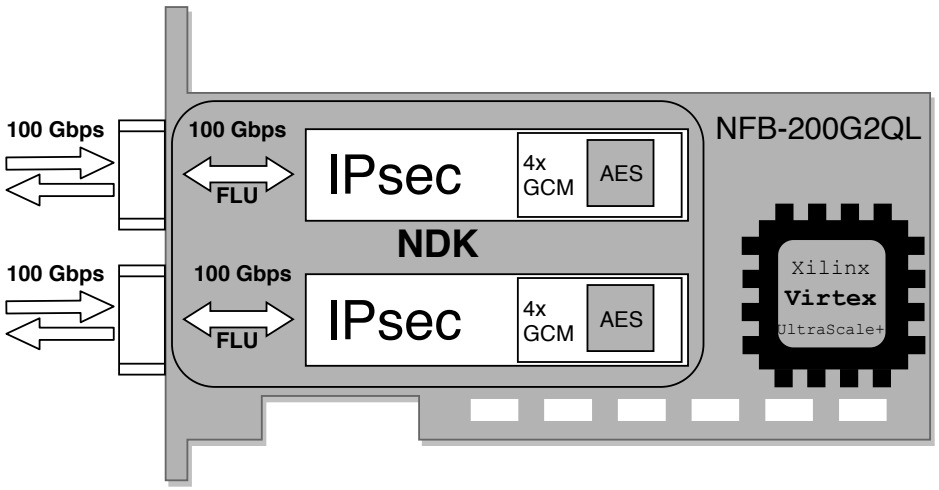


Fig. 4.6: Data flow diagram on the network card.

Vivado was used to synthesize the final firmware of the whole system. The resulting firmware for a network card consists of the following parts: IPsec, AES and GCM. In addition, the firmware includes components of the NDK framework that arrange communication through network interfaces and the PCI Express bus.

Tab. 4.2: Utilization and performance per component.

Component	Frequency (MHz)	LUT [-]	FF [-]	Throughput (Gbps)
AES	378	10 125	2 946	48.38
GCM	245	8 973	1 238	31.36
AES (norm.) [4]	200	9 920	1 282	25.6
GCM (norm.) [4]	200	3 640	1 475	25.6
IPsec	200	8 457	6 913	25.6
SAD/SPD	200	2 308	1 078	25.6
NDK	200	99 713	85 437	204.8

The utilization of the Virtex UltraScale+ chip is summarized in Table 4.2. Frequency denotes the maximum frequency at which the unit is able to run. Moreover, we determine and investigate the maximum throughput T of individual components as follows:

$$T = f * N_b \tag{4.1}$$

where f denotes the maximum frequency and N_b represents the width of the data bus in bits. For example, the implementation of the AES component can work at the maximum frequency of 378 MHz and the width of the data bus is 128 bits, thus the maximum throughput of the encryption could be 48.38 Gbps according to the equation 4.1.

However, for a practical deployment, the crucial limitation factor is the maximum frequency of the slowest hardware component. In our case, the limitation factor is the NDK that works on the frequency of 200 MHz. To achieve the full throughput of 200 Gbps using 200 MHz components, we had to deploy eight parallel AES-GCM components within two IPsec components (Fig. 4.6 and Fig. 4.4). Table 4.3 summarizes resource utilization of all components including NDK synthesized for Virtex UltraScale+XCVU7P chip. The proposed system takes 80.5 % of available LUT resources (Logic + Memory), which is close to the maximum occupation in practice, given that the routing would become congested with a larger percentage of occupation. Altogether, the system makes use of 10.5 % of the available flip-flops and 31.1 % of the available BRAM. The final architecture was easily synthesizable on the chip using the available resources. The encryption/decryption is fully functional at a speed of 200 Gbps, using a frequency of 200 MHz. Therefore it is not necessary to increase the frequency or to choose any other, possibly more powerful and expensive, chip.

[4]Normalized implementation for frequency 200 MHz.

Tab. 4.3: Hardware Utilization.

	LUT as Logics	LUT as Memory	Register as Flip Flop	BRAM
Available	394 560	394 560	1 576 320	1 440
AES 8×	79 360 20.1 %	32 <0.1 %	10 256 0.7 %	—
GCM 8×	29 120 7.4 %	2 072 0.5 %	11 800 0.7 %	—
IPsec 2×	72 796 18.4 %	7 255 1.8 %	56 447 3.6 %	131 9.1 %
SAD/SPD	4 616 1.2 %	130 <0.1 %	2 156 0.1 %	88 6.1 %
NDK	99 713 25.3 %	20 961 5.3 %	85 437 5.4 %	229 15.9 %
Total	285 605 72.4 %	30 450 8.1 %	166 096 10.5 %	448 31.1 %

4.6 Summary

We presented the architecture and the results of the practical implementation of a hardware encryption system that is able to encrypt data at 200 Gbps on a single device. To the best of our knowledge, this is currently the fastest implementation of a practically demonstrable secure authenticated encryption scheme on a commercial FPGA network card. Our design is extremely efficient and requires clocking at only 200 MHz, which is very promising for a future speed increase. Besides the theoretical design and practical implementation results, we also provided details about the integration with the necessary supporting modules, namely the IPsec subsystem and the authentication subsystem based on smart cards.

Final hardware implementation was realized utilizing the NFB-200G2QL FPGA network card, therefore it was not possible to measure the real power consumption. However, we created the power traces based on the simulation to observe the theoretical leakage of the realized hardware implementation. In the following text, we describe the main findings. In order to prove the functionality of masking proposed

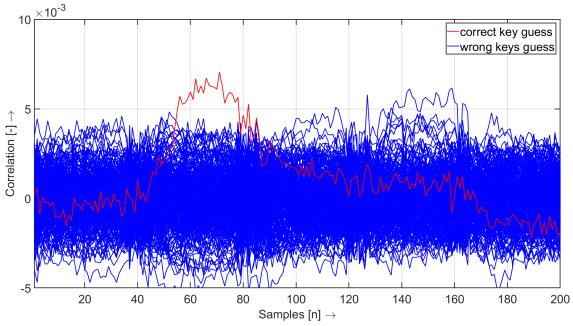


Fig. 4.7: Result of the CPA attack for unprotected HW implementation.

