Instituto Tecnológico y de Estudios Superiores de Occidente

Department of Mathematics and Physics
Master of Data Science

On the Security of Embedded Systems Against Side-channel Attacks

THESIS to obtain the **DEGREE** of
MASTER IN DATA SCIENCE

A thesis presented by: **Alberto Arjona Cabrera**

Thesis Advisor: **Dr. Fernando I. Becerra López**

Tlaquepaque, Jalisco, November, 2020

Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Department of Mathematics and Physics
Master of Data Science Approval Form

Thesis Title:
On the Security of Embedded Systems Against Side-channel Attacks

Author:
Alberto Arjona Cabrera

Thesis Approved to complete all degree requirements for the Master of Science Degree in Data Science.

Thesis Advisor, **Dr. Fernando I. Becerra López**

Thesis Reader, **Dr. Riemann Ruiz Cruz**

Thesis Reader, **Dr. Juan Diego Sánchez Torres**

Academic Advisor, **M. Juan Carlos Martínez Alvarado**

Tlaquepaque, Jalisco, November, 2020

On the Security of Embedded Systems Against Side-channel Attacks

Alberto Arjona Cabrera

Abstract

Side-Channel Analysis (SCA) represents a serious threat to the security of millions of smart devices that form part of the so-called Internet of Things (IoT). On the other hand, perform the "right- fitting" cryptographic code for the IoT is a highly challenging task due to the reduced resource constraints of must of the IoT devices and the variety of cryptographic algorithms on disposal. An important criterion to assess the suitability of a light-weight cipher implementation, with respect to the SCA point of view, is the amount of energy leakage available to an adversary.

In this thesis, the efficiency of a selected function that is commonly used in AES implementations in the perspective of Correlation Power Analysis (CPA) attacks are analyzed, leading to focus on the very common situation where the exact time of the sensitive processing is drowned in a large number of leakage points.

In the particular case of statistical attacks, much of the existing literature essentially develop the theory under the assumption that the exact sensitive time is known and cannot be directly applied when the latter assumption is relaxed, being such a particular aspect for the simple Differential Power Analysis (DPA) in contrast with the CPA.

To deal with this issue, an improvement that makes the statistical attack a real alternative compared with the simple DPA has been proposed. For the power consumption model (Hamming Weight model), and by rewriting the simple DPA attacks in terms of correlation coefficients between Boolean functions. Exhibiting properties of S-boxes relied on CPA attacks and showing that these properties are opposite to the non-linearity criterion and to the propagation criterion assumed for the former DPA.

In order to achieve this goal, the study has been illustrated by various attack experiments performed on several copies implementations of the light-weight AES chipper in a well-known micro-controller educative platform within an 8-bit processor architecture deployed on a 350 nanometers CMOS technology.

The Side-channel attacks presented in this work have been set in ideal conditions to capture the full complexity of an attack performed in real-world conditions, showing that certain implementation aspects can influence the leakage levels. On the other side, practical improvements are proposed for specific contexts by exploring the relationship between the non-linearity of the studied selection function and the measured leakages, with the only pretension to bridge the gap between the theory and the practice. The results point to new enlightenment on the resilience of basic operations executed by common light-weight ciphers implementations against CPA attacks.

Contents

List of Figures

List of Tables

Mad Science. High Voltage Vacuum Plasma. Dedicated to Purpose Beyond Reason. Liquid Laser Prototyping. Inventions Wrapped in Art. Yttrium Argon Laser Surface Etching Point. Cloud Mapping, Aimed Directly to The Head, Pulsating Power In Between the Eyes, You Decide, With No Remorse, and Absolutely No Adult Supervision Allowed...

9091, 9901, 333667, 909091, 99990001, 999999000001, 9999999900000001, 909090909090909091, 1111111111111111111, 11111111111111111111111, 900900900900990990990991, ...

1 Introduction

Contents

`\Side-channel attacks` use observations made during the execution of an implementation of a `\cryptographic algorithm` to recover secret information. From the multitude of side-channel attacks, `\correlation power analysis` (CPA) stands out as a very efficient and reliable technique. Its success is augmented by the minimally invasive methods employed for the acquisition of the side-channel information. Some of the most frequently used sources of side-channel `\leakage` are the `\power consumption` or the `\electromagnetic emissions` (EM) of a device under attack.

Nowadays, the `\Advanced Encryption Standard`(AES)[1] is the most popular `\symmetric cryptographic algorithm` in use. It is widely deployed to `\secure data` in transit or at rest. Various `\network protocols` rely on AES in different modes of operation to provide security services such as confidentiality and authenticity. The usage spectrum of the AES stretches from powerful servers and personal computers to resource constrained devices such as wireless sensor nodes. [2]

The assumption usually done about the attacker has full control of the `\AES` input is not the case in a real-world communication protocol[3] [4], when often a major part of the input is fixed, and only a few bytes are variable. Moreover, sometimes the `\attacker` cannot control these variable bytes and it has to passively observe executions of the `\targeted algorithm` without being able to trigger encryptions of her own free will. Part of the present analysis relay on how much control of the AES input does an attacker need to recover the `\secret key` of the cipher by performing a side-channel attack against a communication[5] protocol.

In the present work, the focus is made on `\CPA attacks` thanks to their efficiency and reliability. By opting for a non-invasive

[1] NIST. *Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication (FIPS), 197 edition, 2001

[2] C. O'Flynn and Z. Chen. *Power analysis attacks against IEEE 802.15.4 nodes. Lecture Notes in Computer Science*, volume 9689. Springer, constructive side-channel analysis and secure design - 7th international workshop edition, April 2016

[3] E. Prouff. *DPA Attacks and S-boxes. InFast Software Encryption*. Springer, 4th edition, 2005

[4] R. Verdult L. Batina J. Balasch, B. Gierlichs and I. Verbauwhede. *Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. Lecture Notes in Computer Science*, volume 7178. Springer, in o. dunkelman, editor, ct-rsa edition, 2012

[5] J. G. Proakis and D. K. Manolakis. *Digital Signal Processing*. Prentice-Hall, 4th edition, May 2006

measurement setup and hence selecting the \EM emissions of the target \processor as a source of side-channel leakage.

1.1 *Power Analysis and Algorithmic Security*

In \computer security, a side-channel attack is any attack based on information gained from the implementation of a computer system, rather than weaknesses in the implemented algorithm itself[6]. \Timing information, \power consumption, \electromagnetic leaks or even sound can provide an extra source of information, which can be exploited[7] [8].

[6] for example, \crypto-analysis and software bugs

[7] E. Prouff. *DPA Attacks and S-boxes. InFast Software Encryption.* Springer, 4th edition, 2005

[8] C. O'Flynn and Z. Chen. *Power analysis attacks against IEEE 802.15.4 nodes. Lecture Notes in Computer Science*, volume 9689. Springer, constructive side-channel analysis and secure design - 7th international workshop edition, April 2016

1.2 *Why do those attacks exist?*

Computer security is a deep layered domain. \Predicting and \modeling those attacks is very difficult. Each \layer of security impacts the assumptions made by others. The \software developer assumes that the hardware designer did his job well. As a result, \security faults often involve unanticipated interactions between components. Components which are made by different people.

Power analysis provides a way to "take a look inside" what it is supposed to be \tamper-proof hardware. For example, \Advanced Encryption Standard (AES) key schedule involves rotating 128-bit key \registers [9]. The \device where the implementation runs, typically shifts the register in order to handle \high order multiplications, but other \processor demanding operations are executed too. The \power analysis can distinguish between these events, enabling an adversary to determine the bits of the \secret key.

[9] NIST. *Advanced Encryption Standard (AES).* Federal Information Processing Standards Publication (FIPS), 197 edition, 2001

Implementations of algorithms such as AES and \Triple Data Encryption Standard (DES) that are believed to be mathematically [10] may be trivially breakable using power analysis attacks. As a result, power analysis attacks combine elements of algorithmic \crypto-analysis and implementation security in \microelectronic devices.

[10] J. Daemen and V. Rijmen. *The design of Rijndael: AES the Advanced Encryption Standard.* Springer-verlag edition, 2002

The equipment necessary for performing \power analysis attacks is widely available. For example, most digital storage \oscilloscopes provide the necessary \data collection functionality, and the \data analysis is typically performed using conventional PCs. Products designed for \lab testing purposes, equipped with high-resolution and high-speed \digital analog converters (DAC´s), are also available in the commercial market.

Specifically, the \correlation power analysis (CPA) is an \attack that allows to find a \secret encryption key that is stored on a \victim device [11].

[11] R. Verdult L. Batina J. Balasch, B. Gierlichs and I. Verbauwhede. *Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. Lecture Notes in Computer Science*, volume 7178. Springer, in o. dunkelman, editor, ct-rsa edition, 2012

There are four steps to a CPA attack:

1. Write down a model for the victim's `\power consumption`. This model will look at one specific point in the `\encryption algorithm`. That is, after step 2 of the `\encryption process` [12], the intermediate result is x, so the power consumption is $f(x)$.
2. Get the victim to encrypt several different plain-texts. Record a trace of the victim's `\power consumption` during each of these encryptions.
3. Attack small parts (subkeys)[13] of the secret key[14]:
   - Consider every possible option for the `\subkey`. For each guess and each `\trace`, use the known `\plain-text` and the guessed subkey to calculate the power consumption according to our model.
   - Calculate the `\Pearson correlation coefficient` between the modeled and actual power consumption. Do this for every data point in the traces.
   - Decide which subkey guess correlates best to the `\measured traces`.
4. Put together the best subkey guesses to obtain the full `\secret key`.

[12] For a better vision, take a look in the figure 5.2

[13] R. Verdult L. Batina J. Balasch, B. Gierlichs and I. Verbauwhede. *Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. Lecture Notes in Computer Science*, volume 7178. Springer, in o. dunkelman, editor, ct-rsa edition, 2012

[14] For a better vision, take a look in the figure 5.1

2 General Objective

Different side-channel estimators may have different efficiencies. However, their fair comparison is a difficult task since many factors come into play. Particularly, their intrinsic statistical properties and the quality of their estimation are significant factors, by performing several tries concentrated on the evaluation of the efficiency of certain attacks. First attempts were focused on finding a link between the Signal-to-noise ratio (SNR) of the power measurements and the effectiveness of the attack, by presenting a statistical model for CPA and finding in this way an approximation of the success rate.

