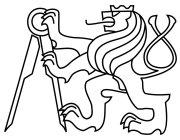
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	Analysis of information content retrieved in a side channel attack
Student:	Marek Pikna
Supervisor:	Ing. Jiří Buček
Study Programme:	Informatics
Study Branch:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2016/17

Instructions

Study the methods of information content measurement in a current (power) or electromagnetic side channel. For a simple microcontroller executing a selected cipher, measure the side channel with a known secret key. Create an estimate of information (entropy) that can be retrieved about the key from the side channel measurements. Verify the estimation in the case of an unknown key. Use differential power analysis (DPA).

References

Will be provided by the supervisor.

L.S.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 3, 2016

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Computer Science

Bachelor's thesis

Analysis of information content retrieved in a side channel attack

Marek Pikna

Supervisor: Ing. Jiří Buček

13th May 2016

Acknowledgements

I would like to thank my supervisor Ing. Jiří Buček for his help. I would also like to thank Brandon McCartney for his unlasting support and positivity.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on 13th May 2016 .....

Czech Technical University in Prague

Faculty of Information Technology

Citation of this thesis

Pikna, Marek. *Analysis of information content retrieved in a side channel attack*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

Abstrakt

Tato bakalářská práce využívá informační metriky známé jako odhadová entropie za účelem analýzy informačního obsahu v postranním kanálu. Útok na postranní kanál, který byl v této práci zvolen, je diferenciální (resp. korelační) odběrová analýza. Tento útok byl proveden na zařízení, které nebylo nijak chráněno proti útokům postranními kanály. Následná analýza ukázala, jak velké množství informačního obsahu je přítomno v měřeních postranního kanálu. Při tuto analýzu se ukázala odhadová entropie jako excelentní bezpečnostní metrika pro určení míry ochrany zařízení proti útokům postranními kanály.

Klíčová slova Diferenciální odběrová analýza, korelační odběrová analýza, AES, entropie, odhadová entropie

Abstract

This bachelor thesis utilizes an information metric known as guessing entropy in order to analyze the information content present in a side channel. The side-channel attack chosen in this thesis is the differential (or more specifically,

correlation) power analysis. Utilizing the correlation power analysis attack on a device lacking protection against it, it is discovered how much information content is present in the power measurements. During this process, guessing entropy shows itself as a highly viable security metric when it comes to deciding how well a device is protected against side-channel attacks.

Keywords Differential power analysis, correlation power analysis, AES, entropy, guessing entropy

Contents

List of Figures

List of Tables

Introduction

The standard perception of an encryption device can be compared to that of a black box - a device which receives an input plaintext and produces nothing but the output - in our case the ciphertext. An attack on such a device is then trying to exploit the encryption algorithm and its weaknesses.

However, in reality, these devices produce other outputs as well, such as time spent on various operations, the electromagnetic leaks or power consumption. This kind of information is called *side-channel information*.

Side-channel attacks are forms of attacks which instead of exploiting a weakness in the algorithm try to exploit the knowledge of side-channel information. While these attacks do not necessarily give the adversary the knowledge of the secret key used in encryption, they are still able to provide him with information that he would not be able to otherwise know, such as the algorithm used for encryption.

The goal of this thesis is to research the potential application of information theoretic metrics in relation to *differential power analysis* (or more specifically, *correlation power analysis*, which is one of the possible side-channel attacks.

Chapter 1

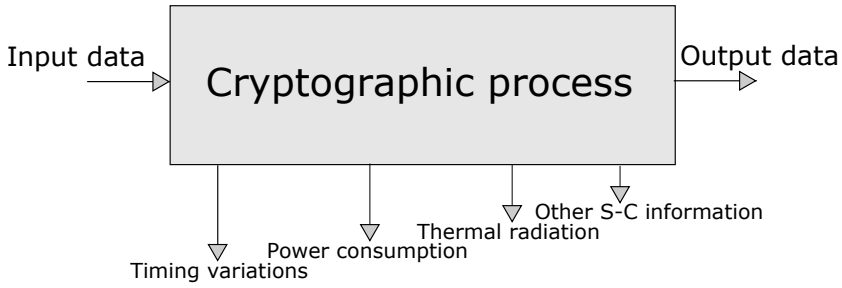
Power analysis

The core component of integrated circuits are the transistors, which are able to work as switches. While the current flows through the circuit, power is being consumed. This power consumption can be externally observed, which is the core idea for side-channel attacks using power analysis.

Figure 1.1: Naive assumption of a cryptographic device

While certain ciphers, such as AES, are considered to be mathematically very strong [1], it is possible to break these ciphers using power analysis attacks. Due to this fact, any cryptographic device which can be directly accessed by the adversary for the purpose of measuring the power consumption needs to be properly secured against these attacks.

Figure 1.2: Actual information available from a cryptographic device



1.1 Simple power analysis

Simple power analysis (SPA) is the starting point of the power analysis attacks. The attack utilizes visual observation of the current used by the device in order to retrieve the secret key [2]. A good example of utilization of this attack can be seen on the RSA cipher. By observing the variations in power usage, specific operations, such as square and multiply in RSA, can be identified. By doing so, the adversary is able to compute the secret key.

Thanks to the fact that SPA utilizes visual observation of the current as it is used by the device, it is also possible to see the order of instructions which had been executed by the device. Thus, it is possible for the adversary to see which conditional branches had been taken. Using that knowledge, the adversary can break the DES cipher as it is visible which segments of the power measurements correspond to the "0" and "1" bits that had been used in the process to decide whether a conditional jump should occur [3].

1.2 Preventing simple power analysis

Simple power analysis depends on the ability of the adversary to visually examine the information leaked from the device. Therefore, simple power analysis can be prevented by introducing noise into the data that the adversary can measure. Another possible option that can be used to prevent SPA is to implement the cryptographic algorithm used in a way that will avoid usage of secret intermediates or key for conditional branches. [3]

Even though SPA is relatively easy to prevent, it has been reported in [4] that many smart cards have been found to be vulnerable to SPA.

1.3 Differential power analysis

Differential power analysis (DPA), instead of visual examination of the information related to power usage leaked from the device, statistically examines the data correlating to the secret key. This statistic examination is exactly what makes DPA so strong, as DPA is able to break even cryptographic devices which are able to produce a lot of noise in their power traces. Not only that, but the adversary also does not need to know anything about the cryptographic device [2]. However, unlike SPA, proper DPA attack requires a significantly higher amount of power traces, which the adversary might potentially not be able to obtain.

The statistical techniques in DPA involve the evaluation of the differences between the means of power traces. The basic steps involved in this process are:

1. Performing cryptographic operations (either encryption or decryption) on a device with different sets of data.

2. Measuring and recording the power consumption traces as well as the data output processed during each cryptographic operation.

3. Partitioning the power traces into different subsets according to a specific property of the state processed.

4. Checking statistical differences between the subsets. A data leak occurs when differences have been observed.

Utilizing these steps, it then becomes possible to retrieve the secret key simply from the power measurements [5].

1.3.1 Correlation power analysis

Correlation power analysis (CPA) is an extension of the differential power analysis. Unlike the standard DPA, it is required to create a model of power consumption in order to approximate the power consumption of the target device during its cryptographic operation. Afterwards, this model of predicted power consumption is applied to a specific key hypothesis. The hypothetical power consumption for a key is then compared with the actual power measurements. The hypothetical power consumption values for a specific key providing the highest correlation with the actual power consumption should then provide the adversary the secret key [5].

When the architecture of the target device is known, it is possible to create a very accurate model of the power consumption. However, even when this information is not known to the adversary, it is possible to use a more general model. Commonly used models are based on the Hamming weight or the Hamming distance [2] [5].

The Hamming weight model (HW) is very basic and works under the assumption that the amount of power consumed during the cryptographic process is proportional to the number of bits with the value '1' during an operation [5]. This power model is generally not well suited to describing the power consumption in CMOS circuits as the power consumption of CMOS circuits is dependent on transitions in the circuit, rather than on the processed value [5]. However, even though that is the case, Hamming weight model can still be used, although the Hamming distance model tends to be preferred when it can be used.