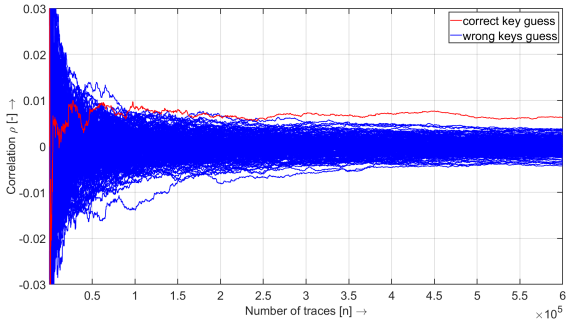


Fig. 4.8: Size of the correlation for unprotected HW implementation.

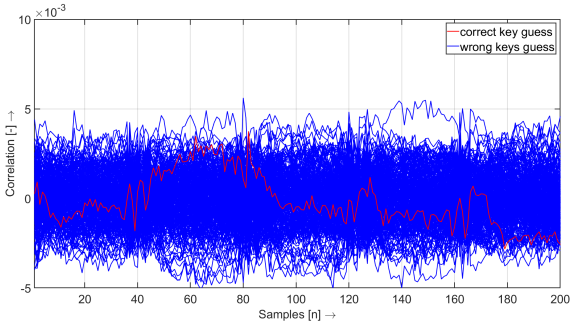


Fig. 4.9: Results of the CPA attack for parallel HW implementation.

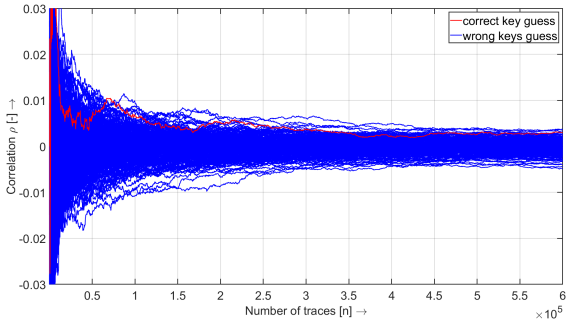


Fig. 4.10: Size of the correlation for parallel HW implementation.

scheme, we realized the power analysis of unprotected (non parallel) AES hardware implementation. In the next step, we compare the obtained results with power analysis targeted at hardware implementation proposed protected by hiding. Note that no masks are utilized during the encryption/decryption process only the four parallel AES core are implemented.

Same as in the previous examples, we attack the non-linear byte substitution (SubBytes) of the AES in the first round. The power traces corresponding to the whole process of encryption were created by simulation. Created power traces together with the corresponding plain text and cipher text were imported to the Matlab where the power analyses based on correlation coeficient were realized. Altogether, $600,000$ power traces were created for the encryption key: $K_{s1} = [42, 138, 236, 244, 69, 67, 231, 207, 141, 31, 115, 14, 106, 251, 199, 152]$.

Obtained results are depicted in Figures 4.7, 4.10, 4.9 and 4.10. We can observe the following facts. The correct key guess for unprotected implementation can be revealed without any problem based on 250 000 power traces (Fig. 4.7 and 4.10). If we compare the results for protected implementation (Fig. 4.9 and 4.10), the

attacker can not reveal the value of the first secret key byte even with the use of maximum 600 000 power traces. We demonstrate results for the first secret key byte, however the identical situation occurs for remaining bytes of the encryption key. We can conclude that proposed countermeasure techniques works and brings resistance of the implementation at least for 600 000 power traces. In practice, the attacker can measure more power traces to try reveal the secret. However, success is not guaranteed if leakage is suppressed by parallel processing. It would be appropriate to repeat the power analysis of protected implementation based on 2 million power traces.

5 Conclusion

This habilitation thesis guides the reader through the power analysis fundamentals including the countermeasure techniques. Moreover, the thesis focuses on practical aspects of the protected implementation and description of the possible attacks. The results and observations try to support the realization of protected cryptographic algorithms implementation in future. This will be possible, because the readers will understand the underlying concepts of the power analysis, how to protect the implementation and how to evaluate the real leakage of the cryptographic device. The expected contribution of the thesis is both pedagogical and scientific.

The ***pedagogical contribution*** is mostly addressed in Chap. 2. In the first two introductory sections (Sec. 2.1 and Sec. 2.2), we describe the fundamentals of the power analysis methods. In particular, we explain the principles of profiling power analysis attacks utilizing the standard Gaussian approach, profiling based on machine learning and non-profiling power analysis attacks based on the Correlation coefficient and Difference of Means. This knowledge introduces the basic building blocks to understand the main principle of the power analysis. In order to design the secure implementation, it is crucial to understand the essence of the possible attacks. The following theoretical section (Sec. 2.3) describes the basic countermeasure techniques that are divided into two basic groups: masking and hiding. However, these techniques can also be attacked very easily in practice, therefore it is crucial to pay attention to the correct implementation of the countermeasure. This issue is addressed in Sec. 2.4 and in Sec. 3.1, where we describe masking and hiding countermeasure techniques including the power analysis from a practical point of view. More precisely, the Boolean masking and the shuffling of crucial operations of AES is attended in our education text as well as a short current state description. As an elementary example, we bring in to play the DPA Contest because it is world wide known and freely available. Therefore, the reader can verify the obtained results which is the best way to understand the explained issues. Parts of the chapter are used in a university textbook of the Information Security study program at Brno University of Technology, where the author is involved. Moreover, the contents of the sections was presented by the author at invited lectures, for the Military Research Institute and the Smart Cards & Devices Forum.