While these preliminaries are only focused on the correct key guess[1], a new approach is proposed, determining the exact success rate of CPA in assuming a uniform setting in terms of the leakage model, particularly providing an estimation of the success rate depending on the relationship between the correct and incorrect key hypothesis[2], the number of measurements and the SNR. Providing also, a methodology to evaluate side-channel estimators giving the example of the application with DPA. The approach consists of estimating the success rate of DPA due to the characterization of the physical implementation as well as the cryptographic algorithm. Finally, come up with the generalized estimation results of the former idea, which has been restricted to the application of one-bit DPA, to any additive estimators and show a feasible application to CPA.

In addition to the above, it is formulated a framework that can be carried out to classify various estimators, which could, perhaps, close the gap between purely practical and purely theoretical evaluations and to gain awareness in the current hardware weakness to serve in the sake of cybersecurity improvement.

[1] Finding the secret key, being such the principal goal of the performed attack, by exploiting the public knowledge of the encryption algorithms.

[2] Confusion Rate

3 Theoretical Framework

Contents

When \statistical power analysis is used against \cryptographic devices, the principal trends considered are \differential power analysis (DPA), \linear regression \power analysis (LRPA) and \correlation power analysis (CPA).

A classical model is used for the \power consumption of \cryptographic devices [1] [2]. It is based on the \Hamming distance of the data handled with regard to an unknown but constant reference state. Once validated experimentally, it allows an \optimal attack to be derived that is called correlation power analysis(CPA). CPA has been proposed to use the \correlation factor between the \power samples and the \Hamming weight of the handled data. In this section are also explained the defects of former approaches such as differential power analysis.

[1] R. Verdult L. Batina J. Balasch, B. Gierlichs and I. Verbauwhede. *Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. Lecture Notes in Computer Science*, volume 7178. Springer, in o. dunkelman, editor, ct-rsa edition, 2012

[2] C. O'Flynn and Z. Chen. *Power analysis attacks against IEEE 802.15.4 nodes. Lecture Notes in Computer Science*, volume 9689. Springer, constructive side-channel analysis and secure design - 7th international workshop edition, April 2016

3.1 Notations

- \Random variables are denoted by large letters X, Y, Z.
- A realization of a random variable, said X, is denoted by the corresponding lowercase letter, said x.
- A sample of several \observations of X is denoted by $(x_i)_i$. It will sometimes be viewed as a vector defined over the definition set of X.
- The notation $(x_i)_i \leftarrow X$ denotes the instantiating of the set of observations $(x_i)_i$ from X.

- The `\mean` of X is denoted $\mathbb{E}[X]$, its `\standard deviation` by $\sigma[X]$ and its `\variance` by $\mathrm{var}[X]$. The latter equals $\mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$.
- The `\co-variance` of two random variables X and Y is denoted by $\mathrm{cov}(X, Y)$ and satisfies $\mathrm{cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$.
- When needed to specify the variable on which `\statistics` are computed, the variable will be wrote in subscript (for example $\mathbb{E}_X[Y]$ instead of $\mathbb{E}[Y]$).
- The notation $\vec{X}$ will be used to denote column vectors and $\vec{X}[u]$ will denote its u^{th} coordinate.
- Calligraphic letters will be used to denote a matrix.
- The elements of a matrix $\mathcal{M}$ will be denoted by $\mathcal{M}[i][j]$.
- Classical additions and multiplications (over real values, vectors or matrices) are denoted by $+$ and $\times$ respectively.
- Scalar-vector operations are denoted by $\cdot$ and $/$ (all the coordinates of the vector are multiplied, and respectively divided, by the scalar).
- When applied to vectors or matrices, the symbols $(.)^2$ and $\sqrt{.}$ denote the operation consisting in computing the square and the square root of all the vector/matrix coordinates, respectively.
- A function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ will be called a (n, m)-function[3].

[3] $\mathbb{F}_2$ represents the Galois field of two elements.

3.2 Modeling Power Consumption

\Electronic computers (micro-controllers, FPGA's, GPU's, DSP's, among others) have two components to their power consumption. First, static power consumption which is the power required to keep the device running [4]. This \static power depends on things like the number of transistors inside the device. Secondly, and more importantly, dynamic power consumption, it depends on the data moving around inside the device. Every time a bit is changed from a 0 to a 1 (or vice versa), some current is required to charge and discharge the data lines. The \dynamic power is the part to get interested in, because it can tell what's happening inside.

[4] R. Verdult L. Batina J. Balasch, B. Gierlichs and I. Verbauwhede. *Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. Lecture Notes in Computer Science*, volume 7178. Springer, in o. dunkelman, editor, ct-rsa edition, 2012

One of the simplest models for power consumption is the \Hamming distance [5]. The Hamming Distance between two binary numbers is the number of different bits in the numbers.

[5] C. O'Flynn and Z. Chen. *Power analysis attacks against IEEE 802.15.4 nodes. Lecture Notes in Computer Science*, volume 9689. Springer, constructive side-channel analysis and secure design - 7th international workshop edition, April 2016

Then Hamming Distance model in CPA attacks could be used. Finding a point in the \encryption algorithm where the victim changes a variable from x to y, then it could be estimated that the \power consumption is proportional to \Hamming distance.

3.3 The Hamming Distance Consumption Model

In a m-bit microprocessor, binary data is coded$D = \sum_{j=0}^{m-1} d_j 2^j$, with the bit values $d_j = 0$ or 1. Its \Hamming weight is simply the number of bits set to 1, $H(D) = \sum_{j=0}^{m-1} d_j$. Its integer values stand between 0 and m. If D contains m independent and uniformly distributed \bits, the whole word has an average Hamming weight $\mu_H = m/2$ and a variance $\sigma_H^2 = m/4$.

It is generally assumed that the data \leakage through the power side-channel depends on the number of bits switching from one state to the other at a given time. A \microprocessor is modeled as a \state-machine where transitions from state to state are triggered by events such as the edges of a \clock signal. This seems relevant when looking at a logical \elementary gate as implemented in \CMOS technology. The current consumed is related to the \energy required to flip the bits from one state to the next. It is composed of two main contributions: the capacitor's charge and the short circuit induced by the gate transition. [6]

[6] Curiously, this elementary behavior is commonly admitted but has never given rise to any satisfactory model that is widely applicable. Only hardware designers are familiar with simulation tools to \forensic the current consumption of microelectronic devices.

By adopting the \transition model, basic questions arises; what is the reference state from which the bits suppose to switch? Does this reference is variable depending on different \hardware architectures? Assuming here that this reference state is a \constant machine word, R, which is unknown, but not necessarily zero. It will always be the same if the same \data manipulation always occurs at the same time, although this assumes the absence of any `\desynchronizing` effect. Moreover, it is assumed that switching a bit from 0 to 1 or from 1 to 0 requires the same amount of energy and that all the machine bits handled at a given time are perfectly balanced and consume the same.

These restrictive assumptions are quite realistic and affordable without any thorough knowledge of `\microelectronic devices`. They lead to a convenient expression for the `\leakage model`. Indeed the number of `\flipping bits` to go from R to D is described by $H(D \oplus R)$ also called the `\Hamming distance` between D and R. This statement encloses the `\Hamming weight` model which assumes that $R = 0$. If D is a uniform random variable, so is $D \oplus R$, and $H(D \oplus R)$ has the same mean $m/2$ and variance $m/4$ as $H(D)$

Assuming that we have a `\linear relationship` between the `\current consumption` and $H(D \oplus R)$, which could be interpreted as a tacit limitation but considering a chip as a large set of elementary `\electrical components`, this linear model fits reality quite well. It does not represent the entire consumption of a `\chip` but only the data dependent part. This does not seem unrealistic because the bus lines are usually considered as the most consuming elements within a `\micro-controller`. All the remaining things in the `\power consumption` of a chip are assigned to a term denoted b which is assumed independent [7] from the other variables: b encloses offsets, time dependent components and noise. Therefore the basic model for the data dependency can be written as equation 3.1 shows:

[7] Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 1st edition, April 2020. ISBN 978-1108455145

$$W = aH(D \oplus R) + b \tag{3.1}$$

where a is a scalar gain between the Hamming distance and W the power consumed.

3.4 Pearson's Correlation Coefficient

Once a way to model the `\power consumption` is on disposal, it is needed a way to compare the `\power estimation` to the measured traces. A helpful tool for finding this relationship is through `\Pearson correlation coefficient` [8], which is shown in equation 3.2 as follows:

[8] Kjell Johnson Max Kuhn. *Applied Predictive Modeling*. Springer, 1st edition, 2013. ISBN 978-1461468486

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{E[(X-\mu_X)^2]E[(Y-\mu_Y)^2]}} \tag{3.2}$$

This `\correlation coefficient` will always be in the range [-1, 1]. It describes how closely the random variables X and Y are related [9]:

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Agosto 2006. ISBN 978-0387310732

- If Y always increases when X increases, it will be 1;
- If Y always decreases when X increases, it will be -1;

- If Y is totally independent of X, it will be 0.

These equations are referred as \normalized cross-correlation [10]. There are typically used to pick out patterns in noisy signals. For example, in \digital imaging, correlation can be used to find where an object is in a room. In the side-channel attack, algorithm will be looking for a pre-calculated \power consumption pattern, in the \noisy measured \power traces signals.

[10] Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 1st edition, April 2020. ISBN 978-1108455145

3.5 *The Linear Correlation Factor*

A linear model implies some relationships between the variances of the different terms considered as random variables: $\sigma_W^2 = a^2\sigma_H^2 + \sigma_b^2$. Classical statistics introduce the correlation factor ρ_{WH} between the \Hamming distance and the measured power to assess the \linear model fitting rate. It is the \co-variance between both random variables normalized by the product of their standard deviations. Under the \uncorrelated noise assumption, this definition leads to equation 3.3

$$\rho_{WH} = \frac{\operatorname{cov}(W,H)}{\sigma_W\sigma_H} = \frac{a\sigma_H}{\sigma_W} = \frac{a\sigma_H}{\sqrt{a^2\sigma_H^2 + \sigma_b^2}} = \frac{a\sqrt{m}}{\sqrt{ma^2 + 4\sigma_b^2}} \tag{3.3}$$

This equation complies with the well known property: $-1 \leq \rho w H \leq +1$: [11] for a perfect model the \correlation factor tends to ±1 if the \variance of noise tends to 0, the sign depending on the sign of the linear gain a. If the model applies only to l independent bits amongst m, a \partial correlation still exist as shown in the to equation 3.4.