The Hamming distance model (HD) extends the Hamming weight model and uses the number of logic transitions during cryptographic operations to describe the power consumption. The assumptions in this model are:

- A static bit does not contribute to the power consumption.
- A transition from '0' to '1' consumes the same amount of power as the transition from '1' to '0'.

[5]

The Hamming distance model can then be evaluated as

$$HD(v, v') = HW(v \oplus v')$$

where v' is the previous value of v before the cryptographic operation (or operations).

These are not the only relatively generic power models, examples of others worth noting are:

- least significant bit (LSB) model
- zero-value model.

The *least significant bit (LSB)* model simply models the power consumption according to the least significant bit of the hypothetical intermediate values. The *zero-value* model, on the other hand, assumes that the power consumption for data value 0 is lower than for any other value [2].

1.3.1.1 Evaluating correlation

Pearson's correlation coefficient is a very common metric used to evaluate linear relation between data. Because of that, it can be utilized in the DPA attack. Not only that, but the correlation coefficient also helps in comparing different CPA attacks between each other [2].

Formally, Pearson's correlation coefficient is defined as

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}.$$

Where

- $\rho_{X,Y}$ is the correlation coefficient between two random variables
- $cov(X,Y)$ is the covariance between two random variables
- σ_X is the standard deviation of X
- E is the expected value
- μ_X is the mean of X.

The value of the correlation coefficient ranges from -1 to 1, where a value of -1 indicates a completely negative linear relationship, the value of 0 indicates no linear relationship and 1 indicates perfect positive linear relationship.

It is also possible to estimate the value of Pearson's correlation coefficient for a sample. With n measurements of variables X and Y written as x_i and y_i, then the correlation coefficient can be estimated using the formula

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of X and Y and s_x and s_y are the sample standard deviations of X and Y.

It is also worth noting that the Pearson's correlation coefficient is not necessarily guaranteed to be defined. In case that the standard deviation of any variable is zero, the correlation coefficient is simply undefined.

In the process of correlation power analysis, the variables X and Y are used to represent the samples from actual power traces and from the power consumption hypotheses [2] [5].

Even though Pearson's correlation coefficient is commonly used, the attack can also be executed by using another metric to decide the relationship between the hypothetical power consumption values and the actual power traces. Such an example can be the so-called distance-of-means. However, these models only allow binary power models. Therefore, an attack based on correlation coefficient can be stronger since its power model can describe the hypothetical power consumption better than a binary one would [2].

1.4 Preventing differential power analysis

Preventing differential power analysis is however a more difficult task to accomplish than preventing simple power analysis. The main goal of DPA prevention is the reduction of any dependence between the data used by the cryptographic device and the power consumption [6]. There are various techniques which can be used and which accomplish this in slightly different ways. Some of these techniques are:

- Introduction of noise into power measurements [3]
- Reduction of signal sizes [3]
- Hiding [2]
- Masking [2]

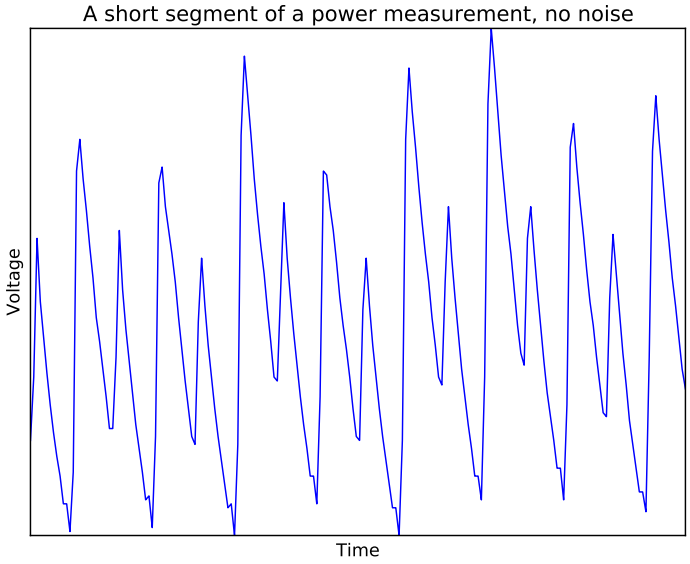
Even though DPA can recover a key when there's a lot of noise in the power traces, introducing noise into power measurements can still be a viable counter-measure. The reason for that is that the number of power consumption samples required for the attack rises up, thus rendering the cryptographic device safer against DPA, especially in situations where the adversary might not be able to record a high amount of traces.

A similar result can be achieved when the signal sizes are reduced. This can be done in various ways, for example by choosing operations that leak less information or by physically shielding the device. However, in practice,

shielding the cryptographic device heavily might not be feasible due to the cost and device's size required. [3]

The effect of adding noise into power measurement samples can be seen on the figure 1.4 and compared to the power measurement samples obtained from an unprotected device presented on the figure 1.3.

Figure 1.3: A short segment of a power measurement sample

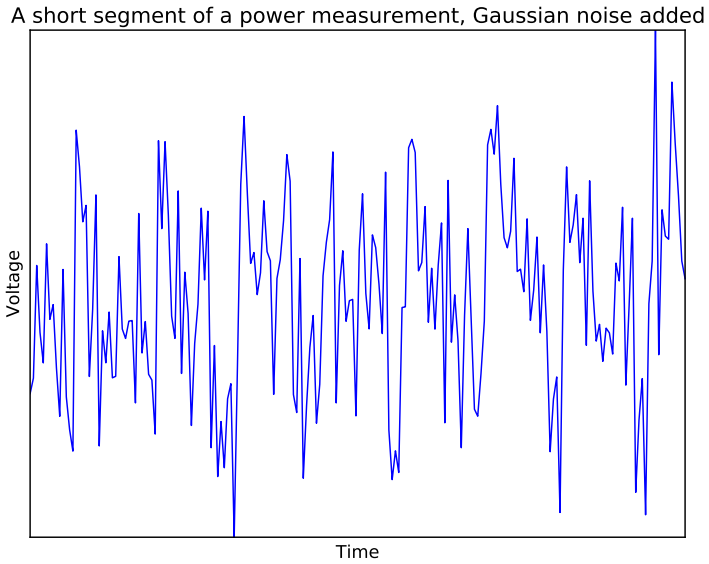


However, even if the device leaks a heavily degraded signal with a lot of noise in it, the adversary can still *theoretically* perform the standard DPA (or CPA) attack, as the only problem is the high amount of samples required. Other two techniques which have been presented, hiding and masking, are able to protect the device from the DPA attack, since their direct goal is to remove the connection between the data processed and the power leakages.

In case of hiding, this goal is achieved by removing the connection between power consumption and the intermediate values or the operations that have been performed. Thus, the cryptographic device using the hiding technique performs all the calculations in the same way as an unprotected device would. [2]. This independence between calculations and the power consumption can be achieved in two different ways [2][6].

- The device consumes random amounts of power in each clock cycle.
- The device consumes equal amounts of power in each clock cycle.

Figure 1.4: A short segment of a power measurement sample, Gaussian noise added



The goal of masking, on the other hand, is to create this independence by concealing intermediate values using a random value, which is called a mask. This mask is generated inside the device every single time – meaning that the value is different every time the cryptographic device runs. Thanks to that, the adversary can not know the mask. Because of all these properties, masking has been extensively discussed by researchers [6].

Even though these defenses protect a cryptographic device from the standard first-order DPA, it is possible to attack devices protected by hiding and even those protected by masking, using so-called higher-order DPA, which exploits the joint leakage of multiple intermediate values [7]. Because of that, combining masking or hiding or both with other protections, such as introducing noise into the measurements, can be very useful, as a DPA attack can become infeasible.