The ***scientific contribution*** is addressed mainly in Chap. 3 and Chap. 4. These chapters contain various results from selected author's publications that were published in journals with an impact factor. In this part (Sec. 3.2), we investigate the security of improved protected implementations of AES. Our analysis, focused on exploiting the first-order leakage, discovered mistakes that can be misused in order to recover the whole secret key. Moreover, we focus on finding which profiled

attack has the lowest sensitivity to modifications of the characteristics of leakages (3.3 and 3.4). This is the contribution that reflects the real world situation because datasets often suffer from errors or distortions in the measured leakages. In the last chapter (Chap. 4), we propose the hardware implementation where hiding is based on four parallel encryption cores and a 512 bit data-path. The text presents the full description of the architecture, simulation results and the results of the practical implementation based on the Xilinx Virtex UltraScale+ chip. All defined goals of the habilitation thesis were fulfilled.

Bibliography

[1] Federal information processing standards publication (FIPS 197). Advanced Encryption Standard (AES), 2001.

[2] Akkar, M.-L., Bevan, R., Dischamp, P., and Moyart, D. Power analysis, what is now possible... In *Advances in Cryptology - ASIACRYPT 2000* (2000), T. Okamoto, Ed., vol. 1976 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 489–502.

[3] Altman, N. S. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician 46*, 3 (1992), 175–185.

[4] Anderson, J. R., Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. *Machine learning: An artificial intelligence approach*, vol. 2. Morgan Kaufmann, 1986.

[5] Archambeau, C., Peeters, E., Standaert, F.-X., and Quisquater, J.-J. Template attacks in principal subspaces. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, L. Goubin and M. Matsui, Eds., vol. 4249 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 1–14.

[6] Aumonier, S. Generalized correlation power analysis. In *Proceedings of the Ecrypt Workshop Tools For Cryptanalysis* (2007), vol. 518.

[7] Bar, M., Drexler, H., and Pulkus, J. Improved template attacks. In *COSADE 2010 - First International Workshop on Constructive Side-Channel Analysis and Secure Design* (2010), pp. 81–89.

[8] Bartkewitz, T., and Lemke-Rust, K. Efficient template attacks based on probabilistic multi-class support vector machines. In *Smart Card Research and Advanced Applications* (2013), S. Mangard, Ed., vol. 7771 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 263–276.

[9] Batina, L., Hogenboom, J., and van Woudenberg, J. G. J. Getting more from pca: First results of using principal component analysis for extensive power analysis. In *Proceedings of the 12th Conference on Topics in Cryptology* (Berlin, Heidelberg, 2012), CT-RSA'12, Springer-Verlag, pp. 383–397.

[10] Benadjila, R., Prouff, E., Strullu, R., Cagli, E., and Dumas, C. Study of deep learning techniques for side-channel analysis and introduction

to ASCAD database. *ANSSI, France & CEA, LETI, MINATEC Campus, France. Online verfügbar unter https://eprint. iacr. org/2018/053. pdf, zuletzt geprüft am 22* (2018), 2018.

[11] Bhasin, S., Bruneau, N., Danger, J.-L., Guilley, S., and Najm, Z. Analysis and Improvements of the DPA Contest v4 Implementation. In *Security, Privacy, and Applied Cryptography Engineering* (2014), R. Chakraborty, V. Matyas, and P. Schaumont, Eds., vol. 8804 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 201–218.

[12] Bhasin, S., Danger, J.-L., Guilley, S., and Najm, Z. A low-entropy first-degree secure provable masking scheme for resource-constrained devices. In *Proceedings of the Workshop on Embedded Systems Security* (New York, NY, USA, 2013), WESS '13, ACM, pp. 7:1–7:10.

[13] Bhasin, S., Danger, J.-L., Guilley, S., and Najm, Z. Side-channel leakage and trace compression using normalized inter-class variance. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy* (New York, NY, USA, 2014), HASP '14, ACM, pp. 7:1–7:9.

[14] Bishop, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[15] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[16] Buhrow, B., Fritz, K., Gilbert, B., and Daniel, E. A highly parallel AES-GCM core for authenticated encryption of 400 Gb/s network protocols. In *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)* (Dec 2015), pp. 1–7.

[17] Cagli, E., Dumas, C., and Prouff, E. Convolutional neural networks with data augmentation against jitter-based countermeasures. pp. 45–68.

[18] Canright, D., and Batina, L. A Very Compact "Perfectly Masked" S-box for AES, booktitle = Proceedings of the 6th International Conference on Applied Cryptography and Network Security. ACNS'08, Springer-Verlag, pp. 446–459.

[19] Chari, S., Jutla, C., Rao, J. R., and Rohatgi, P. A cautionary note regarding evaluation of AES candidates on smart-cards. In *In Second Advanced Encryption Standard (AES) Candidate Conference*, pp. 133–147.

[20] Chari, S., Jutla, C. S., Rao, J. R., and Rohatgi, P. Towards sound approaches to counteract power-analysis attacks. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology* (1999), Springer-Verlag, pp. 398–412.

[21] Chari, S., Rao, J., and Rohatgi, P. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002* (2003), B. Kaliski, Koç, and C. Paar, Eds., vol. 2523 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 13–28.

[22] Chari, S., Rao, J. R., and Rohatgi, P. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers* (2002), pp. 13–28.

[23] Chou, J.-W., Chu, M.-H., Tsai, Y.-L., Jin, Y., Cheng, C.-M., and Lin, S.-D. An unsupervised learning model to perform side channel attack. In *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. Tseng, L. Cao, H. Motoda, and G. Xu, Eds., vol. 7818 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 414–425.

[24] Choudary, O., and Kuhn, M. G. Efficient template attacks. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers* (2013), pp. 253–270.

[25] Choudary, O., and Kuhn, M. G. Template attacks on different devices. In *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers* (2014), pp. 179–198.

[26] Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., and Verneuil, V. Horizontal correlation analysis on exponentiation. In *International Conference on Information and Communications Security* (2010), Springer, pp. 46–61.

[27] Coron, J.-S., and Goubin, L. On boolean and arithmetic masking against differential power analysis. In *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems* (London, UK, UK, 2000), CHES '00, Springer-Verlag, pp. 231–237.

[28] Cortes, C., and Vapnik, V. Support-vector networks. *Mach. Learn. 20*, 3 (Sept. 1995), 273–297.