[11] Kjell Johnson Max Kuhn. *Applied Predictive Modeling*. Springer, 1st edition, 2013. ISBN 978-1461468486

$$\rho_{WH_{l/m}} = \frac{a\sqrt{l}}{\sqrt{ma^2 + 4\sigma_b^2}} = \rho_{WH}\sqrt{\frac{l}{m}} \tag{3.4}$$

3.6 *Secret Inference Based on Correlation Power Analysis*

The relationships written above show that if the model is valid the correlation factor is maximized when the \noise variance is minimum. This means that ρ_{WH} can help to determine the reference state R. Assume, just like in DPA, that a set of known but randomly varying data D and a set of related \power consumption W are available. If the 2^m possible values of R are scanned exhaustively they can be ranked by the \correlation factor they produce when combined with the observation W. This is not that expensive when considering an 8-bit micro-controller, the case with many of today's smart cards [12] (jut for example), as only 256 values are to be tested[13].

[12] R. Verdult L. Batina J. Balasch, B. Gierlichs and I. Verbauwhede. *Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. Lecture Notes in Computer Science*, volume 7178. Springer, in o. dunkelman, editor, ct-rsa edition, 2012

[13] On 32 -bit architectures this exhaustive search cannot be applied as such. But it is still possible to work with partial correlation or to introduce prior knowledge.

Let R be the true reference and $H = H(D \oplus R)$ the right prediction on the \Hamming distance. Let R' represent a candidate value and H' the related model $H' = H(D \oplus R')$. Assume a value of R' that has k bits that differ from those of R, then: $H(R \oplus R') = k$. since b is independent from other variables, the \correlation test leads to equation 3.5:

$$\rho_{WH'} = \frac{\operatorname{cov}(aH + b, H')}{\sigma_W \sigma'_H} = \frac{a}{\sigma_W} \frac{\operatorname{cov}(H, H')}{\sigma'_H} = \\ = \rho_{WH}\rho_{HH'} = \rho_{WH}\frac{m-2k}{m} \quad (3.5)$$

This formula shows how the \correlation factor is capable of rejecting wrong candidates for R. For instance, if a single bit is wrong amongst an 8-bit word, the correlation is reduced by $1/4$. If all the bits are wrong, this is $R' = \neg R$, then an anti-correlation should be observed with $\rho_{WH'} = -\rho_{WH}$. In absolute value or if the linear gain is assumed positive ($a > 0$), there cannot be any R' leading to a higher correlation rate than R. This proves the uniqueness of the solution and therefore how the reference state can be determined.

This analysis can be performed on the \power trace assigned to a piece of \code while manipulating known and varying \data. If we assume that the handled data is the result of a \XOR operation between a \secret key word K and a known message word $M, D = K \oplus M$, the procedure described above, that is an \exhaustive search on R and correlation test, should lead to $K \oplus R$ associated with $\max(\rho_{WH})$. Indeed if a \correlation occurs when M is handled with respect to R_1, another has to occur later on, when $M \oplus K$ is manipulated in turn, possibly with a different reference state R_2 (in fact with $K \oplus R_2$ since only M is known). For instance, when considering the first \AddRoundKey function at the beginning of the \AES algorithm \embedded on an 8-bit processor, it is obvious that such a method leads to the whole \key masked by the constant reference byte R_2. If R_2 is the same for all the \key bytes, which is highly plausible, only 2^8 possibilities remain to be tested by exhaustive search to infer the entire key material. This complementary brute force may be avoided if R_2 is determined by other means or known to be always equal to 0 [14]. [15]

[14] E. Prouff. *DPA Attacks and S-boxes. InFast Software Encryption*. Springer, 4th edition, 2005

[15] Valid just, for certain low EMI chips

3.7 *The Estimation*

In a \real case with a set of N \power curves W_i and N associated \random data words M_i, for a given reference state R the known \data words produce a set of N predicted Hamming distances $H_{i,R} = H(M_i \oplus R)$. An estimate $\hat{\rho}_{WH}$ of the \correlation factor ρ_{WH} is given by the equation 3.6:

$$\hat{\rho}_{WH}(R) = \frac{N \sum W_i H_{i,R} - \sum W_i \sum H_{i,R}}{\sqrt{N \sum W_i^2 - (\sum W_i)^2} \sqrt{N \sum H_{i,R}^2 - (\sum H_{i,R})^2}} \tag{3.6}$$

where the summations are taken over the N samples $(i = 1, N)$ at each time step within the power traces $W_i(t)$.

It is theoretically difficult to compute the \variance of the \estimator [16] $\hat{\rho}_{WH}$ with respect to the number of available samples N. In practice a few hundred experiments suffice to provide a workable estimate of the \correlation factor. N has to be increased with the \model variance $m/4$ [17] and in presence of measurement noise level [18]. It is shown that this approach can be seen as a \maximum likelihood model fitting procedure [19] when R is exhausted to maximize $\hat{\rho}_{WH}$.

[16] Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 1st edition, April 2020. ISBN 978-1108455145

[17] Being higher on a 32 -bit architecture

[18] Next results will show that this is more than necessary for conducting reliable tests

[19] Kjell Johnson Max Kuhn. *Applied Predictive Modeling*. Springer, 1st edition, 2013. ISBN 978-1461468486

4 Hypothesis Evaluation

Contents

4.1 *Comparison of Different Selection Functions*

The next notation will be used, unless specified otherwise. The following operators for the corresponding (\bitwise) logical operations is used: “." for AND, "+" for OR, " $\oplus$ " for \XOR. The operators $\boxplus$ and $\boxminus$ denote a modular addition and a modular subtraction, respectively. The two functions $\mathrm{MSB}(x)$ and $\mathrm{LSB}(x)$ are used to extract the most and the least significant byte from a \stream of bits x, respectively. The \S-Box layer of a \block cipher α is represented by S_α, which may involve the application of one or more S-boxes in parallel, depending on the input size and the specifications of the cipher. The symbol $L^{-1}_{i,\mathtt{Fant}}$ stands for the result of the inverse linear layer of Fantomas (Fant) computed with \L-box i, where $i \in \{0,1\}$. Finally, $\mathrm{HW}(x)$ denotes the \Hamming weight of x, whereas $\mathrm{HD}(x,y) = \mathrm{HW}(x \oplus y)$ is the \Hamming distance between x and y

The \attack is not restricted to the $\oplus$ operation only. It also applies to many other operators often encountered in \secret key cryptography. For instance, other arithmetic, \logical operations or

\look-up tables (LUT) can be treated in the same manner by using $\mathrm{H}(\mathrm{LUT}(M \star K) \oplus R)$, where $\star$ represents the involved function i.e. $\oplus, +, -$, OR, AND, or whatever operation.

Let's notice that the ambiguity between K and $K \oplus R$ is completely removed by the \substitution boxes encountered in \secret key algorithms thanks to the \non-linearity of the corresponding LUT: this may require to exhaust both K and R, but only once for R in most cases. To conduct an analysis in the best conditions, emphasizing the benefit of correctly \modeling the whole \machine word that is actually handled and its transition with respect to the reference state R which is to be determined as an unknown of the problem.

Table 4.1 summarizes the non-linearity NL and the mean correlation coefficient difference $\bar{\delta}$ for a total of 16 different selection functions, which are divided into four groups.

Selection function	n	m	NL	$\bar{\delta}$	$\mathrm{SE}_{\bar{\delta}}$
$\varphi_1(x,k) = x \cdot k = x \vee k$	16	8	16384	-0.005	0.074
$\varphi_2(x,k) = x + k = x \wedge k$	16	8	16384	-0.018	0.060
$\varphi_3(x,k) = x \oplus k$	16	8	0	-0.153	0.168
$\varphi_4(x,k) = x \boxplus k$	16	8	0	0.127	0.011
$\varphi_5(x,k,c) = x \boxplus k \boxplus c$	17	8	0	0.121	0.010
$\varphi_6(x \oplus k) = S_{AES}(x \oplus k)$	8	8	112	0.586	0.008
$\varphi_7(x \oplus k) = S_{L_{Blk}}(x \oplus k)$	4	4	4	0.342	0.008
$\varphi_8(x \oplus k) = S_{L_{Blk}}(x \oplus k)$	8	8	64	0.235	0.006
$\varphi_9(x \oplus k) = S_{Picc}(x \oplus k)$	4	4	4	0.339	0.019
$\varphi_{10}(x \oplus k) = S_{Picc}(x \oplus k)$	8	8	64	0.259	0.006
$\varphi_{11}(x \oplus k) = S_{PRIN}(x \oplus k)$	4	4	4	0.269	0.010
$\varphi_{12}(x \oplus k) = S_{PRIN}(x \oplus k)$	8	8	64	0.138	0.004
$\varphi_{13}(x \oplus k) = \mathrm{LSB}\left(L_{1,\mathrm{Fant}}(x \oplus k)\right)$	8	8	0	0.087	0.015
$\varphi_{14}(x \oplus k) = \mathrm{MSB}\left(L_{1,\mathrm{Fant}}(x \oplus k)\right)$	8	8	0	0.041	0.014
$\varphi_{15}(x \oplus k) = \mathrm{LSB}\left(L^{-1}_{2,\mathrm{Fant}}(x \oplus k)\right)$	8	8	0	0.136	0.007
$\varphi_{16}(x \oplus k) = \mathrm{MSB}\left(L^{-1}_{2,\mathrm{Fant}}(x \oplus k)\right)$	8	8	0	0.083	0.018

Table 4.1: Leakages of different selection functions (n and m are the input and output size of the selection function in bits, NL is the non-linearity of the selection function, $\bar{\delta}$ is the mean correlation coefficient difference, and $\mathrm{SE}_{\bar{\delta}}$ is the standard error for a 95% confidence interval).

- $\bar{\delta}$ and SE_{δ} are the mean and the \standard error for a 95% \confidence interval, respectively.
- Block (Blk)
- Fantomas (Fant)
- Piccolo (Picc)
- PRINCE (PRIN)

The first group of selection functions comprises the three logical operations AND, OR, and XOR, which all have a negative value for the mean correlation coefficient difference $\bar{\delta}$. This means that using one of these logical operations as a selection function for a CPA attack is not a very good option. As our results show, only the XOR are sometimes able to recover the correct key k^*, but not AND and OR, whereby AND is slightly more efficient than OR and XOR.