Chapter 2

Measuring quality of DPA

While the adversary is able to break the cryptographic device using differential (or correlation) power analysis, he might not necessarily be able to measure as many traces as he needs to. That can especially be the case when the device is protected by counter-measures, which make the attack less feasible by making the device produce imperfect leakages (either by leaking noise, reducing the signal sizes or using a different strategy). Because of that, an important question arises - how much information does the adversary know about the secret key while he's executing the DPA (or CPA) attack?

At the moment, the metric for measuring the quality of a DPA attack that is used the most (and was also used in DPA contest [8], which is a contest where contestants try to attack a cryptograpghic device using power analysis attacks), is the *success rate*. However, estimating success rate can also be very problematic, as the empirical way involves performing the attack a certain number of times and recording the number of successes. This also poses a problem in deciding security in devices which have been protected against DPA attacks, since the attack might be infeasible for the evaluator, but not necessarily for the attacker [9].

Even though success rate is a very useful metric thanks to its intuitiveness (as a higher rate for lower traces indicates a better attack or a device less protected), the metric itself does not provide any knowledge about the information retrieved about the key so far.

2.1 Guessing entropy

Because of the lack of metrics useful for describing the potential of the attack, another metric, guessing entropy, has recently been used to evaluate the quality of a DPA attack. The second DPA contest [8] also used this specific metric

together with success rate, as a metric for evaluation of the efficiency of an attack (although in a slightly different form of partial guessing entropy).

Guessing entropy was first introduced in [10], but the name was not used at the time. Guessing entropy is a metric which tries to estimate the average number of guesses required to determine the true value of a random variable X, using an optimum strategy. The optimum strategy being one where all the possible values of a random variable X are being ranked from the most to least likely and afterwards guessing the true value in this order.

Formally, if X is a discrete random and the values of X are sorted with decreasing probability, the guessing entropy G(X) is defined as

$$G(X) = \sum_{x=0}^{n} P(x_i)(x+1)$$

where $P(X)$ is the probability mass function. [11]

Further on, partial guessing entropy, as it has been used in the DPA contest, refers to finding the guessing entropy of a specific subkey of the key [12].

2.1.1 Link between Shannon entropy and guessing entropy

Let X be a discrete random variable with possible values $\{x_1, ..., x_n\}$ and $P(X)$ the probability mass function. Shannon entropy is then defined as

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i)$$

[13].

In the article "A Mathematical Theory of Communication", Shannon also explained why the choice of a logarithmic function for information evaluation is ideal. The unit that entropy is measured in depends on the base of logarithm used. In case the logarithm used has base 2, this unit is called a binary digit, or bits for short. If a decimal logarithm is used, the unit is then called a decimal digit [13].

However, guessing entropy, unlike Shannon entropy or other entropy-based metrics, does not use a logarithmic function for its evaluation. Guessing entropy is even measured in a completely different unit - in numbers of guesses

[11]. Because of this a very important question arises - is guessing entropy really linked to Shannon entropy? And if so, how?

It has been proven by Massey in 1994 that the lower bound of the average number of successive guesses using an optimum strategy until the value of a discrete random X is guessed correctly depends on Shannon entropy - and entropy actually provides the lower bound. More specifically, if H(X) represents the Shannon entropy and G(X) the guessing entropy, then

$$G(X) \geq \frac{1}{4} \times 2^{H(X)} + 1$$

provided that H(X) $\geq$ 2 bits [10]. This relationship is also very intuitive, as guessing a value in a system with higher entropy should be more difficult. Massey [10] has also proven that while guessing entropy can be underbounded using entropy, it cannot be overbounded in the same manner.

This link shows that while guessing entropy is measured in different units, an important connection with Shannon entropy has been established.

2.1.2 Guessing entropy as a security indicator

However it is important to note that guessing entropy is not a metric without any problems. The problem mostly stems from the fact that even though a connection between average guesswork and entropy exists, the link does not give us equality between the two.

As it has been presented in *"Guesswork is not a substitute for Entropy"* [14], the lower bound which entropy puts on guesswork is often good enough. However, the expected amount of guesswork and the actual guessability might be completely different.

Not only that, but [11] also mathematically proves that it is possible for a random variable with arbitrarily high guessing entropy which would have a probability of guessing at one try arbitrarily close to 1.

Due to these reasons, it is important to understand that while guessing entropy serves as a reasonable metric for evaluating the expected amount of guesswork when it comes to guessing the value of a secret key, it still does have its own issues. Because of that, it has been proposed in [14] that other means of evaluating guesswork should be used as well, such as calculating the moments of guesswork instead of the mean.

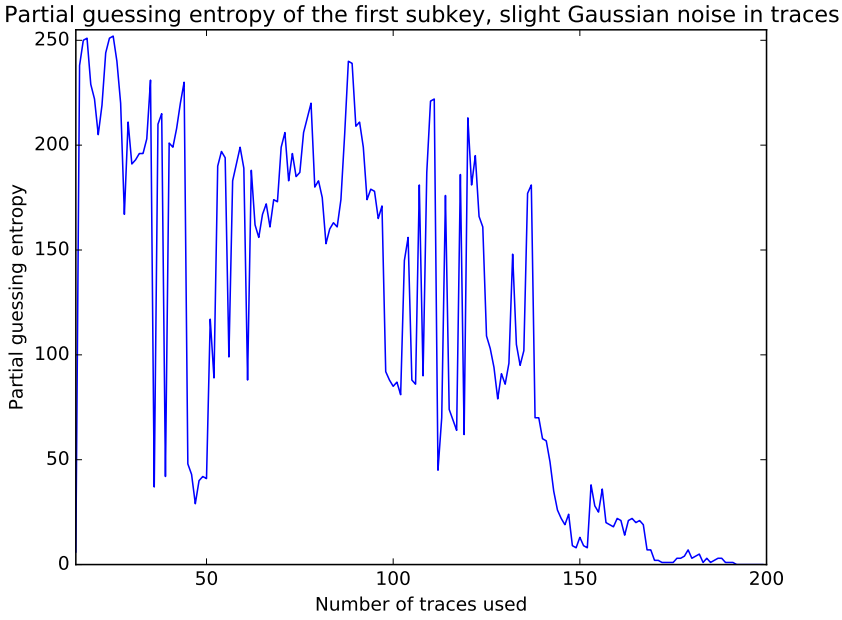
2.1.3 Guessing entropy and CPA

Using guessing entropy as a metric of information, we can estimate how much guessing needs to be done in order to guess the key properly after performing the correlation power analysis attack. The only thing needed for an evaluation of guessing entropy is the ability to sort the hypothetical keys by the order of their probability.

Fortunately enough, this can be easily performed in correlation power analysis. The last step of CPA consists of comparing the hypothetical power consumption values with the power traces. One of the choices commonly used for determining the relationship between these two is using the correlation coefficient. The secret key which the adversary is trying to discover is then going to be the one which correlates the most with the hypothetical consumption values for that secret key [2].

Thanks to this fact, it then becomes possible to sort the hypothetical keys (or subkeys) according to how much they correlate with the real power consumption - thus leading to the hypothetical keys being sorted according to their probability, allowing the evaluation of guessing entropy. The evaluation can simply be done by accumulating the position of the secret key in the ordered list after an arbitrary amount of power traces had been used and then evaluating the average value [15].

Figure 2.1: Example partial guessing entropy progression



As it can be seen on the figure 2.1, partial guessing entropy is a viable metric when it comes to deciding the point where the subkey becomes gets found. Not only that but the effect of added Gaussian noise can be slightly visible as well as an additional trace can completely throw off the correlation analysis at any point, until enough traces are used.

Chapter 3

Advanced Encryption Standard

Advanced Encryption Standard, abbreviated as AES, is a popular symmetric block cipher [16]. The cipher was developed by two Belgian cryptographers, John Daemen and Vincent Rijmen as Rijndael (a play on their names) [17] and was further on announced by National institute of Standards and Technology (NIST) as AES in 2001 [16].