[29] Dworkin, M. J. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. *Special Publication (NIST SP)-800-38D* (2007).

[30] Elaabid, M. A., and Guilley, S. Portability of templates. *J. Cryptographic Engineering 2*, 1 (2012), 63–74.

[31] Everitt, B., Landau, S., Leese, M., and Stahl, D. *Cluster Analysis*. Wiley series in probability and statistics. Wiley, 2011.

[32] Farashahi, R. R., Rashidi, B., and Sayedi, S. M. FPGA based fast and high-throughput 2-slow retiming 128-bit AES encryption algorithm. *Microelectronics Journal 45*, 8 (2014), 1014 – 1025.

[33] Farooq, U., and Aslam, M. F. Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA. *Journal of King Saud University - Computer and Information Sciences 29*, 3 (2017), 295 – 302.

[34] Fei, Y., Luo, Q., and Ding, A. A statistical model for DPA with novel algorithmic confusion analysis. In *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds., vol. 7428 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 233–250.

[35] Fouque, P.-A., Kunz-Jacques, S., Martinet, G., Muller, F., and Valette, F. Power Attack on Small RSA Public Exponent. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop* (2006), vol. 4249 of *Lecture Notes in Computer Science*, Springer, pp. 339–353.

[36] Gierlichs, B., Lemke-Rust, K., and Paar, C. Templates vs. stochastic methods. In *Cryptographic Hardware and Embedded Systems - CHES 2006* (2006), L. Goubin and M. Matsui, Eds., vol. 4249 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 15–29.

[37] Golic, J., and Tymen, C. Multiplicative masking and power analysis of AES. In *Cryptographic Hardware and Embedded Systems CHES 2002* (2003), B. Kaliski, c. Koc, and C. Paar, Eds., vol. 2523 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 198–212.

[38] Goubin, L. A sound method for switching between boolean and arithmetic masking. In *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems* (London, UK, UK, 2001), CHES '01, Springer-Verlag, pp. 3–15.

[39] Grosso, V., Standaert, F.-X., and Prouff, E. Low entropy masking schemes, revisited. In *Smart Card Research and Advanced Applications* (2014), A. Francillon and P. Rohatgi, Eds., vol. 8419 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 33–43.

[40] Guilleyho, S. DPA contest v4, 2013. Available at `http://www.dpacontest.org/v4/rsm_doc.php`.

[41] Guilleyho, S. DPA contest v4.2, 2013. Available at `http://www.dpacontest.org/v4/42_doc.php`.

[42] Hanley, N., Tunstall, M., and Marnane, W. Using templates to distinguish multiplications from squaring operations. *International Journal of Information Security 10*, 4 (2011), 255–266.

[43] Hastie, T., Tibshirani, R., and Friedman, J. Unsupervised learning. In *The Elements of Statistical Learning* (2009), Springer Series in Statistics, Springer New York, pp. 485–585.

[44] He, H., Jaffe, J., and Zou, L. Side channel cryptanalysis using machine learning.

[45] Henzen, L., and Fichtner, W. FPGA parallel-pipelined AES-GCM core for 100G ethernet applications. In *2010 Proceedings of ESSCIRC* (Sept 2010), pp. 202–205.

[46] Herbst, C., Oswald, E., and Mangard, S. An AES smart card implementation resistant to power analysis attacks. In *Applied Cryptography and Network Security, Second International Conference, ACNS 2006, volume 3989 of Lecture Notes in Computer Science* (2006), Springer, pp. 239–252.

[47] Hettwer, B., Gehrer, S., and Güneysu, T. Applications of machine learning techniques in side-channel attacks: a survey. *Journal of Cryptographic Engineering* (04 2019).

[48] Heuser, A., Picek, S., Jovic, A., and Legay, A. On the relevance of feature selection for profiled side-channel attacks.

[49] Heuser, A., and Zohner, M. Intelligent machine homicide - breaking cryptographic devices using support vector machines. In *COSADE* (2012), pp. 249–264.

[50] Heyszl, J., Ibing, A., Mangard, S., De Santis, F., and Sigl, G. Clustering algorithms for non-profiled single-execution attacks on exponentiations.

In *Smart Card Research and Advanced Applications* (2014), A. Francillon and P. Rohatgi, Eds., vol. 8419 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 79–93.

[51] Hofmann, M., and Klinkenberg, R. *RapidMiner: Data mining use cases and business analytics applications*. CRC Press, 2013.

[52] Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., and Vandewalle, J. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering 1*, 4 (2011), 293–302.

[53] Hospodar, G., Mulder, E., Gierlichs, B., Vandewalle, J., and Verbauwhede, I. Least squares support vector machines for side-channel analysis. In *COSADE 2011 - Second International Workshop on Constructive Side-Channel Analysis and Secure Design* (2011), pp. 293–302.

[54] Jap, D., and Breier, J. Overview of machine learning based side-channel analysis methods. In *Integrated Circuits (ISIC), 2014 14th International Symposium on* (Dec 2014), pp. 38–41.

[55] Joye, M., and Olivier, F. Side-channel analysis. In *Encyclopedia of Cryptography and Security, 2nd Ed.* 2011, pp. 1198–1204.

[56] Kent, S. IP Authentication Header. RFC 4302, Dec. 2005.

[57] Kent, S. IP Encapsulating Security Payload (ESP). RFC 4303, Dec. 2005.

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

[59] Korona, M., Skowron, K., Trzepiński, M., and Rawski, M. FPGA implementation of IPsec protocol suite for multigigabit networks. In *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)* (May 2017), pp. 1–5.

[60] Koteshwara, S., Das, A., and Parhi, K. K. Fpga implementation and comparison of AES-GCM and deoxys authenticated encryption schemes. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (May 2017), pp. 1–4.

[61] Kotsiantis, S. B. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial*

Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies (Amsterdam, The Netherlands, The Netherlands, 2007), IOS Press, pp. 3–24.