One can notice the contrast between the huge `\non-linearity` of the AND and OR selection functions on the one side, and all other selection functions listed in Table 4.1 on the other side. It is also interesting to note that these high values of non-linearity are accompanied by (relatively) poor values for the correlation coefficient difference. In the case of the bit-wise logical operations, it seems the high non-linearity values do not provide the useful leakage one normally would expect. This contrasts with the conventional wisdom saying that the higher the non-linearity of a selection function, the more information it leaks in the `\Side-channel analysis` (SCA).

The `\XOR` function is a convenient chosen function because offer a god statistical characterization, with zero non-linearity of the selection function and with the largest value in the mean $\bar{\delta}$ correlation coefficient difference, and also a largest value in the standard error $\mathrm{SE}_{\bar{\delta}}$ for a 95% confidence interval for , it can be confirmed in table 4.1 where other common function in addition, are also evaluated in a statistical fashion.

4.2 *Correlation Power Analysis (CPA) Evaluation*

The idea is to use `\Pearson correlation coefficient` as a distinguisher instead of the `\difference of means` test[1]. CPA combined with classical leakage models like the `\Hamming weight` and `\Hamming distance`, is very efficient on a majority of `\hardware architectures`. Subsequent works on CPA effectiveness essentially consisted in applying `\signal filtering` techniques or `\pre-processing` [2] on the `\leakage traces`

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Agosto 2006. ISBN 978-0387310732

[2] J. G. Proakis and D. K. Manolakis. *Digital Signal Processing*. Prentice-Hall, 4th edition, May 2006

4.2.1 *Attack Description*

In a `\correlation power analysis` attack, the adversary computes, for each plain-text $\mathbf{X_i}$ and each subkey hypothesis $\hat{k}$, an hypothesis tuple $\hat{z}_i$ such that $\hat{z}_i = F\left(x_i, \hat{k}\right)$. Then, a so-called model function $m(\cdot)$ is chosen from $\mathbb{F}_2^m$ into $\mathbb{R}$ and it is applied to each hypothesis which leads to the construction of a new set of so-called `\predictions` [3] $(h_i)_{i \leqslant N} \leftarrow \vec{H} = \mathrm{m}(F(X, \hat{k}))$. This latter set of hypothesis is then compared to the set of leakages $\left(\vec{\ell}_i\right)_{i \leqslant N} \leftarrow \overrightarrow{\mathbf{L}}$ to assess on the

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Agosto 2006. ISBN 978-0387310732

likelihood [4] of $\hat{k}$ as a candidate for one of the subkeys k. The core principle of a CPA is to make the comparison by estimating the \linear correlation between $\vec{H}$ and each coordinate of $\overrightarrow{\mathrm{L}}$ independently [5].

[4] Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 1st edition, April 2020. ISBN 978-1108455145

[5] Kjell Johnson Max Kuhn. *Applied Predictive Modeling*. Springer, 1st edition, 2013. ISBN 978-1461468486

Algorithm 1: Algorithm: CPA - Correlation Power Analysis

```
Input : a set of $d$ -dimensional leakages $\left(\vec{\ell}_i\right)_{i \leqslant N}$ and the
  corresponding plaintexts $(x_i)_{i \leqslant N}$ , a model function m $(\cdot)$
Output: A candidate subkey
/* Leakage mean and variance processing */
for $i = 0$ to $N - 1$ do
    $\mu_{\overrightarrow{\mathrm{L}}} = \mu_{\overrightarrow{\mathrm{L}}} + \vec{\ell}_i$
    $\sigma_{\overrightarrow{\mathbf{L}}} = \sigma_{\overrightarrow{\mathbf{L}}} + \vec{\ell}_i^2$
    $\mu\,\overrightarrow{\mathbf{L}} = 1/N \cdot \mu\,\overrightarrow{\mathbf{L}}$
    $\mathrm{var}_{\overrightarrow{\mathrm{L}}} = 1/N \cdot \sigma_{\overrightarrow{\mathrm{L}}} - \mu^2_{\overrightarrow{\mathrm{L}}}$
/* Hypothesis mean and variance processing */
for $\hat{k} = 0$ to $2^n - 1$ do
    for $i = 0$ to $N - 1$ do
        $z \leftarrow \mathrm{m}\left(F\left(x_i, \hat{k}\right)\right)$ $\mu_{\hat{k}} = \mu_{\hat{k}} + z$ $\sigma_{\hat{k}} = \sigma_{\hat{k}} + z^2$
    $\mu_{\hat{k}} = \mu_{\hat{k}}/N$ $\mathrm{var}_{\hat{k}} = \sigma_{\hat{k}}/N - \mu^2_{\hat{k}}$
/* Correlations processing */
for $\hat{k} = 0$ to $2^n - 1$ do
    /* Test hypothesis $\hat{k}$ for all leakage coordinates /* for $u = 0$
     to $d - 1$ do
        / * Instantaneous attack (at time $u$ ) /* cov $= 0$ for $i = 0$
         to $N - 1$ do
            cov $=$ cov $+ \mathrm{m}\left(F\left(x_i, \hat{k}\right)\right) \times \vec{\ell}_i[u]$
        cor$[\hat{k}][u] =$
          $\left(1/N \times \mathrm{cov} - \mu_k \times \mu_{\overrightarrow{\mathbf{L}}}[u]\right) / \sqrt{\mathrm{var}_k \times \mathrm{var}_{\overrightarrow{\mathbf{L}}}[u]}$
/* Most 1 ikely candidate selections */
candidate $= \mathrm{argmax}_{\hat{k}}\left(\max_u \mathrm{cor}[\hat{k}][u]\right)$
return candidate
```

4.2.2 *CPA Pseudo-code and Algorithm Description*

The algorithm 1 describes the linear regression analysis, providing a general overview on the particularities for the CPA algorithm implementation.

Steps 1-5 in algorithm 1 process the sampling mean and the sampling variance of each coordinates of the \leakage vectors. It results in two vectors $\mu_{\overrightarrow{\mathbf{L}}}$ and $\sigma_{\overrightarrow{\mathbf{L}}}$ with same dimension as $\overrightarrow{\mathrm{L}}$. Steps 6-12 in algorithm 1 process the (sampling) mean and the (sampling) variance of the hypothesis sample $\left(\mathrm{m}\left(F\left(x_i,\hat{k}\right)\right)\right)_i$ for each `\key candidate` $\hat{k}$. When N grows, those latter mean and variance quickly tend toward $\mathbb{E}[(\mathfrak{m}(F(X,\hat{k}))]$ and $\mathrm{var}[(\mathfrak{m}(F(X,\hat{k}))]$, which can be directly deduced from $m(\cdot)$ and F. For example if F is an sbox and $m(\cdot)$ is the `\Hamming weight` function, we have $\mathbb{E}[(\mathfrak{m}(F(X,\hat{k}))] = m/2$ and $\mathrm{var}[(\mathfrak{m}(F(X,\hat{k}))] = m/4$ (assuming X is uniform). Eventually, Steps 13-18 in algorithm 1 process, for each `\key candidate` $\hat{k}$, the `\correlation` between $\left(\mathrm{m}\left(F\left(x_i,\hat{k}\right)\right)\right)_i$ and each sample $\left(\vec{\ell}_i[u]\right)_i$ where u denotes the u^{th} coordinate of the `\leakage vectors`.

4.2.3 *CPA Efficiency*

The law of `\total expectation` in equation 4.1 implies:

$$\mathrm{cov}(X,Y) = \mathbb{E}[\mathbb{E}[(X-\mathbb{E}[X])(Y-\mathbb{E}[Y])|X]] \tag{4.1}$$

where the conditional means are viewed as random variables that functionally depend on X [6]. Due to the linearity of the mean, the equation 4.1 is also equivalent to $\mathrm{cov}(X,Y) = \mathbb{E}[(X-\mathbb{E}[X])\mathbb{E}[(Y-\mathbb{E}[Y]) \mid X]]$. When applied in the context of the correlation coefficients computation in algorithm 1, this equation leads to a significant efficiency improvement.

[6] Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 1st edition, April 2020. ISBN 978-1108455145

4.2.4 *Detailed Complexities of the CPA attack*

A detailed counting of the elementary operations for the CPA based attack is given in the table 4.2 along with the memory budget needed for the implementation of the elementary operations for the CPA based attack.

Method	Operations Complexity	Memory Complexity
Classical CPA	$d \times 2^n \times (2N+6)$	$2^{n+1}+2d$
CPA with Improvement	$2N + 2^n \times (1 + d \times (2^n+6))$	$2^n(d+4)$

Table 4.2: Detailed Complexities of the CPA attack. Number of operations with respect to the number of traces N and the dimension d of the traces.

4.3 *Linear Regression Analysis (LRA) Evaluation*

4.3.1 *Attack Description*

In `\LRA`, the adversary chooses a so-called basis of functions $(\mathrm{m}_p)_{1\leqslant p\leqslant s}$ with the only condition that m_1 is a constant function (usually $\mathrm{m}_1 = 1$). Then, for each x_i and each `\subkey hypothesis` $\hat{k}$, the prediction $\hat{z}_i = F\left(x_i,\hat{k}\right)$ is calculated. The basis functions m_p are then applied to the $\hat{z}_i$ independently, leading to the construction of a $(N \times s)$ -matrix $\mathcal{M}_{\hat{k}} \doteq \left(\mathrm{m}_p\left(F\left(x_i, \hat{k}\right)\right)\right)_{i,p}$. The comparison of this matrix with the set of d -dimensional leakages $\left(\vec{\ell}_i\right)_{i \leqslant N} \leftarrow \overrightarrow{\mathbf{L}}$ is done by processing a \linear regression of each coordinate of $\vec{\ell}_i$ in the basis formed by the row elements of $\mathcal{M}_k$. Namely, a real-valued $(s \times d)$ -matrix $\mathcal{B}_k$ with column vectors $\overrightarrow{\beta_1}, \ldots, \vec{\beta}_d$ is estimated in order to minimize the error when approximating $\vec{\ell}_i^{\top}$ by $\left(\mathrm{m}_1\left(F\left(x_i, \hat{k}\right)\right), \cdots, \mathrm{m}_s\left(F\left(x_i, \hat{k}\right)\right)\right) \times \mathcal{B}_{\hat{k}}$. The \matrix $\mathcal{B}_{\hat{k}}$ is defined in equation 4.2 such that:

$$\mathcal{B}_{\hat{k}} = \underbrace{\left(\mathcal{M}_k^{\top} \times \mathcal{M}_{\hat{k}}\right)^{-1} \times \mathcal{M}_k^{\top}}_{\mathcal{P}_k} \times \mathcal{L} \tag{4.2}$$

where $\mathcal{L}$ denotes the $(N \times d)$ -matrix with the $\vec{\ell}_i^{\top}$ as row vectors. In the following, the u^{th} column vector of $\mathcal{L}$ (composed of the u^{th} coordinate of all the $\vec{\ell}_i$) is denoted by $\overrightarrow{\mathcal{L}}[u]$. Moreover, the $(s \times N)$ -matrix $\left(\mathcal{M}_{\hat{k}_\sigma}^{\top} \times \mathcal{M}_k\right)^{-1} \times \mathcal{M}_{\hat{k}}^{\top}$, which does not depend on the \leakage values, is denoted by $\mathcal{P}_{\hat{k}}$. To quantify the \estimation error, the goodness of \fit model is used and the \correlation coefficient of determination $\mathcal{R}^2$ is computed for each u. The latter is defined by $\mathcal{R}^2 = 1-$ SSR/SST, where SSR and SST respectively denote the \residual sum of squares (deduced from $\mathcal{B}_{\hat{k}}$) and the \total sum of squares (deduced from $\mathcal{L}$) [7].