AES is a symmetric block cipher encrypting and decrypting data using a block size of 128 bits (16 bytes) and is capable to do so using a key of 128, 192 or 256 bits. The block consisting of 16 bytes is arranged in a 4 by 4 matrix known as the *State* [16].

Figure 3.1: AES State

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

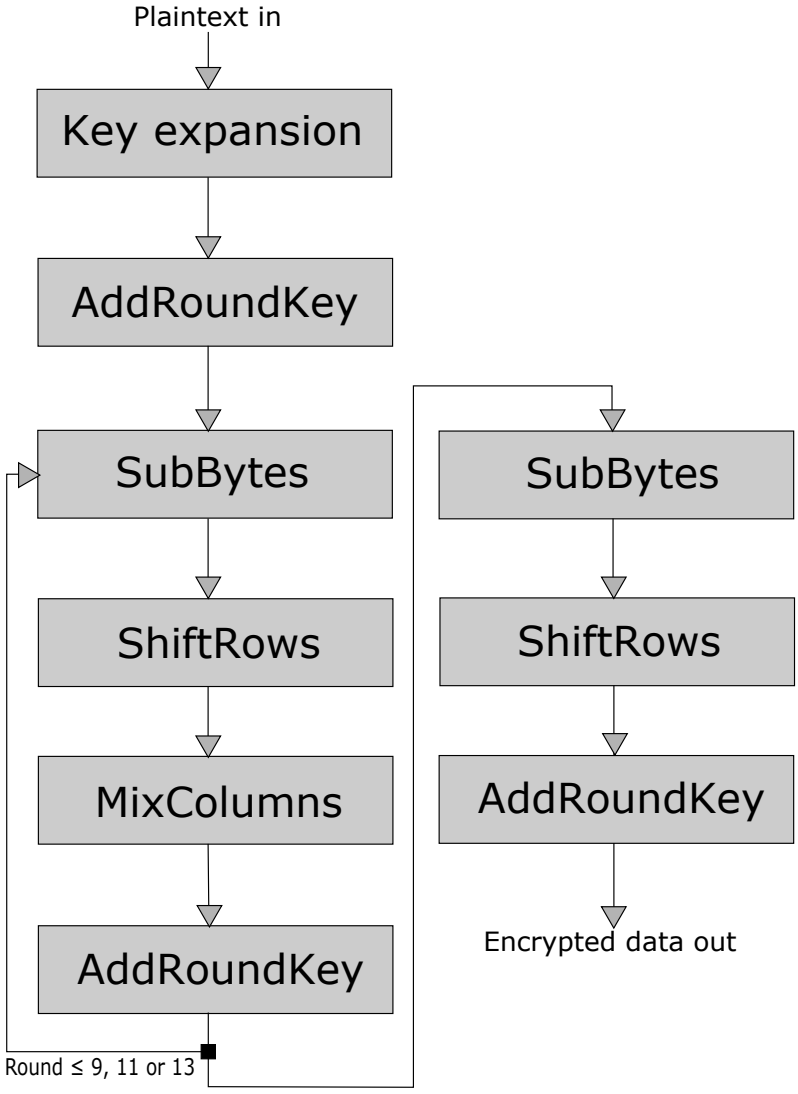
The State, on which all of the transformations are being done, is created from the input array (in) according to the scheme:

$$s_{r,c} = in_{r+4c}$$

for $0 \leq \mathrm{r} < 4$ and $0 \leq \mathrm{c} < 4$ [16].

The number of round operations depend on the size of the key. Either 10, 12 or 14 rounds are done for a 128, 192 or 256 bit key respectively. The order of operations can be seen in the figure 3.2.

Figure 3.2: The flow of AES



The first step before the initial round utilizes *key expansions*. Round keys are derived from the cipher key for each round using a specific key schedule. Afterwards, the 4 main steps done in each round are *AddRoundKey*, *SubBytes*, *ShiftRows* and *MixColumns* (with the exception of the last round).

AddRoundKey is done every round. As a specific round key is generated

for every round from the initial key, the round key is combined with the State using the bitwise XOR operation.

SubBytes is an operation providing non-linearity in the cipher. Each byte is transformed independently using a substitution table (S-box), which is constructed using two transformations

1. The first step consists of calculation of the multiplicative inversion of each byte in $GF(2^8)$ (with the byte {00} being mapped to itself).
2. Afterwards, an affine transformation is applied. A fixed 8x8 bit matrix is multiplied with the calculated multiplicative inverse (in $GF(2)$). Afterwards, the result is masked using the bitwise XOR operation with the byte {63}.

ShiftRows operates on rows of the AES State. Each row except for the first one is cyclically shifted by a specific offset.

- The second row is shifted by one position to the left.
- The third row is shifted by two positions to the left.
- The fourth row is shifted by three positions to the left.

MixColumns provides diffusion in the cipher by multiplying each column of the state with a fixed matrix.

3.1 Side-channel attacks and AES

AES is considered to be one of the top ciphers, even to the point that the U.S. government announced that the cipher can be used to protect classified information up to the SECRET level (TOP SECRET level requires use of AES using either 192 or 256 bit key lengths) [18].

Currently, the best key-recovery attack on AES is able to recover the key with time complexity $2^{126.01}$ (AES-128), $2^{189.91}$ (AES-192) and $2^{254.27}$ (AES-256) [1], which is a very slight improvement over the bruteforce method and is still not feasible in realistic scenarios.

However, side-channel attacks are not trying to exploit the cipher directly, but rather uses information leaking from the device through the side-channels. And while AES has been shown to be strong against any kind of traditional attack, differential power analysis is potentially a feasible attack on AES [5].

3.1.1 Differential power analysis attack on AES

During the DPA contest v2 [8], where participants were executing a DPA attack on an FPGA implementation of AES which was unprotected. Using various kinds of differential power analysis, certain contestants were able to achieve global success rate higher than 80% after analyzing less than 10,000 power traces. Even though obtaining such an amount of traces might not be possible in certain scenarios, it still shows how feasible DPA and its variations can be against AES, especially when compared to other potential kinds of attacks. As it has been reported in [5], the number of traces required for the DPA attack to extract a 128-bit key tends to be around thousands.

When attacking the AES cipher using DPA, several properties of the cipher can be exploited [5].

The first property which can be exploited is right at the beginning of AES. The initial plaintext input is masked with the round key using the XOR operation in the *AddRoundKey* operation. This method, however, requires the adversary to have access to the plaintext values used.

Another exploitable property happens right at the end of AES. The final ciphered output is the result of operations *SubBytes* and *ShiftRows* XOR-ed with the final round key. An advantage at this point is that last round does not involve any MixColumns diffusion. While the relationship between the state (especially at the end of the cipher) and the key is not as clear as in the previous case, it is possible to compute the initial key from the final round key by reversing the key expansion.

A property which can be effective to exploit is the substitution in the *SubBytes* operation of AES. This has a specific advantage when it comes to the magnitude of the attack space. The *SubBytes* operation operates on each byte of the AES State independently. Because of that, the same set of traces can be used for all bytes, which significantly reduces the possibilities the adversary needs to count with. In case of AES-128, instead of searching all the 2^{128} possible values of the key, the attack space only has 16×2^8 combinations.

3.1.2 Correlation power analysis attack on AES

The first important step when attacking the AES cipher using correlation power analysis is to select a particular data register for the attack. When it comes to the AES algorithm, a relationship between the data and power consumption can be found well in the *SubBytes* operation. It is possible to target the AES S-Box used in the first round as well as in the last round. Afterwards, the actual execution of the attack can be done in a few simple

steps [2].

Firstly, the power consumption is recorded. In order to perform these measurements of power traces, the adversary records the data inputs and their corresponding power consumption measurements. Combining these, it is possible to create a matrix $\mathbf{T}$ of size $D \times T$, where D is the amount of input plaintext values which have been used and T denotes the length of the traces.