[62] Koziel, B., Azarderakhsh, R., and Jao, D. Side-channel attacks on quantum-resistant supersingular isogeny Diffie-Hellman. In *International Conference on Selected Areas in Cryptography* (2017), Springer, pp. 64–81.

[63] Kur, J., Smolka, T., and Svenda, P. Improving Resiliency of Java Card Code Against Power Analysis. In *Mikulaska kryptobesidka, Sbornik prispevku* (2009), pp. 29–39.

[64] Kutzner, S., and Poschmann, A. On the Security of RSM - Presenting 5 First- and Second-Order Attacks. In *Constructive Side-Channel Analysis and Secure Design* (2014), E. Prouff, Ed., vol. 8622 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 299–312.

[65] Lemsitzer, S., Wolkerstorfer, J., Felber, N., and Braendli, M. Multi-gigabit GCM-AES architecture optimized for FPGAs. In *Cryptographic Hardware and Embedded Systems - CHES 2007* (Berlin, Heidelberg, 2007), P. Paillier and I. Verbauwhede, Eds., Springer Berlin Heidelberg, pp. 227–238.

[66] Lerman, L., Bontempi, G., and Markowitch, O. Side channel attack: an approach based on machine learningn. In *COSADE 2011 - Second International Workshop on Constructive Side-Channel Analysis and Secure Design* (2011), pp. 29–41.

[67] Lerman, L., Bontempi, G., and Markowitch, O. The bias-variance decomposition in profiled attacks. *J. Cryptographic Engineering 5*, 4 (2015), 255–267.

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

[69] Lerman, L., Bontempi, G., Taieb, S. B., and Markowitch, O. A time series approach for profiling attack. In *Security, Privacy, and Applied Cryptography Engineering - Third International Conference, SPACE 2013, Kharagpur, India, October 19-23, 2013. Proceedings* (2013), pp. 75–94.

[70] Lerman, L., Martinasek, Z., and Markowitch, O. Robust profiled attacks: should the adversary trust the dataset? *IET Information Security 11*, 4 (2017), 188–194.

[71] Lerman, L., Medeiros, S. F., Bontempi, G., and Markowitch, O. A machine learning approach against a masked AES. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers* (2013), pp. 61–75.

[72] Lerman, L., Medeiros, S. F., Veshchikov, N., Meuter, C., Bontempi, G., and Markowitch, O. Semi-supervised template attack. In *Constructive Side-Channel Analysis and Secure Design - 4th International Workshop, COSADE 2013, Paris, France, March 6-8, 2013, Revised Selected Papers* (2013), pp. 184–199.

[73] Lerman, L., Poussier, R., Bontempi, G., Markowitch, O., and Standaert, F. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers* (2015), pp. 20–33.

[74] Lerman, L., Veshchikov, N., Markowitch, O., and Standaert, F.-X. Start simple and then refine: Bias-variance decomposition as a diagnosis tool for leakage profiling. *IEEE Transactions on Computers 67*, 2 (2017), 268–283.

[75] Liu, B., Ding, Z., Pan, Y., Li, J., and Feng, H. Side-channel attacks based on collaborative learning. In *International Conference of Pioneering Computer Scientists, Engineers and Educators* (2017), Springer, pp. 549–557.

[76] Maghrebi, H., Portigliatti, T., and Prouff, E. Breaking cryptographic implementations using deep learning techniques. pp. 3–26.

[77] Mahmoud, A., Rührmair, U., Majzoobi, M., and Koushanfar, F. Combined Modeling and Side Channel Attacks on Strong PUFs. *IACR Cryptol. ePrint Arch. 2013* (2013), 632.

[78] Mangard, S., Oswald, E., and Popp, T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[79] Martinasek, Z. *Kryptoanalyza postrannimi kanaly*. PhD thesis, Vysoke uceni technicke v Brne, fakulta elektrotechniky a komunikacnich technologii, 2013.

[80] Martinasek, Z., Clupek, V., and Trasy, K. General scheme of differential power analysis. In *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on* (July 2013), pp. 358–362.

[81] Martinasek, Z., Hajny, J., and Malina, L. Optimization of power analysis using neural network. In *Smart Card Research and Advanced Applications*, A. Francillon and P. Rohatgi, Eds., vol. 8419 of *Lecture Notes in Computer Science*. Springer International Publishing, 2013, pp. 94–107.

[82] Martinasek, Z., Hajny, J., Malina, L., and Matousek, D. Hardware-accelerated encryption with strong authentication. *Security and Protection of Informationl 1*, 9 (5 2017), 59–73.

[83] Martinasek, Z., Iglesias, F., Malina, L., and Martinasek, J. Crucial pitfall of DPA contest V4.2 implementation. *Secur. Commun. Networks 9*, 18 (2016), 6094–6110.

[84] Martinasek, Z., Macha, T., Raso, O., Martinasek, J., and Silhavy, P. Optimization of differential power analysis. *PRZEGLAD ELEKTROTECHNICZNY 87*, 12 (2011), 140 – 144.

[85] Martinasek, Z., Macha, T., and Stancikk, P. Power side channel information measurement. In *Research in telecommunication technologies RTT2010* (September 2010).

[86] Martinasek, Z., Macha, T., and Zeman, V. Classifier of power side channel. In *Proceedings of NIMT2010* (September 2010).

[87] Martinasek, Z., and Machu, P. New side channle in cryptography. In *Proceedings of the 17th Conference Student EEICT 2011* (April 2011).

[88] Martinasek, Z., Malina, L., and Trasy, K. *Profiling Power Analysis Attack Based on Multi-layer Perceptron Network*. Springer International Publishing, Cham, 2015, pp. 317–339.

[89] Martinásek, Z., Nečas, O., Zeman, V., and Martinásek, J. Diferenciální elektromagnetická analýza. *Elektrorevue - Internetový časopis (http://www.elektrorevue.cz 2011*, 60 (2011), 1 – 6.

[90] Martinasek, Z., Petrik, T., and Stancik, P. Conditions affecting the measurement of power analysis. In *Research in telecommunication technologies RTT2011* (September 2011).