[7] Kjell Johnson Max Kuhn. *Applied Predictive Modeling*. Springer, 1st edition, 2013. ISBN 978-1461468486

4.3.2 *LRA Efficiency*

Let us focus on the \best candidate selection step in a classical \LRA. Each \subkey hypothesis k is first associated with a score which is the greatest instantaneous coefficient of determination when testing it for all temporal coordinates u. It is denoted by $\max_u \mathcal{R}[\hat{k}][u]$ in algorithm 4.3. The second phase of the selection consists in the processing of the maximum $\operatorname{argmax}_k \left(\max_u \mathcal{R}[\hat{k}][u]\right)$. The purpose of the latter step is to identify the candidate that maximises the greatest instantaneous coefficient. Implicitly, such a classical approach by total maximisation of the distinguisher value assumes that the most likely candidate corresponds to the greatest value taken by the distinguisher not only over all \subkey hypothesis but also over all the \leakage times. This assumption relies on another one, often done in the \embedded security community, which states that the value of a distinguisher computed between wrong hypothesis (computed for a wrong subkey value or a wrong time) and the leakage values tends toward its minimum value (often 0) when the \sample size N increases. However, as already noticed in several experiments, both assumptions are often not verified in practice, where the adversary must for instance deal with the \ghost peaks phenomenon. The situation is even worst for the \LRA attacks since the vector of coefficients β (and thus the set of \predictions) depends not only on $\hat{k}$ but also on the attack time u (it is not the case for the \CPA as the \predictions $\mathrm{m}(F(x, \hat{k}))$ are the same for each instantaneous \attack). The strength of the \LRA, namely its ability to adapt to the instantaneous \leakage, is also its weakness as it makes it difficult to compare the different instantaneous attacks results.

4.3.3 LRA Pseudo-code

The algorithm 2 describes the linear regression analysis, providing a general overview on the particularities for the LRA algorithm implementation.

Algorithm 2: LRA - Linear Regression Analysis

```
Input : a set of $d$ -dimensional leakages $\left(\vec{\ell}_i\right)_{i \leqslant N}$ and the
 corresponding plain-texts $(x_i)_{i \leqslant N}$ , a set of model functions
 $\left(\mathrm{m}_p\right)_p \leqslant_s$ Output: A candidate subkey $\hat{k}$
/* Processing of the leakage Total Sum of Squares (SST) /* for
 $i = 0$ to $N-1$ do
    $\mu_{\vec{\mathrm{L}}} = \mu_{\vec{\mathrm{C}}} + \vec{\ell}_i$
    $\sigma_{\vec{\mathrm{L}}} = \sigma_{\vec{\mathrm{L}}} + \vec{\ell}_i^2$
$\overrightarrow{\mathrm{SST}} = \sigma \overrightarrow{\mathrm{L}} - 1/N \cdot \mu^2_{\vec{\mathrm{L}}}$
/* Processing of the $2^n$ predictions matrices /*
$\mathcal{M}_k$ and $\mathcal{P}_k$ /*
for $\hat{k} = 0$ to $2^n - 1$ do
    /* Construct the matrix $\mathcal{M}_{\hat{k}}$ and $\mathcal{P}_{\hat{k}}$ /*
    for $p = 1$ to $s$ do
        for $i = 0$ to $N-1$ do
            $\mathcal{M}_{\hat{k}}[i][p] \leftarrow m_p\left[F\left(x_i, \hat{k}\right)\right]$
    $\mathcal{P}_{\hat{k}} = \left(\mathcal{M}_{\hat{k}}^\top \times \mathcal{M}_{\hat{k}}\right)^{-1} \times \mathcal{M}_{\hat{k}}^\top$
for $\hat{k} = 0$ to $2^n - 1$ do
    /* Test hyp. $\hat{k}$ for all leakage coordinates */
    for $u = 0$ to $d-1$ do
        /* Instantaneous attack (at time $u$) */
        $\vec{\beta} = \mathcal{P}_{\hat{k}} \times \overrightarrow{\mathcal{L}}[u]$
        /* Compute an estimator $\vec{E}$ of
         $\overrightarrow{\mathcal{L}}[u] = \left(\vec{\ell}_0[u], \cdots, \vec{\ell}_{N-1}[u]\right)^\top$ */
        $\vec{E} = \mathcal{M}_{\hat{k}} \times \vec{\beta}$
        /* Compute the estimation error (i.e. the SSR) */
        $\mathrm{SSR} = 0$
        for $i = 0$ to $N-1$ do
            $\mathrm{SSR} = \mathrm{SSR} + \left(\vec{E}[u] - \vec{\ell}_i[u]\right)^2$
        /* Compute the coefficient of determination */
        $\mathcal{R}[\hat{k}][u] = 1 - \mathrm{SSR}/\overrightarrow{\mathrm{SST}}[u]$
/* Most likely candidate selections */
candidate = $\mathrm{argmax}_{\hat{k}}\left(\max_u \mathcal{R}[\hat{k}][u]\right)$
return candidate
```

4.3.4 *Detailed Complexities of the LRA attack*

A detailed counting of the elementary operations for the LRA based attack is given in the table 4.3 in order to provide the operation complexity.

Method	Operations Complexity
Classical LRA	$2^n dN(2sd + 2s + 3)$
LRA with Improvement	$N(d+1) + 2^n d\,(2^n(sd + 2s + 3) + 3)$

Table 4.3: Operations complexity of the LRA attack. The number of operations with respect to the number of traces N, the dimension d of the traces and the size s of the regression basis.

In addition a memory budget needed for the implementation of the elementary operations for the LRA based attack is given in the following table 4.4 in order to provide the memory resources usage.

Method	Memory Complexity
Classical LRA	$d + 2^n * N * s + 2^n * N * s + N * d$
LRA with Improvement	$d + 2^{2n} * s + 2^{2n} * s + 2^n * d$

Table 4.4: Memory complexity of the LRA attack. The number of operations with respect to the number of traces N, the dimension d of the traces and the size s of the regression basis.

4.4 *Comparison with Simple DPA*

Just considering the practical implementation of DPA against the \AES substitutions (1 st round). In fact the well-known DPA attack [8] works quite well only if the following assumptions are fulfilled:

[8] E. Prouff. *DPA Attacks and S-boxes. InFast Software Encryption*. Springer, 4th edition, 2005

- 1. \Word space assumption: within the word hosting the predicted bit, the contribution of the non-targeted bits is independent of the targeted bit value. Their average influence in the curves pack of 0 is the same as that in the curves pack of 1. So the attacker does not need to care about these bits.
- 2. \Guess space assumption: the predicted value of the targeted bit for any wrong \subkey guess does not depend on the value associated to the correct guess.
- 3. \Time space assumption: the power consumption W does not depend on the value of the targeted bit except when it is explicitly handled.

But when confronted to the experience, the attack comes up against the following issues.

Issue I. For the correct guess, \DPA peaks appear also when the targeted bit is not explicitly handled. This is worth being noticed albeit not really embarrassing. However this contradicts the third assumption.

Issue II. Some DPA peaks also appear for wrong guesses: they are called \ghost peaks. This fact is more problematic for making a decision and comes in contradiction with the second assumption.

The reason why wrong guesses may generate DPA peaks is that the distributions of an \S-Box output bit for two different guesses are deterministic and so possibly partially correlated [9]. The following example is very convincing about that point. Let's consider the leftmost bit of the fifth \S-Box of the \AES when the input data D varies from 0 to 63 and combined with two different subkeys: $\mathrm{MSB}\left(\mathrm{SBox}_5(D \oplus 0x00)\right)$ and $\mathrm{MSB}\left(\mathrm{SBox}_5(D \oplus 0x36)\right)$. Both series of \bits are respectively listed hereafter, with their \bitwise XOR on the third line:

1101101010010110001001011001001110101001011011010101001000101101
1001101011010110001001011110100101010110101101001010100100111001
0100000001000000000000000100000100000100000001000000000000010100

The third line contains 8 set \bits, revealing only eight \errors of prediction among 64. This example shows that a \wrong guess, say 0, can provide a \good prediction at a rate of 56/64, that is not that far from the correct one $0x36$. The result would be equivalent for any other pair of subkeys K and $K \oplus 0x36$. Consequently a substantial concurrent DPA peak will appear at the same location than the right one. The \weakness of the contrast will disturb the guesses ranking especially in presence of high \Signal to Noise Ratio (SNR). [10]

Issue III. The true DPA peak given by the right guess may be smaller than some \ghost peaks, and even null or negative! This seems somewhat amazing and quite confusing for an attacker. The reasons must be searched for inside the crudeness of the optimistic first assumption.

[9] E. Prouff. *DPA Attacks and S-boxes. InFast Software Encryption*. Springer, 4th edition, 2005

[10] This particular counter-example proves that the ambiguity of DPA does not lie in imperfect estimation but in wrong basic hypothesis.