The second step involves preparation for the power analysis. Input plaintext values are combined with all key hypotheses using the specific cryptographic algorithm used in the spot which the adversary is attacking. In case of attacking the first round of *SubBytes* operation, this can be done by evaluating the intermediate values of *AddRoundKey* and *SubBytes* operations. This creates a matrix $\mathbf{V}$ of size $D \times K$, where K denotes the total number of possible values which the key can have. A column j of the matrix $\mathbf{V}$ then has all the *hypothetical intermediate values* for a key k_j.

Afterwards, the chosen power model is applied to the hypothetical intermediate values. Each element $v_{i,j}$ of matrix $\mathbf{V}$ is mapped to a new element $h_{i,j}$ in the matrix $\mathbf{H}$ containing *hypothetical power consumption values.*

The last step involves comparing the hypothetical power consumption values with the actual power traces. As it has been described in chapter 1, correlation coefficient is an excellent choice for this. Every column h_j of matrix $\mathbf{H}$ is compared with each column t_i of matrix $\mathbf{T}$. The attacker therefore compares the hypothetical power consumption values of all key hypotheses with the recorded traces at every position. This results in the matrix $\mathbf{R}$ of size $K \times T$, where an element $r_{i,j}$ is the result of the comparison between columns h_i and t_j.

The matrix R reveals two important pieces of information at the indices with the highest value:

- The secret key used by the cryptographic device.
- The moment when the intermediate value is processed by the device.

[2]

Figure 3.3: Visualization of correlation values in matrix R for a correct subkey (*HD* model)

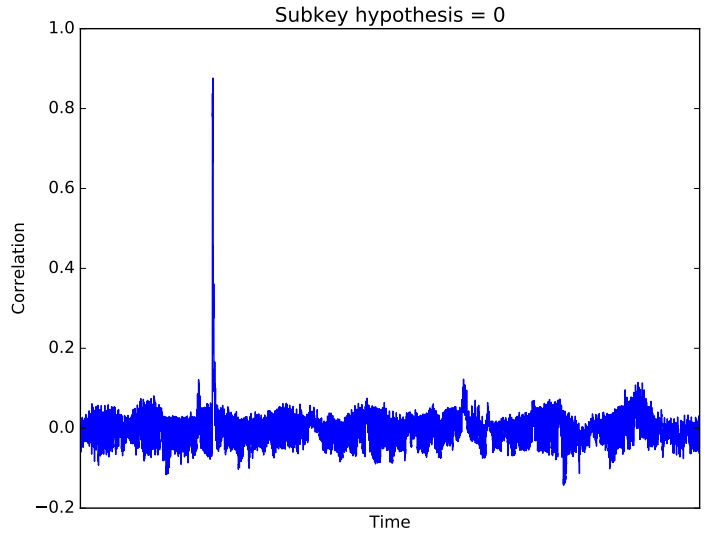
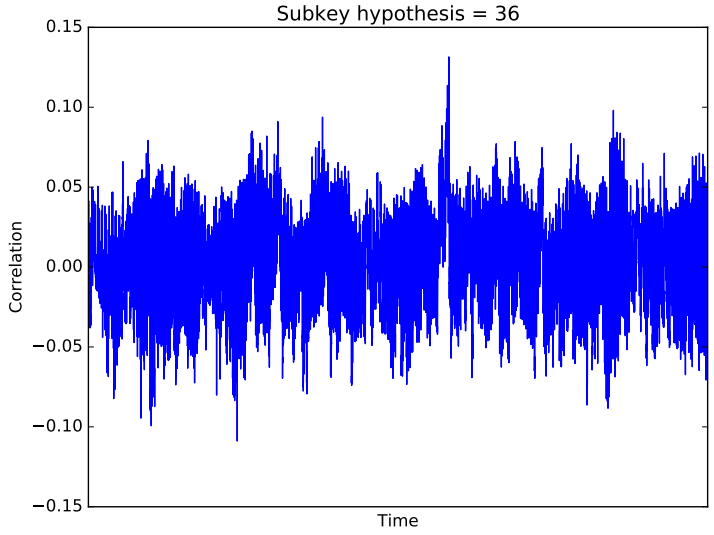


Figure 3.4: Visualization of correlation values in matrix R for an incorrect subkey (*HD* model)



As it can be seen in figure 3.3, when the subkey is found, a significant peak

can be observed in the corresponding row of the matrix **R**. The other rows on the other hand lack this significant peak and their values tend to be relatively close to each other.

Chapter 4

Experiments

The goal of this chapter is to empirically decide whether guessing entropy is a helpful metric for providing information about side-channel leakages. That is being decided by attacking a device using the CPA attack in various situations, those being:

- Unprotected device
- Device producing noise into its power samples
- Device reducing signal quality - cutting bits of information

It is important to note that the device from which the power measurements were taken did not possess any protection against CPA - both noise and removal of information in the power traces were simulated. Also, when evaluating the partial guessing entropy of the key, only one key had been used. In order to see whether the results can be expected consistently, all the power measurements were divided into smaller population sets.

4.1 Measuring the power traces

The traces were measured using the *Agilent DSO-X3012A* oscilloscope. The device performing the AES cipher was the smartcard *ATMega Card* using the *ATMega163* circuit. Used card reader was *Gemalto IDBridge CT30*. Using this method, 4000 power traces were taken.

For the CPA attacks on a device protected by noise introduction, additional Gaussian noise has been applied to the power traces by drawing random samples from normal (Gaussian) distribution.

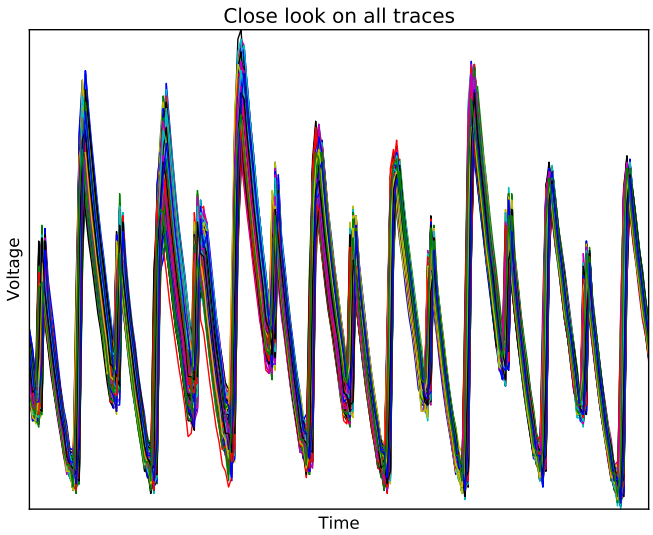
For the CPA attacks on a device protected by changing the amount of information retrieved, this had been simulated by applying a mask to the power measurements result using the bitwise AND operation. As every measurement at a point was represented by one byte, in order to remove one bit of information, a mask $\{11111110_2\}$ (254) was applied to each measurement using the AND operation. This was done similarly for other bits as well.

Because the CPA attack involves comparing the hypothetical power consumption values with the power measurement traces at any time, it is important that the power traces are aligned. Misaligned power traces can occur in case of the device being protected by e.g. dummy operations or even in temperature variations which can slightly alter the clock speed. [19]

4.1.1 Trace alignment

In order to execute the attack, it was necessary to see whether the measured power traces were aligned properly. A visualization with an overlay of all power traces during a short period of time had to be created. If the peaks in power measurements happen around the same time, the traces are aligned and no further operations are necessary. If the traces are somehow misaligned, it is necessary to align the traces properly before executing the CPA attack [19].

Figure 4.1: Overlay of 200 power traces in a small time segment



In case of misalignment of traces, several strategies exist. Two similar strategies have been tested in [19]. The first one being aligning the traces

according to the global maximum and assume its position is characteristic for all the traces. However, this strategy had failed to work as the global maximums were too far apart and in the end it was not even possible to break the device using DPA. Therefore, another strategy had to be used. Instead of using a global maximum, the traces were aligned according to the local maximum of the area of traces which had been targeted in the attack. This strategy had shown to be effective enough to provide a performance boost in most trace sets.