[91] Martinásek, Z., Petřík, T., and Stančík, P. Parametry ovlivňující proudovou analýzu mikroprocesoru vykonavajícího funkci addroundkey. *Elektrorevue - Internetovy časopis (http://www.elektrorevue.cz 2011*, 51 (2011), 1 – 6.

[92] Martinasek, Z., and Zeman, V. Innovative method of the power analysis. *Radioengineering 22*, 2 (2013).

[93] Martinasek, Z., Zeman, V., Malina, L., and Martinasek, J. k-Nearest neighbors algorithm in profiling power analysis attacks. *Radioengineering 25*, 2 (2016), 365–382.

[94] Martinasek, Z., Zeman, V., Sysel, P., and Trasy, K. Near electromagnetic field measurement of microprocessor. *PRZEGLAD ELEKTROTECHNICZNY 89*, 2a (2013), 203 – 207.

[95] Martinasek, Z., Zeman, V., and Trasy, K. Simple electromagnetic analysis in cryptography. *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems 1*, 1 (2012), 1 – 6.

[96] Martinez, A. M., Mart'inez, A. M., and Kak, A. C. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23* (2001), 228–233.

[97] Martinásek, Z., Člupek, V., Zeman, V., and Sysel, P. Základní metody diferenciální proudové analýzy. 1–10.

[98] McGrew, D., and Viega, J. The Galois/counter mode of operation (GCM). *submission to NIST Modes of Operation Process 20* (2004).

[99] Mesquita, D., Techer, J.-D., Torres, L., Sassatelli, G., Cambon, G., Robert, M., and Moraes, F. Current mask generation: a transistor level security against dpa attacks. In *SBCCI* (2005), pp. 115–120.

[100] Messerges, T. Using Second-Order Power Analysis to Attack DPA Resistant Software. In *Cryptographic Hardware and Embedded Systems - CHES 2000* (2000), Koç and C. Paar, Eds., vol. 1965 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 238–251.

[101] Mladenic, D., Brank, J., Grobelnik, M., and Milic-Frayling, N. Feature selection using support vector machines. In *The 27th Annual International ACM SIGIR Conference (SIGIR 2004)* (2004), pp. 234–241.

[102] Moradi, A., Guilley, S., and Heuser, A. Detecting hidden leakages. In *Applied Cryptography and Network Security* (2014), I. Boureanu, P. Owesarski, and S. Vaudenay, Eds., vol. 8479 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 324–342.

[103] Mukhtar, N., Mehrabi, M. A., Kong, Y., and Anjum, A. Machine-Learning-Based Side-Channel Evaluation of Elliptic-Curve Cryptographic FPGA Processor. *Applied Sciences 9*, 1 (2019), 64.

[104] Muresan, R., Vahedi, H., Zhanrong, Y., and Gregori, S. Power-smart system-on-chip architecture for embedded cryptosystems. In *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis* (New York, NY, USA, 2005), CODES+ISSS '05, ACM, pp. 184–189.

[105] Nabney, I. T. *NETLAB: algorithms for pattern recognition*. Advances in Pattern Recognition. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[106] Nassar, M., Souissi, Y., Guilley, S., and Danger, J.-L. RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012* (March 2012), pp. 1173–1178.

[107] Nemec, M., Sys, M., Svenda, P., Klinec, D., and Matyas, V. The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 1631–1648.

[108] Nguyen, T. T., Nguyen, V. C., Huynh, T. V., Luong, Q. Y. H., and Dang, T. H. Performance enhancement of encryption and authentication IP cores for IPSec based on multiple-core architecture and dynamic partial reconfiguration on FPGA. In *2018 2nd International Conference on Recent Advances in Signal Processing, Telecommunications Computing (SigTelCom)* (Jan 2018), pp. 126–131.

[109] Oswald, E. Enhancing simple power-analysis attacks on elliptic curve cryptosystems. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2002), Springer, pp. 82–97.

[110] Oswald, E., Mangard, S., Herbst, C., and Tillich, S. Practical second-order dpa attacks for masked smart card implementations of block ciphers. In *Topics in Cryptology CT RSA 2006*, D. Pointcheval, Ed., vol. 3860 of

Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 192–207.

[111] Oswald, E., Mangard, S., and Pramstaller, N. Secure and Efficient Masking of AES - A Mission Impossible?, 2004. Elisabeth.Oswald@iaik.tugraz.at 12573 received 4 Jun 2004.

[112] Oswald, E., Mangard, S., Pramstaller, N., and Rijmen, V. A Side-Channel Analysis Resistant Description of the AES S-Box. In *Fast Software Encryption* (2005), H. Gilbert and H. Handschuh, Eds., vol. 3557 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 413–423.

[113] Oswald, M. E., Mangard, S., Herbst, C., and Tillich, S. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In *Topics in Cryptology - CT-RSA 2006* (2006), D. Pointcheval, Ed., vol. 3860 of *Lecture Notes in Computer Science*, Springer, pp. 192 – 207.

[114] Pagh, R., and Rodler, F. F. Cuckoo hashing. *Journal of Algorithms 51*, 2 (2004), 122 – 144.

[115] Park, A., Shim, K.-A., Koo, N., and Han, D.-G. Side-channel attacks on post-quantum signature schemes based on multivariate quadratic equations. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 500–523.

[116] Perin, G., Imbert, L., Torres, L., and Maurine, P. Attacking randomized exponentiations using unsupervised learning. In *Constructive Side-Channel Analysis and Secure Design* (2014), E. Prouff, Ed., vol. 8622 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 144–160.

[117] Picek, S., Heuser, A., and Guilley, S. Template attack versus bayes classifier. *Journal of Cryptographic Engineering 7*, 4 (2017), 343–351.

[118] Picek, S., Heuser, A., Jovic, A., and Legay, A. Climbing down the hierarchy: hierarchical classification for machine learning side-channel attacks. In *International Conference on Cryptology in Africa* (2017), Springer, pp. 61–78.