5 Implementation

Contents

5.1 Experimental Setup and Results

For each `\power trace` a pair of the `\plain-text` and the `\encrypted \cipher-text` is required[1]. Therefore all the information needed is on disposal with excepting the `\secret key`. It is the goal of the `\differential power analysis` to extract the secret key using the mentioned power traces, `\plain-text`, `\cipher-text` and the knowledge of the `\encryption algorithm` by creating the hypothesis of the `\power consumption` and correlating it to the `\measured traces`.

[1] At this point you should either given or were able to measure the power consumption (traces) of any electronic device by yourself

5.2 Correlational Power Analysis - Key Recovery

5.2.1 Schedule

1. Plot one `\power trace`, checking that it is complete.
2. Check the `\alignment of traces` (they overlay correctly, triggering works).
3. Select the appropriate part of the traces (for example the part containing the first round). Reading the appropriate number of traces.

4. Depending on the measurements, a correction of mean values may have to be performed (if the measurements "wander" in voltage over time). It can be done so by subtracting from each trace its `\mean value`.

5. Recover the `\secret key` using the DPA with correlation coefficients.

5.2.2 *Methods*

Use DPA with `\correlation analysis`. The core of the method is as follows

1. Choose an intermediate value that depends on `\data` and `\key`

2. Measure the power traces while encrypting the data

3. Build a matrix of `\hypothetical intermediate values` inside the cipher for all possible keys and traces

4. Using a `\power model`, compute the matrix of hypothetical power consumption for all keys and traces

5. Statistically evaluate which `\key hypothesis` best matches the measured power in each individual time

For better vision of the previous method take a look in the figure 5.1. The `\right key` (part of key) is determined by Key Hypothesis -> Intermediate Value -> Power Consumption, best correlating to actually measured consumption at some moment. Repeating the analysis for other parts of the key, until the whole key is determined, the idea is shown in the figure 5.1.

5.2.3 *AES Weakness*

In `\AES`, the `\plain text block` is first XORed with the primary key and then goes through 10 rounds of processing. Each round consists of:

- SubByte

- ShiftRow

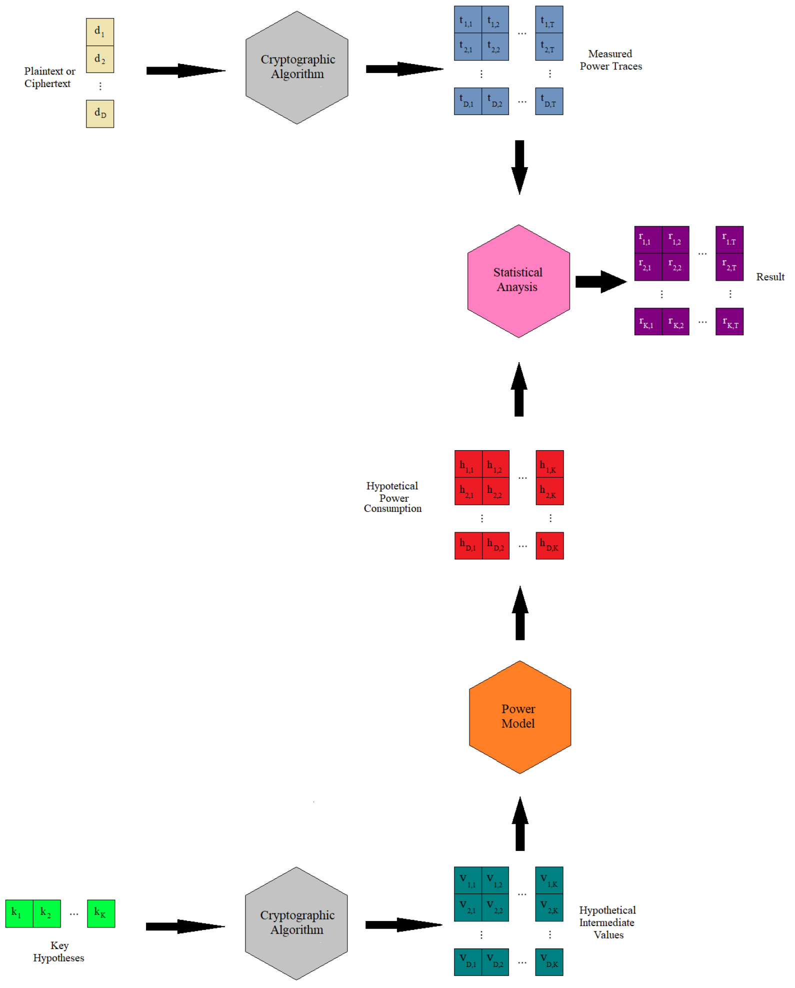
- MixColumn

- AddRoundKey

and it could be visualized in figure 5.2.

By looking for the `\roundkey` of the last round (the only one without the `\MixColumn` step).

In order to implement a `\DPA attack`, an attacker first observes m `\encryption` operations and T power traces captures with k samples each.



Plaintext or Ciphertext
d_1
d_2
d_D
Cryptographic Algorithm
Measured Power Traces
$t_{1,1}$ $t_{1,2}$ … $t_{1,T}$
$t_{2,1}$ $t_{2,2}$ … $t_{2,T}$
$t_{D,1}$ $t_{D,2}$ … $t_{D,T}$
Statistical Anaysis
Result
$r_{1,1}$ $r_{1,2}$ … $r_{1,T}$
$r_{2,1}$ $r_{2,2}$ … $r_{2,T}$
$r_{K,1}$ $r_{K,2}$ … $r_{K,T}$
Hypotetical Power Consumption
$h_{1,1}$ $h_{1,2}$ … $h_{1,K}$
$h_{2,1}$ $h_{2,2}$ … $h_{2,K}$
$h_{D,1}$ $h_{D,2}$ … $h_{D,K}$
Power Model
k_1 k_2 … k_K
Key Hypotheses
Cryptographic Algorithm
Hypothetical Intermediate Values
$v_{1,1}$ $v_{1,2}$ … $v_{1,K}$
$v_{2,1}$ $v_{2,2}$ … $v_{2,K}$
$v_{D,1}$ $v_{D,2}$ … $v_{D,K}$

Figure 5.1: Block Diagram illustrating steps 3 to 5 of the CPA attack. It also outlines the architecture of a framework that could be carried out to classify various estimators (pink hexagon).

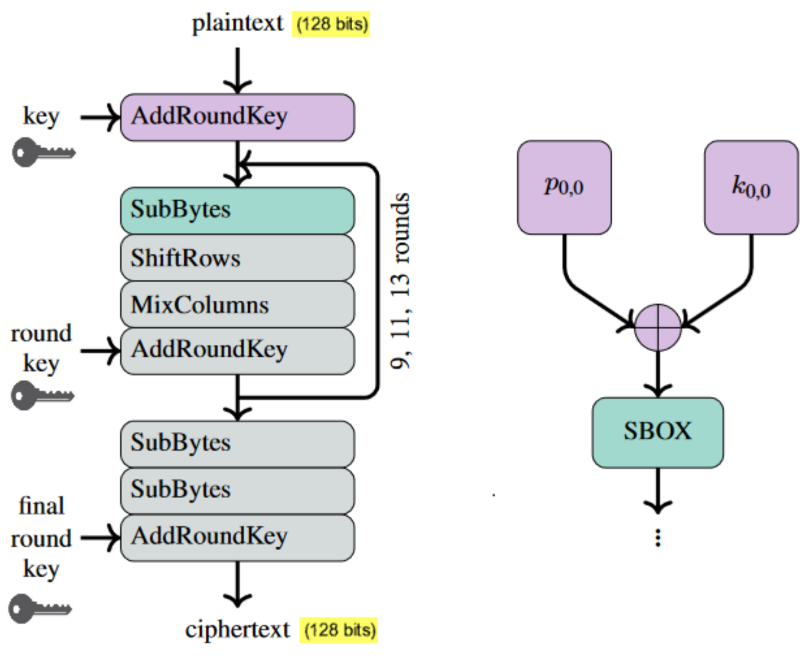


Figure 5.2: Block diagram for the AES algorithm, for a general overview.

The algorithm 3 presents a pseudo-code for a common implementation of AES-128 cipher. In this algorithm is marked the precise point were the attack is performed on the 1st round of the encryption process.

Algorithm 3: AES-128 Cipher

```
Input: 128 bits (plain-text)
Output: 128 bits (cipher-text)
w: 44 words, 32 bits each (expanded key)
state = in;
AddRoundKey(state, w[0, Nb-1]);
for round = 1 to Nr˘1 do
    SubBytes(state); // <—— Attack this point in round 1!
    ShiftRows(state);
    MixColumns(state);
    AddRoundKey(state, w[round*Nb, (round+1)*Nb-1]);
SubBytes(state);
ShiftRows(state);
AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1]);
out = state;
```

5.2.4 Correlation for Attack Performing

After taking some measurements, It'll put on disposal D power traces t, and each of these traces will have T data points. Using subscript notation, $t_{d,j}$ will refer to point j in trace $d(1 \leq d \leq D, 0 \leq j < T)$.

Also estimating the \power consumption in each trace using the model. Supposing there are I different subkeys that it is wanted to try. Then, $h_{d,i}$ will refer to the \power estimation in trace d, assuming that the \subkey is $i(1 \leq d \leq D, 0 \leq i < I)$.

With these \data, it could be evaluated how well the model and measurements match for each guess i and time j, by finding how t and h correlate over the D traces[2]. One way of calculating this is as indicated in equation 5.1:

[2] Kjell Johnson Max Kuhn. *Applied Predictive Modeling*. Springer, 1st edition, 2013. ISBN 978-1461468486

$$r_{i,j} = \frac{\sum_{d=1}^{D} \left[\left(h_{d,i} - \overline{h_i} \right) \left(t_{d,j} - \overline{t_j} \right) \right]}{\sqrt{\sum_{d=1}^{D} \left(h_{d,i} - \overline{h_i} \right)^2 \sum_{d=1}^{D} \left(t_{d,j} - \overline{t_j} \right)^2}} \tag{5.1}$$

There is an alternative form for the \correlation equation that is suited to use for online calculations - it allows to add one trace at a time without re-summing all of the past data. This form is presented in equation 5.2 as follows:

$$r_{i,j} = \frac{D \sum_{d=1}^{D} h_{d,i} t_{d,j} - \sum_{d=1}^{D} h_{d,i} \sum_{d=1}^{D} t_{d,j}}{\sqrt{\left(\left(\sum_{d=1}^{D} h_{d,i} \right)^2 - D \sum_{d=1}^{D} h_{d,i}^2 \right) \left(\left(\sum_{d=1}^{D} t_{d,j} \right)^2 - D \sum_{d=1}^{D} t_{d,j}^2 \right)}} \tag{5.2}$$

Note that these two proposals are equivalent [3].