Even more complex techniques exist for alignment of traces, such as the technique of so-called *elastic alignment*. This strategy uses *FastDTW* - dynamic time warping to align a set of power traces. Dynamic time warping is used in speech recognition as human speech does not say each word after the other with the same length of a pause. This strategy had shown to be very effective and increase the success rate heavily [20].

In this case, since the device was not protected in any way, the traces were aligned properly and no specific operation with them was required.

4.2 Executing the attack

The correlation power analysis attack has been executed in the *Python* programming language, using the *NumPy* mathematical library for optimizing the attack. The choice of language was mostly due to high readability of the language for the reader as well as the fact that other languages suited for mathematic calculations such as *MATLAB* or *Wolfram Mathematica* are not as commonly available as *Python* is.

The implementation of the CPA attack on AES-128 is as it has been described in chapter 3 - no adjustment to any defenses were implemented. The implementation can be found in the accompanied file *AES_CPA.py*. In order to improve the execution time of the attack, attacks on each subkey run in parallel using the *joblib* library.

At first, the chosen power model for describing the power consumption was the *Hamming distance* model. However, it was not possible to retrieve the secret key completely. Even though the first 15 subkeys were perfectly retrievable, the last one was not. Because the assembly source code of the AES implementation did not show any difference between the SubBytes operation performed for any of the first 15 subkeys and the last one, it was not believed that the software implementation causes this unexpected behavior. Thus, the entire power traces and the correlation points were analyzed. Unexpectedly, the *Hamming distance* model correlated at other points in time which were

expected. Instead of correlating at the point when the device stored the value retrieved from the *S-Box*, the model correlated after the S-Box operation in each cycle ended - thus correlating at the load operation of the next subkey. The reason for this is not known and it shows that the *HD* model does not necessarily describe any hardware perfectly.

Therefore, a different power model had to be used. Instead of the *HD* model, the *Hamming weight* model had been used, which was able to break all the subkeys perfectly. Not only that, but when the correlation point was analyzed, the correlation happened right during the operations where it was expected - right after the value was retrieved from the *S-Box*.

4.2.1 Evaluating partial guessing entropy

As guessing entropy is an average number of guesses required using optimum strategy to guess the correct key, the complete set of traces has been divided into a set of smaller trace sets of the same sizes.

In order to evaluate partial guessing entropy for each subkey and for every possible amount of traces from the measured trace set, it was then necessary to save partial guessing entropy from each run.

This data was saved into a file in the format of a matrix - each row contained partial guessing entropy for each subkey at a specific number of traces used in the CPA attack. Therefore the matrix was of size $T \times 16$, where T was the amount of traces used. Each element of these matrices was then averaged with each other in order to evaluate the final value of partial guessing entropy.

4.3 Results - unprotected device

The 4000 power traces were divided into 10 sets of 400 traces. The secret key was retrieved from each trace set with a relatively low number of traces needed. The complete amount of traces needed for partial guessing entropy of all subkeys to be consistently 0 (meaning that no change in any partial guessing entropy for the trace set has been observed) can be seen on the table 4.1.

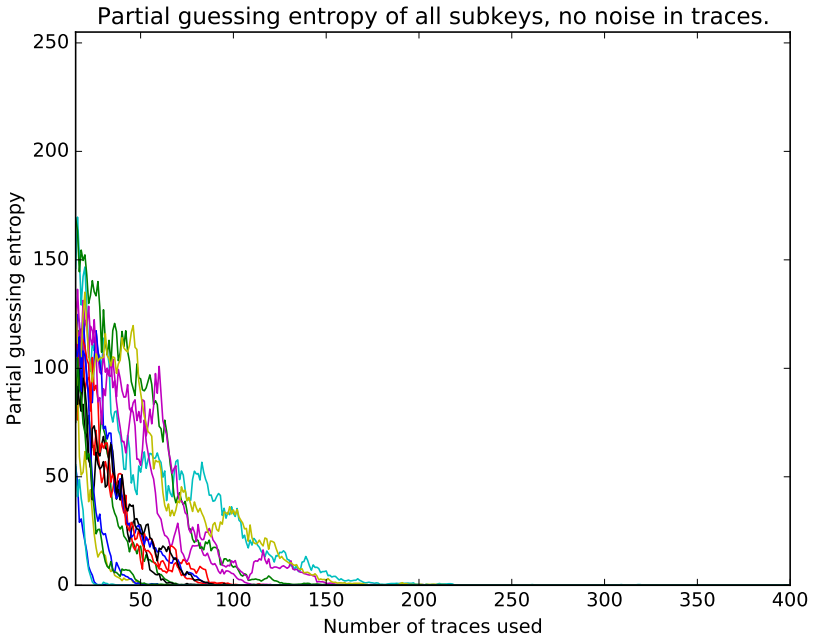
It is important to note that the values in the table 4.1 refer to *consistent* zero guessing entropy. In most cases, the secret key was already known - but addition of newer traces made it possible for a certain subkey to correlate slightly more than the correct one, resulting in partial guessing entropy of the subkey to be 1 instead of 0. This can be seen on the figure 4.2, which shows how well was the key known. Even though it was necessary to use over 200 traces to be absolutely sure that the secret key was recovered properly, this

Table 4.1: Amount of traces required to know the secret key perfectly - unprotected device

Trace set #	**Traces needed**
Trace set 1	228
Trace set 2	203
Trace set 3	331
Trace set 4	202
Trace set 5	214
Trace set 6	310
Trace set 7	208
Trace set 8	216
Trace set 9	351
Trace set 10	212

was the case only for a specific subkey - all the others were perfectly known. Therefore, even if the attacker could not get the amount of traces required, using a bruteforce method at this moment would be extremely fast.

Figure 4.2: Partial guessing entropy averaged from 10 tracesets taken from an unprotected device.

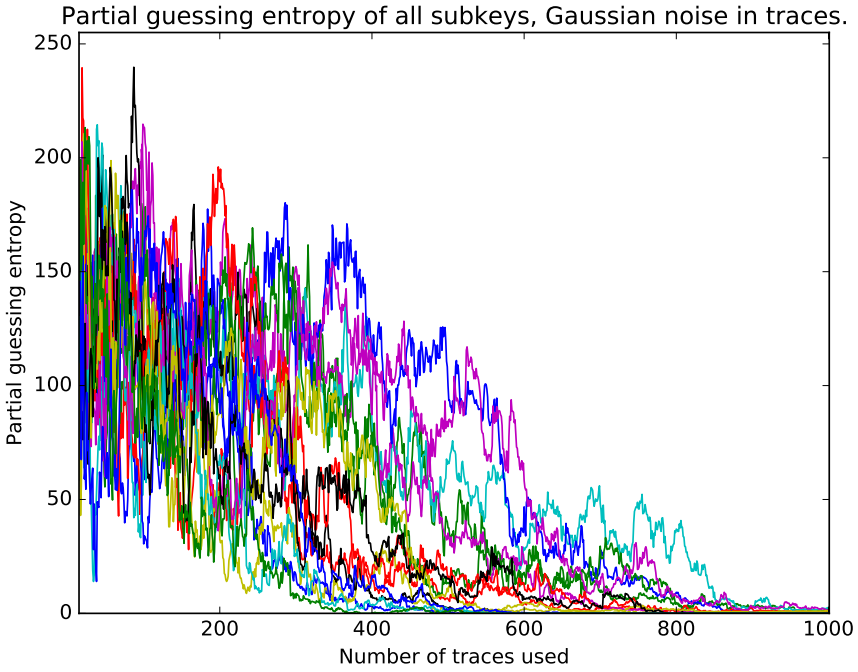


The amount of traces which were needed to break the device (mostly between 200 and 300) is lower than expected (for example the DPA contest v2 [8] provides 20,000 traces for their unprotected device, yet certain attacks were not able to recover the secret key). This is mostly attributed to the high quality of power measurements and the naive implementation of AES which leaks information heavily.