[119] Prouff, E., and Rivain, M. A Generic Method for Secure SBOX Implementation. In *Information Security Applications* (2007), S. Kim, M. Yung, and H.-W. Lee, Eds., vol. 4867 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 227–244.

[120] Pus, V., Kekely, L., and Korenek, J. Design methodology of configurable high performance packet parser for FPGA. In *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems* (April 2014), pp. 189–194.

[121] Qu, S., Shou, G., Hu, Y., Guo, Z., and Qian, Z. High throughput, pipelined implementation of AES on FPGA. In *2009 International Symposium on Information Engineering and Electronic Commerce* (May 2009), pp. 542–545.

[122] Quisquater, J.-J., and Samyde, D. Automatic code recognition for smart cards using a Kohonen neural network. In *Proceedings of the 5th conference on Smart Card Research and Advanced Application Conference - Volume 5* (Berkeley, CA, USA, 2002), CARDIS'02, pp. 6–6.

[123] Read, J., Pfahringer, B., Holmes, G., and Frank, E. Classifier chains for multi-label classification. *Machine Learning 85*, 3 (2011), 333–359.

[124] Rechberger, C., and Oswald, E. Practical template attacks. In *Information Security Applications* (2005), C. Lim and M. Yung, Eds., vol. 3325 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 440–456.

[125] Renauld, M., Standaert, F., Veyrat-Charvillon, N., Kamel, D., and Flandre, D. A formal study of power variability issues and side-channel attacks for nanoscale devices. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings* (2011), pp. 109–128.

[126] Saeedi, E., Hossain, M. S., and Kong, Y. Side-channel information characterisation based on cascade-forward back-propagation neural network. *Journal of Electronic Testing 32*, 3 (2016), 345–356.

[127] Saeedi, E., Kong, Y., and Hossain, M. S. Side-channel attacks and learning-vector quantization. *Frontiers of Information Technology & Electronic Engineering 18*, 4 (2017), 511–518.

[128] Schindler, W. On the optimization of side-channel attacks by advanced stochastic methods. In *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings* (2005), vol. 3386 of *Lecture Notes in Computer Science*, Springer, pp. 85–103.

[129] Schindler, W., Lemke, K., and Paar, C. A stochastic model for differential side channel cryptanalysis. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings* (2005), pp. 30–46.

[130] Sheffer, Y., and Fluhrer, S. Additional Diffie-Hellman Tests for the Internet Key Exchange Protocol Version 2 (IKEv2). RFC 6989, July 2013.

[131] Smekal, D., Frolka, J., and Hajny, J. Acceleration of AES encryption algorithm using field programmable gate arrays. *IFAC-PapersOnLine 49*, 25 (2016), 384 – 389. 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

[132] Sokolova, M., and Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information Processing and Management 45*, 4 (2009), 427–437.

[133] Soliman, M. I., and Abozaid, G. Y. FPGA implementation and performance evaluation of a high throughput crypto coprocessor. *Journal of Parallel and Distributed Computing 71*, 8 (2011), 1075–1084.

[134] Standaert, F.-X., Malkin, T., and Yung, M. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT* (2009), pp. 443–461.

[135] Standaert, F.-X., Malkin, T. G., and Yung, M. A unified framework for the analysis of side-channel key recovery attacks. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques* (Berlin, Heidelberg, 2009), EUROCRYPT '09, Springer-Verlag, pp. 443–461.

[136] Technologies, N. Netcope FPGA boards. Available at `https://www.netcope.com/en/products/fpga-boards`.

[137] Timon, B. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2019*, 2 (Feb. 2019), 107–131.

[138] Tsoumakas, G., and Katakis, I. Multi-label classification: An overview. *Int J Data Warehousing and Mining 2007* (2007), 1–13.

[139] Vliegen, J., Reparaz, O., and Mentens, N. Maximizing the throughput of threshold-protected AES-GCM implementations on FPGA. In *2017 IEEE*

2nd International Verification and Security Workshop (IVSW) (July 2017), pp. 140–145.

[140] Whitnall, C., and Oswald, E. Robust Profiling for DPA-Style Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings* (2015), pp. 3–21.

[141] Wolpert, D., and Macready, W. G. No free lunch theorems for optimization. *IEEE Trans. Evolutionary Computation 1*, 1 (1997), 67–82.

[142] Xilinx. Virtex ultrascale+, https://www.xilinx.com/products/silicon-devices/fpga/virtex-ultrascale-plus.html, (online).

[143] Yang, S., Zhou, Y., Liu, J., and Chen, D. Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In *Information Security and Cryptology - ICISC 2011* (2012), H. Kim, Ed., vol. 7259 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 169–185.

[144] Ye, X., and Eisenbarth, T. On the vulnerability of low entropy masking schemes. In *Smart Card Research and Advanced Applications* (2014), A. Francillon and P. Rohatgi, Eds., vol. 8419 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 44–60.

[145] Zhang, Z., Wu, L., Wang, A., and Mu, Z. Improved leakage model based on genetic algorithm. *IACR Cryptology ePrint Archive 2014* (2014), 314.

[146] Zhao, J., Zeng, X., Han, J., and Chen, J. Simplified AES Algorithm Resistant to Zero-Value power analysis and its VLSI Implementation. In *Solid-State and Integrated Circuit Technology, 2006. ICSICT'06. 8th International Conference on* (2006), IEEE, pp. 1937–1940.

Author's selected publications since 2014

[147] Lerman, L., Martinasek, Z., and Markowitch, O. Robust profiled attacks: should the adversary trust the dataset? *IET Information Security 11*, 4 (2017), 188–194.

[148] Malina, L., Clupek, V., Martinasek, Z., Hajny, J., Oguchi, K., and Zeman, V. Evaluation of Software-Oriented Block Ciphers on Smartphones. In *Revised Selected Papers of the 6th International Symposium on Foundations and Practice of Security - Volume 8352* (Berlin, Heidelberg, 2013), FPS 2013, Springer-Verlag, p. 353–368.

[149] Malina, L., Popelova, L., Dzurenda, P., Hajny, J., and Martinasek, Z. On feasibility of post-quantum cryptography on small devices. *IFAC-PapersOnLine 51*, 6 (2018), 462 – 467. 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018.