[3] Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 1st edition, April 2020. ISBN 978-1108455145

5.2.5 Picking a Subkey

The last step is to use the values of $r_{i,j}$ to decide which \subkey matches the traces most closely. There are two steps to perform this:

- For each subkey i, find the highest value of $|r_{i,j}|$. This will discard the time information - just wanting to know how good the guess done, it was, - but not taking care about where those guess matched the trace.
- Looking at the maximum values for each subkey, find the highest value of $|r_i|$. The location i of this maximum is the best guess and it is correlated more closely with the traces than any other guess.

Note that, the working with \absolute values only, because the sign of the relationship doesn't care about. The important to know is that a \linear correlation exists [4].

[4] Kjell Johnson Max Kuhn. *Applied Predictive Modeling*. Springer, 1st edition, 2013. ISBN 978-1461468486

5.3 Laboratory Results

5.3.1 Real World Implementation

To be successful a \DPA attack needs to follow a few steps:

- Generate a huge amount of encryption/decryption operations using the target \cryptosystem and key.
- Trace, for each operation, both \power consumption and I/O (clear text and/or cipher text)
- Take a byte from the \S-Box [5] output and use it as a separator.
- Make the average out of each sub group.
- Repeat operation for each possible value of a byte (255 values) (8-bit MCU).
- Take the highest average and enjoy finding a byte of the key.
- Repeat for each byte of the key.

[5] E. Prouff. *DPA Attacks and S-boxes. InFast Software Encryption*. Springer, 4th edition, 2005

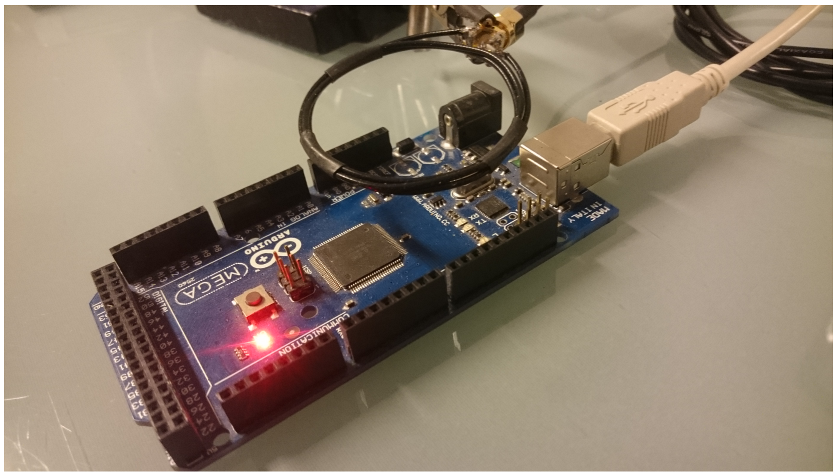


Figure 5.3: A small loop antenna can be used to pick up electromagnetic (EM) signals.

The laboratory set up is being shown in the figure 5.3. The \AES implementation on a former well known platform, to search for \side-channel effects. By averaging the \spectrogram over multiple traces, the following spectrogram can be obtained (captured at 3.5MHz). A repeating pattern occurs that is caused by the individual rounds of the AES algorithm, take a look in the figure 5.4.

Even though it was not possible to find any \correlation between the spectrogram and \key bits. Most likely, the time resolution of the spectrogram is too low, to get a good correlation [6].

[6] This is not a real problem

The next attempt was done by using a demodulated \time series signal. A new \static alignment was implemented, that works on

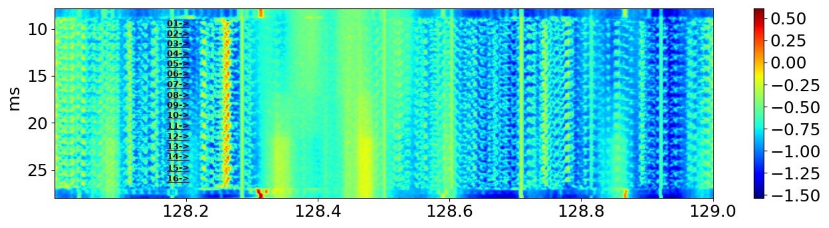


Figure 5.4: Spectrogram captured at 3.5MHz. A repeating pattern occurs that is caused by the individual rounds of the AES algorithm.

\time series signal was implemented. Now there are some good signals around the \clock frequency of 16MHz (141MHz in that figure 5.5). Interestingly the second \harmonics are also very strong (157MHz). They are very interesting as they could be analyzed without using an \up-converter.

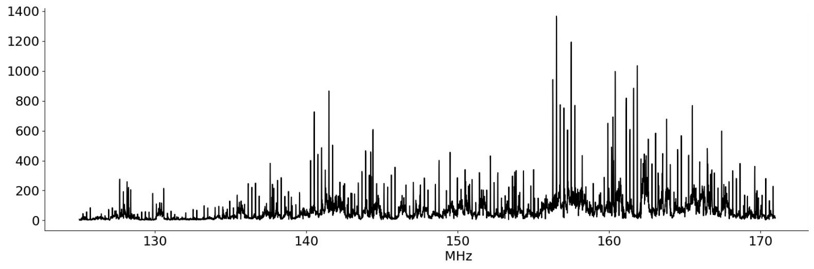


Figure 5.5: Demodulated time series signal. There are some good signals around the clock frequency of 16 MHz (141 MHz in the graph). The second harmonics are also very strong (157 MHz in the graph)

By selecting a carrier close to the \clock frequency at 16.253MHz, and with a 233kHz bandwidth \low pass filter [7] and applying in the signal an \amplitude demodulation. The resulting signal has a lot of \high-frequency components, looking a bit noisy, see the figure 5.6. Even though after 6022 traces this noise still remains, so it is actually caused by the CPU and could contain \side-channel information.

[7] 70 times lower than clock frequency

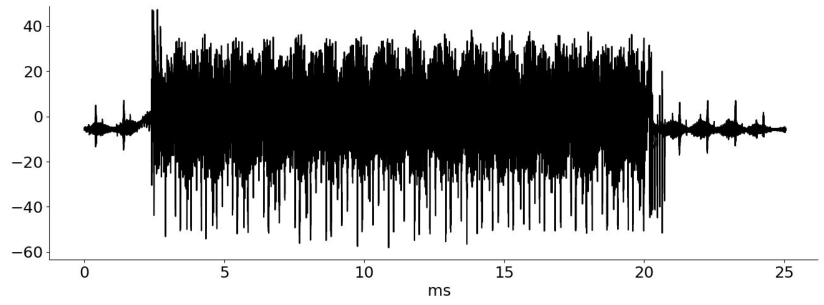


Figure 5.6: Resulting signal, by selecting a carrier close to the clock frequency at 16.253 MHz, and with a 233 kHz bandwidth low pass filter. The signal has a lot of high-frequency components, it is actually caused by the CPU and could contain side-channel information.

5.3.2 Breaking AES using CPA

A good method for breaking \AES with a \power analysis is the so-called correlation power analysis (CPA). Its most advantage in comparison to \differential power analysis (DPA) is, that we can correlate the \power consumption with any arbitrary function. It makes use of the Pearson correlation which looks as equation 5.3 shows at follow:

$$\operatorname{corr}(X,Y) = \frac{n\sum_i x_i y_i - \sum_i x_i \sum_i y_i}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}} \tag{5.3}$$

This formula gives the correct result, only by having to keep on the track of the number of traces and the sum of $x_i, x_i^2, x_i y_i, y_i$ and y_i^2. The \correlation coefficient can then be computed using the equation above.

In the case of \AES, we use the fact that an \XOR operation with an 1 bit [8] uses more power than XOR with a 0 bit . Depending on the challenge, the number of \bit flips is \predicted and directly correlated with the \power consumption. The important part of the \AES is the application of the S-Box in the first (or last) round, as equation 5.4 shown.

[8] Known as bit flip

$$r = r \oplus \operatorname{sbox}_i \left[l \oplus k_i \right] \tag{5.4}$$

l and r are parts of the \plain-text and can be controlled by the \attacker, k_i is a 6-bit subkey for the i-th \S-Box which is a \nonlinear \lookup table. The number of \bit flips depends on a part of the \plain-text and a part of the key. Correlating the hamming weight of the output of the S-Box with the actually measured \power consumption, equation 5.5.

$$\operatorname{corr}\left(s_t, \operatorname{ham}\left(\operatorname{sbox}_i\left[r \oplus k_i\right]\right)\right) \tag{5.5}$$

This correlation is computed for all possible 64 subkeys and every \time step of the \power trace. For the right subkey, we expect a high \correlation with the actual \measured sample-st exactly at the point, where the \S-Box operation happens. The image 5.7 shows the correlation for the correct subkey of the first S-Box (black) and the \mean trace (gray) after 6022 traces. The \spike (high correlation) at the beginning of the AES routine is caused by the XOR operation with the S-Box output.

By comparing the maximum \correlation over time for all possible subkeys, the correct subkey had the highest \correlation after 1500 traces, as figure 5.8 shown. The distance between first and second winner is quite significant and a total \correlation of 0.1 not too bad,

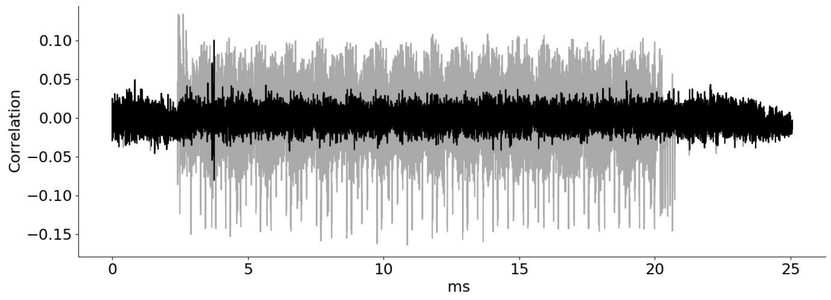


Figure 5.7: Correlation for the correct subkey of the first S-Box (black) and the mean trace (gray) after 6022 traces. The spike (high correlation) at the beginning of the AES routine is caused by the XOR operation with the S-Box output.

especially as the underlying signal had a 70 times lower \bandwidth than the \CPU clock of the \device unit test (DUT).