4.4 Results - noise in measurements

The 4000 power traces were divided into 4 sets of 1000 traces. None of the trace sets allowed for a complete, stable retrieval of the secret key. However, most of the subkeys were retrieved and those that were not had a very low partial guessing entropy (below 5).

Figure 4.3: Partial guessing entropy averaged from 4 trace sets with Gaussian noise.



Even though the figure 4.3 is quite erratic, a few important points can be deduced from the figure:

- Added noise changes certainty of the subkey being the correct one heav-

ily. A few traces which behave differently from the rest can make the correct subkey become more improbable than before.

- Certain subkeys leak the information better than others - even through Gaussian noise. While certain subkeys required a high amount of traces before their partial guessing entropy became lower than 10, other subkeys leaked the information better so it was possible to retrieve them with a lower amount of traces.

When analyzing the certainty of the key, guessing entropy shows how viable metric it actually is. Not only can it be seen how much can the adversary be potentially certain of the key after attacking on a certain amount of traces, it even shows its viability as a security metric - the effect of the defensive measure can be visually seen and more defenses can be applied, if need be.

4.5 Results - cutting bits of information

Due to the relatively high amount of masks applied, only 4 sets of 400 power traces were used in these experiments. These trace sets were the same as the first four sets used in the experiments with an unprotected device. The purpose of these masks was to remove certain bits from the power measurements. In each experiment, an additional bit was removed from the power traces, starting at removing the least significant bit and ending at keeping only the two most significant bits. Using this method, information content was taken out of the power measurements.

Firstly, two hypotheses have been made before the experiments were performed:

- The lowest bits don't necessarily provide any information and can differ simply based on small temperature differences or other reasons. Thus, their removal could potentially even *lower* the entropy (or the guessing entropy).

- When enough information content is removed, the CPA attack should be difficult to perform, as the lower bits should be the ones providing the small differences which help the execution of the attack. Since these differences are trying to be discovered using statistical methods (correlation analysis, in this case), with only the highest bits present, the attack should become difficult to perform.

An interesting thing to note is that when enough information content was removed, the attack itself required more traces in general, as at certain times the correlation could not be evaluated (division by zero).

The effect of removing information from the power measurements can be seen on the figure 4.4, which shows how much the signal quality degrades each time a bit of information is removed from the power measurements.

4.5.1 Lowest bit removed from the measurements

The expected behaviour in this case was that the guessing entropy should be roughly the same as in the case of the unprotected device (potentially even better, if the least significant bit was mostly just noise in measurements).

As in the case of the unprotected device, the secret key was retrieved perfectly. The amount of traces required for a stable retrieval can be seen on the table 4.2.

Table 4.2: Amount of traces required to know the secret key perfectly - LSB always 0

Trace set #	**Traces needed**
Trace set 1	233
Trace set 2	211
Trace set 3	344
Trace set 4	158

When compared to the table 4.1, the hypothesis is proven mostly correct. The amount of traces which were required to retrieve the secret key perfectly were roughly the same. Not only that, but in the case of the fourth set, a lower number of traces was required.

4.5.2 Two lowest bits removed from the measurements

The estimated values of guessing entropy were still to be roughly the same as in the case of the unprotected device. The reason for this hypothesis is that the two least significant bits were not expected to carry a lot of information.

Just as before, the secret key had been retrieved perfectly in a stable manner in all sets of traces. The amount of traces required for a stable retrieval of the key can be seen on the table 4.3.

Just as in the case before, the expected behaviour was correct. In certain cases, the removal of information content even helped in retrieval of the secret key.

Table 4.3: Amount of traces required to know the secret key perfectly - 2 LSBs always 0

Trace set #	**Traces needed**
Trace set 1	203
Trace set 2	211
Trace set 3	342
Trace set 4	193

4.5.3 Three lowest bits removed from the measurements

From visual observation of the effect of removing three lowest bits from the measurements 4.4, no real change in the difficulty of executing the attack was expected due to the fact that the signal still wasn't degraded heavily.

Table 4.4: Amount of traces required to know the secret key perfectly - 3 LSBs always 0

Trace set #	**Traces needed**
Trace set 1	266
Trace set 2	253
Trace set 3	214
Trace set 4	180

This hypothesis had proven mostly correct. In two sets, the attack became more difficult to perform. However, in the other two, it became easier - especially in the case of the third trace set.

Still, as the three lowest significant bits did not degrade the signal very much, retrieving the secret key required almost as much work as in the case of the unprotected device.

4.5.4 Four lowest bits removed from the measurements

At this point, visual observation of the signal on figure 4.4 shows that the signal begins to degrade in quite an obvious manner. Therefore, the expected behaviour in this case was to see all tracesets requiring more traces in order to retrieve the secret key.

While the secret key had been retrieved perfectly in a stable manner using all of the trace sets, the required work started to become visibly higher. Not only that, but all four sets required more work than in any of the cases before.

Table 4.5: Amount of traces required to know the secret key perfectly - 4 LSBs always 0

Trace set #	Traces needed
Trace set 1	378
Trace set 2	281
Trace set 3	356
Trace set 4	246

4.5.5 Five lowest bits removed from the measurements

Because the signal quality has degraded even heavier, it was expected that it might not be possible to retrieve the secret key in a stable manner in certain trace sets.

Table 4.6: Amount of traces required to know the secret key perfectly - 5 LSBs always 0

Trace set #	Traces needed
Trace set 1	392
Trace set 2	308
Trace set 3	N/A
Trace set 4	338

In the case of the third trace set, it was not possible to retrieve the secret key completely as the partial guessing entropy of one subkey did not become 0. Still, even though the work required to retrieve the key in a stable manner was higher, lower amount of traces was required to know the secret key almost perfectly, as in all cases most of the subkeys were retrieved earlier. Because of that, even in the case of the third traceset where the secret key was not retrieved, the guessing entropy was so low that the amount of guesswork required would be extremely small.

4.5.6 Six lowest bits removed from the measurements

At this point, the signal became so degraded that it was expected that more trace sets would not allow for a perfect stable retrieval of the secret key.

The CPA attack became harder to perform yet again - however, still, potentially reasonable to perform. Due to these results, it is visible that the most information required for the CPA attack is present in the most significant bits.

Table 4.7: Amount of traces required to know the secret key perfectly - 6 LSBs always 0

Trace set #	**Traces needed**
Trace set 1	N/A
Trace set 2	324
Trace set 3	N/A
Trace set 4	373

4.5.7 Only the most significant bit remaining

An interesting question arose - since the most significant bits had shown themselves empirically to be the most important ones for the CPA attack, would it be possible to retrieve the secret key using only the most significant bit?

Table 4.8: Amount of traces required to know the secret key perfectly - only MSB used

Trace set #	**Traces needed**
Trace set 1	361
Trace set 2	369
Trace set 3	N/A
Trace set 4	361

Surprisingly enough, it was possible to perform the attack only with the knowledge of the most significant bit - thus only having the values 0 and 128 retrieved from the power measurements.

As it can be seen on the figure 4.5, most subkeys were successfully retrieved in the attack. Compared to the figure 4.2, it can be seen exactly how much more was the attack difficult to perform - and the fact that the difference is not that high. Due to these reasons it is expected for these measurements that the bits containing the most information required for the attack are the most significant ones.

4.5.8 Empirically checking the results

Cutting off bit after bit starting at the least significant bits showed exactly which bits were the most important in performing the attack - especially using guessing entropy as a metric to determine how good the attack is and how much information content is present in the power traces.

In order to test whether these theorems were correct, the attack was performed on 400 measurements with only the three least significant bits in the power measurements used.

In this case, only 5 subkeys were retrieved properly, with the rest having their partial guessing entropy either relatively low (between 1 and 20) or relatively high (from 79 to 249). Even though the attack space became smaller, its size is still enormous for a brute force attack.