[150] Martinasek, Z., Clupek, V., and Trasy, K. Acoustic attack on keyboard using spectrogram and neural network. In *2015 38th International Conference on Telecommunications and Signal Processing (TSP)* (2015), pp. 637–641.

[151] Martinasek, Z., Dzurenda, P., and Malina, L. Profiling power analysis attack based on MLP in DPA contest V4.2. In *39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27-29, 2016* (2016), IEEE, pp. 223–226.

[152] Martinasek, Z., Hajny, J., and Malina, L. Optimization of power analysis using neural network. In *Smart Card Research and Advanced Applications*, A. Francillon and P. Rohatgi, Eds., vol. 8419 of *Lecture Notes in Computer Science*. Springer International Publishing, 2013, pp. 94–107.

[153] Martinasek, Z., Hajny, J., Malina, L., and Matousek, D. Hardware-accelerated encryption with strong authentication. In *Security and Protection of Information* (june 2017), no. 1, pp. 1–10.

[154] Martinasek, Z., Hajny, J., Smekal, D., Malina, L., Matousek, D., Kekely, M., and Mentens, N. 200 Gbps Hardware Accelerated Encryption System for FPGA Network Cards. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security* (New York, NY, USA, 2018), ASHES '18, Association for Computing Machinery, p. 11–17.

[155] Martinasek, Z., Iglesias, F., Malina, L., and Martinasek, J. Crucial pitfall of DPA contest V4.2 implementation. *Secur. Commun. Networks 9*, 18 (2016), 6094–6110.

[156] Martinasek, Z., and Malina, L. Profiling power analysis attack based on multi-layer perceptron network. In *MMCTSE* (2014). in print.

[157] Martinasek, Z., Zapletal, O., Vrba, K., and Trasy, K. Power analysis attack based on the MLP in DPA contest v4. In *38th International Conference on Telecommunications and Signal Processing, TSP 2015, Prague, Czech Republic, July 9-11, 2015* (2015), IEEE, pp. 154–158.

[158] Martinasek, Z., Zeman, V., Malina, L., and Martinasek, J. k-Nearest neighbors algorithm in profiling power analysis attacks. *Radioengineering 25*, 2 (2016), 365–382.

[159] Smekal, D., Hajny, J., and Martinasek, Z. Comparative analysis of different implementations of encryption algorithms on FPGA network cards. In *15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018* (IFAC-PapersOnLine, may 2018), no. 6, IFAC-PapersOnLine, pp. 312–317.

[160] Smekal, D., Hajny, J., and Martinasek, Z. Hardware-accelerated Twofish core for FPGA. In *2018 41st International Conference on Telecommunications and Signal Processing (TSP)* (2018), pp. 1–5.

[161] Smekal, D., Hajny, J., Martinasek, Z., Malina, L., Vrba, K., and Matousek, D. Vysokorychlostní šifrování se silnou autentizací na platformě FPGA. In *Sborník MKB 2017* (december 2017), pp. 45–53.

[162] Zeman, V., and Martinasek, Z. Kryptografie v informatice. University textbook, UTKO, FEKT, VUT v Brně, UTKO, FEKT, VUT v Brně, january 2015.

[163] Zeman, V., and Martinasek, Z. Ochrana informací šifrováním pro integrovanou výuku vut a VŠB-TUO. University textbook, UTKO, FEKT, VUT v Brně, UTKO, FEKT, VUT v Brně, january 2015.

List of abbreviations

AES	Advances Encryption Standard
AH	Authentication Headers
APDU	Application Protocol Data Unit
API	Application Programming Interface
BRAM	Block Random Access Memory
CTA	Classical Template Attack
CMOS	Complementary Metal–Oxide–Semiconductor
CPA	Diferential Power Analysis based on Correlation coefficient
CV	Cross-Validation
DC	Direct Current
DH	Diffie-Hellman
DPA	Differential Power Analysis
DS	Data Set
DT	Decision Trees
EC	Elliptic-Curve
ESP	Encapsulating Security Payload
ETA	Efficient Template Attack
FF	Flip-Flop
FIFO	First IN First OUT
FN	False Negative
FP	False Positive
FPGA	Field-Programmable Gate Array
GCM	Galois Counter Mode
GE	Guessing Entropy
GF	Galois Field
GSR	Global Success Rate
HD	Hamming Distance
HFE	Header Field Extractor
HW	Hamming Weight
IKE	Internet Key Exchange
IP	Interesting Points
IPsec	Internet Protocol Security
LDA	Linear Discriminant Analysis
LTU	Look-Up-Tables
MIA	Mutual Information Analysis
MK	Master Key
MKA	Master Key Authentication

MKE	Master Key Encryption
ML	Machine Learning
MLP	MultiLayer Perceptrons
NDK	Netcope Development Kit
NICV	Normalized Inter-Class Variance
NIST	National Institute of Standards and Technology
OS	Operation System
PA	Power Analysis
PCA	Principal Components Analysis
PCIe	Peripheral Component Interconnect Express
PGE	Partial Guessing Entropy
PKI	Public Key Infrastructure
PSK	Pre-Shared Keys
PSR	Partial Success Rate
RF	Random Forests
ROC	Receiver Operator Characteristic
RSA	Rivest Shamir Adleman
RSM	Rotating Sbox Masking
SA	Stochastic Approach
SeA	Security Association
SAD	Security Association Database
SAM	Security Access Module
SCA	Side-Channel Analysis
SNR	Signal-to-Noise Ratio
SODPA	Second-Order Differential Power Analysis
SOSD	Sum Of Squared pairwise Differences
SPA	Simple Power Analysis
SPD	Security Policy Database
SRAM	Static Random Access Memory
SVM	Support Vector Machines
TA	Template-based Attack
TN	True Negative
TP	True Positive
USB	Universal Serial Bus
VHDL	VHSIC Hardware Description Language
VI	Virtual Interface
XOR	Exclusive-OR
ZV	Zero-vale model