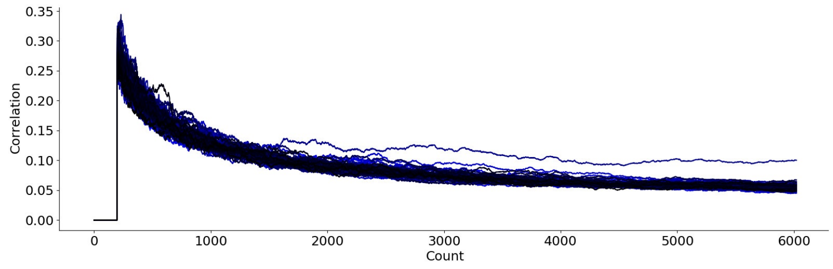


Figure 5.8: Graphic comparing the maximum correlation over time for all possible subkeys, the correct subkey had the highest correlation after 1500 traces.

The MCU had been programmed such that it had 67 76 89 79 88 98 A6 57 65 F7 65 77 5B 87 68 8C as the chiper key. Here it prints the keybytes with the highest correlation coefficients $\rho_{\max_1}$ in descending order. There are 16 columns as there are 16 key-bytes. The first row matches exactly to the key used. For convenience of comparing they are tabulated in Table 5.1, where the column named SBox K_1 shows the set of keybytes which gave the highest correlation. The column $\rho_{\max_1}$ is the correlation coefficients for each of the keybytes in SBox K_1. The SBox K_2 and $r_{\max_2}$ are for the set of keybytes which has the next highest correlation and so on.

As it is shown, the column SBox K_1 matches the exact key used. The correlation coefficients for those keys are quite high where all are more than 0.13. But these correlation coefficients for the key with second highest correlation and the third highest correlation are much lesser where both of them are less than 0.08. Therefore the fact that the correlation coefficients of SBox K_1 and SBox K_2 are having a large gap between them compared to the gap between the correlation coefficients of SBox K_2 and SBox K_3, lets us confidently decides SBox K_1 would give us the correct key. [9]

[9] For getting the correct key not even 1700 traces were necessary.

The table 5.2 shows the ratio gotten with LRA based attack. Showing

Key byte	$SBox$ K_1	$r_{\max_1}$	$SBox$ K_2	$r_{\max_2}$	$SBox$ K_3	$r_{\max_3}$
0	67	0.13794	D0	0.07278	54	0.07261
1	76	0.13790	C6	0.07292	26	0.07272
2	89	0.13826	F9	0.07271	7D	0.07253
3	79	0.13764	49	0.07279	8B	0.07248
4	88	0.13812	F4	0.07258	B4	0.07251
5	98	0.13895	52	0.07269	8E	0.07251
6	A6	0.13783	98	0.07276	12	0.07241
7	57	0.13776	41	0.07262	63	0.07247
8	65	0.13814	95	0.07278	18	0.07251
9	F7	0.13789	30	0.07249	3F	0.07243
10	65	0.13901	10	0.07283	83	0.07249
11	77	0.13796	CF	0.07264	F0	0.07249
12	5B	0.13780	4D	0.07248	65	0.07235
13	87	0.13802	44	0.07256	B2	0.07253
14	68	0.13801	7E	0.07286	4E	0.07278
15	8C	0.13869	A8	0.07271	38	0.07269

Table 5.1: List of correlation ratios, using 5000 traces as reference. The column named SBox K_1 shows the set of keybytes which gave the highest correlation value. The correlation coefficients for those keys are quite high in comparison with the next highest correlations

the LRA rate is not that high than presented before for CPA, showing that the correct guess not always stands out with a good contrast, and even more, in some cases the highest score are not achieved by the correct key byte, leading in false positive detections. but it proves to remain exploitable and thus a robust indicator.

Whereas it has been successfully demonstrated that both DPA and CPA techniques could be viable in deducing the full 16-byte key of AES-128 by monitoring the power consumption of an MCU which implements the AddRoundKey and SubBytes steps in round 1 of AES. However, side-by-side comparison of the results produced by DPA and CPA demonstrate that the `\Hamming weight` Power Model approach may produce key guess results which are easier to interpret from an analytics perspective due to the lack of harmonic noise (`\ghost peaks`) in comparison with the `\difference of means` or the LRA attack techniques. Since results produced here have been gathered in a white-box testing environment, variation in findings may exist when applying these techniques in real-life devices.

The core limitation in this work is how applicable the methodology is when attempts are made to perform attacks on real-life cryptographic devices running AES-128 under a black-box environment. By building an accurate selection function (DPA) or power model (CPA), one could predict the output of the final AddRoundKey step given known plaintext inputs. If prediction of output is correct, then one simply reverses each round of AES until they have the original starting cipher

Key byte	$SBox$ K_1	LRA_{Key}	$SBox$ K_2	$LRA_{\max_{1,2}}$	$SBox$ K_3	$LRA_{\max_3}$
0	67	0.07794	7D	0.07791	D4	0.07689
1	76	0.07590	51	0.07572	2A	0.07522
2	89	0.07826	9F	0.07649	DA	0.07513
3	79	0.07564	94	0.07578	71	0.07495
4	88	0.07812	75	0.07651	AB	0.07587
5	98	0.08095	B8	0.07658	70	0.07599
6	A6	0.07453	69	0.07471	31	0.07468
7	57	0.07376	13	0.07467	12	0.07428
8	65	0.08145	59	0.07642	47	0.07581
9	F7	0.07893	C1	0.07543	AF	0.07498
10	65	0.08018	8F	0.07468	38	0.07431
11	77	0.07567	7C	0.07469	01	0.07449
12	5B	0.07805	B4	0.07546	35	0.07383
13	87	0.07322	AA	0.07583	2E	0.07560
14	68	0.08016	43	0.07788	EB	0.07768
15	8C	0.08191	CA	0.07516	55	0.07515

Table 5.2: List of LRA ratios, using 5000 traces as reference. The column named SBox K_1 shows the set of correct key-bytes which not always gives the highest LRA values, leading in false positive detections (ghost peaks).

key values.

The other more challenging problem is of knowing when to capture power traces in a real-life device running cryptographic operations. Further complexity is introduced since different devices may behave in a distinct manner depending on their hardware and software implementations. As a general suggestion to solve this problem, one should first attempt to identify the actual cryptographic algorithm running on the device. For this case, SPA alone may prove valuable in performing such purpose, if it is identified that communication occurs between a real-life device and a PC under the attackers control during known cryptographic functions, it may be possible to use such responses as a capturing trigger. In the worst case scenario, it may still be possible to attack a device by gathering enough traces and ensuring they are aligned in a coherent manner.

6 Conclusions

The presented work was about the study on the effectiveness and efficiency of the CPA, the LRA and the simple DPA attacks when performed in a context where the exact time of the sensitive computations is not known.

Following a practical approach, the leakage of various selection functions widely used in existing light-weight ciphers for an 8-bit processor was investigated. By analyzing how these results relate to the intuition about side-channel leakages based on the non-linearity of the selection function.

The previous experience on a large set of chips over the last years has convinced on the validity of the Hamming distance model and the advantages of the CPA method against LRA and simple DPA, in terms of efficiency, robustness and number of experiments. However the main drawback of CPA regards the characterization of the leakage model parameters. As it is more demanding than DPA, and the method may seem more difficult to implement.

Using the knowledge gained from the evaluation of selection functions, unprotected software was attacked in implementations of eight well-known light-weight ciphers, namely AES, Fantomas, LBlock, Piccolo, PRINCE, RC5, Simon, and Speck. By grouping the results of the experiments into classes according to the observed resistance against CPA attacks.

The lessons learned from these experiments helped to select the appropriate leakage functions to attack the light-weight block cipher. Thereby, the mainly imperfection of leakage evaluation based on non-linearity, was the XOR bit-wise operation.

It was showed that unprotected implementations of the AES based on the S-box and T-table strategies can be broken even when the attacker controls only one input byte of the cipher with less than 1500 electromagnetic traces acquired from a 8-bit processor in about one hour. Knowledge of the implementation strategy does not significantly improve the attack outcome, nor does it reduce the attack complexity. An important and reassuring conclusion is that all the countermeasures designed against DPA offer the same defensive efficiency against the attack model based LRA or CPA. This is not that surprising since those countermeasures aim at undermining the common prerequisites that three approaches are based on: side-channel observably and intermediate variable predictability.

Finally considering the popularity of the AES chipper algorithm and because the unprotected implementation of AES was broken with the smallest number of power traces, Thus, unprotected implementations of the AES should not be used to secure the communication between end devices in secured network protocols in any case.

Future work will be relied on expanding the previous analysis to 16-bit, 32-bit and 64-bit micro-controller architecture devices. Also considering the expansion of Hamming weight power consumption models to more advanced power models based in new emerging energy theories like Energy behavior in Quantum Field Theory, as possible example.

Bibliography

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Agosto 2006. ISBN 978-0387310732.

J. Daemen and V. Rijmen. *The design of Rijndael: AES the Advanced Encryption Standard.* Springer-verlag edition, 2002.

R. Verdult L. Batina J. Balasch, B. Gierlichs and I. Verbauwhede. *Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. Lecture Notes in Computer Science*, volume 7178. Springer, in o. dunkelman, editor, ct-rsa edition, 2012.

Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. *Mathematics for Machine Learning*. Cambridge University Press, 1st edition, April 2020. ISBN 978-1108455145.

Kjell Johnson Max Kuhn. *Applied Predictive Modeling*. Springer, 1st edition, 2013. ISBN 978-1461468486.

NIST. *Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication (FIPS), 197 edition, 2001.

C. O'Flynn and Z. Chen. *Power analysis attacks against IEEE 802.15.4 nodes. Lecture Notes in Computer Science*, volume 9689. Springer, constructive side-channel analysis and secure design - 7th international workshop edition, April 2016.

J. G. Proakis and D. K. Manolakis. *Digital Signal Processing*. Prentice-Hall, 4th edition, May 2006.

E. Prouff. *DPA Attacks and S-boxes. InFast Software Encryption*. Springer, 4th edition, 2005.

Index