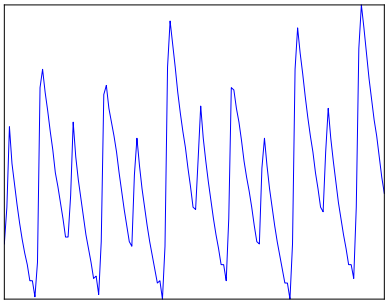
4.6 Summary of experiments

Guessing entropy had empirically proven itself to be a good security metric when it comes to deciding how much work the adversary needs to put into breaking the device using a side-channel attack. Not only that, but a few points about the CPA attack on an unprotected device and guessing entropy can also be made:

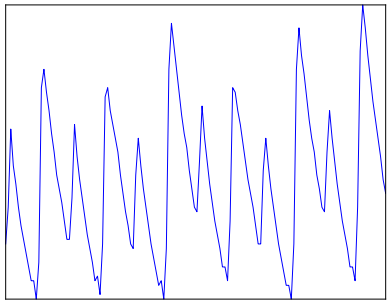
- The values of guessing entropy on various tracesets in one experiment were very close to each other. As this had been tested on multiple tracesets, an estimate of guessing entropy which came from one trace set can be enough to determine the guesswork required.
- Progression of guessing entropy can show how well can the secret key be retrieved - and more specifically, the effect of a defensive measure on the difficulty of the attack.
- Guessing entropy at a given time also leads to information about whether the attack space needed for a brute force attack becomes smaller - and if so, how much. In most experiments, even if the amount of required traces to perform the attack was relatively high, the guessing entropy was very low after a significantly lower amount of traces. In that case, even a brute force attack would potentially be viable.

An important thing to note is that these findings came from an unprotected device with protections being simulated. However, it is believed that these simulations were close enough to the potential real behaviour of the devices.

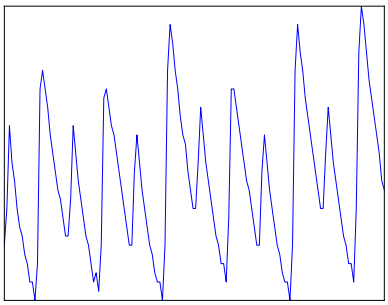
When it came to the experiments utilizing the removal of information content in the power measurements, an interesting point had presented itself - the CPA attack became progressively harder, but still viable, with it mostly dependent on the values of the more significant bits, rather than the small differences between each values. This had been proven by the fact that the attack was not not possible to perform with utilizing only the least significant bits.



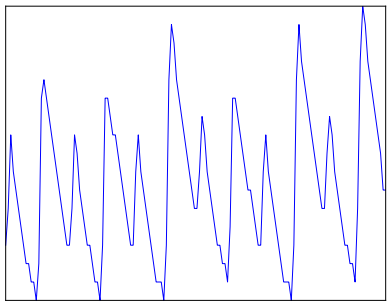
(a) No bits removed



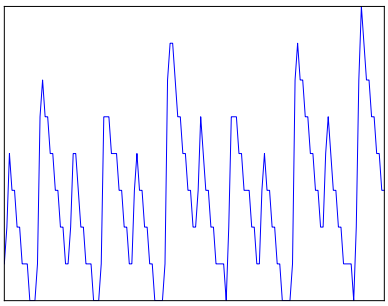
(b) Mask {1111 1110_2} applied



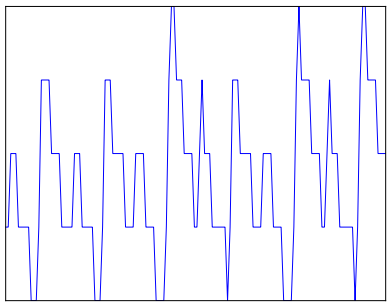
(c) Mask {1111 1100_2} applied



(d) Mask {1111 1000_2} applied



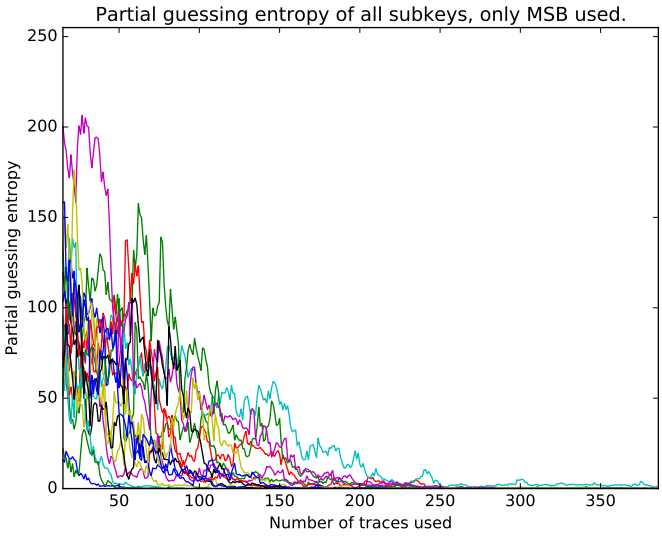
(e) Mask {1111 0000_2} applied



(f) Mask {1110 0000_2} applied

Figure 4.4: The effect of removing information content from the power measurements

Figure 4.5: Partial guessing entropy averaged from 4 trace sets using only the most significant bit.



Conclusion

The side-channel attack called correlation power analysis attack on the AES algorithm had been researched and implemented in the *Python* language. Utilizing the metric called *guessing entropy*, it became possible to determine the information content present in the power measurements.

The metric itself had proven itself to be highly viable when it came to empirically deciding how well the device is protected against the attack - as well as showing how difficult (or easy) it is for the adversary to perform a brute force attack after performing the correlation power analysis attack on devices protected against side channel attacks.

The analysis of guessing entropy in attacks where information content was removed from the side channel also showed which bits provided the most information required for the correlation power analysis attack.

Bibliography

[1] Tao, B.; Wu, H. Improving the biclique cryptanalysis of AES. In *Information Security and Privacy*, Springer, 2015.

[2] Mangard, S.; Oswald, E.; Popp, T. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[3] Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Advances in Cryptology—CRYPTO'99*, Springer, 1999, pp. 388–397.

[4] Kocher, P.; Jaffe, J.; Jun, B.; et al. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, volume 1, no. 1, 2011: pp. 5–27.

[5] Meritt, K. Differential Power Analysis attacks on AES. 2012. Available from: `https://people.rit.edu/kjm5923/DPA_attacks_on_AES.pdf`

[6] Ptáček, L. Power analysis of AES. 2008. Available from: `http://unida.cz/th/143199/fi_b/bc_thesis_orig.pdf`

[7] Oswald, E. Power analysis attacks. Available from: `http://www.cs.bris.ac.uk/Research/Seminars/departmental/2007-03-29_DeptSeminar_Elisabeth_Oswald.pdf`

[8] Duc, G. DPA Contest. Available from: `http://www.dpacontest.org/`

[9] Lomné, V.; Prouff, E.; Rivain, M.; et al. How to estimate the success rate of higher-order side-channel attacks. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, Springer, 2014, pp. 35–54.

[10] Massey, J. L. Guessing and entropy. In *Information Theory, 1994. Proceedings., 1994 IEEE International Symposium on*, IEEE, 1994, p. 204.

[11] Cederlöf, J. Authentication in quantum key growing. 2005.

[12] O'Flynn, C.; Chen, Z. D. Side channel power analysis of an AES-256 bootloader. In *Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on*, IEEE, 2015, pp. 750–755.

[13] Shannon, C. E. A Mathematical Theory of Communication. 1948.

[14] Malone, D.; Sullivan, W. Guesswork is not a substitute for Entropy. In *Irish Information Technology and Telecommunication conference, IT&T 2005*, 2005, pp. 1–5. Available from: `http://eprints.maynoothuniversity.ie/6302/`

[15] Köpf, B.; Basin, D. An information-theoretic model for adaptive side-channel attacks. In *Proceedings of the 14th ACM conference on Computer and communications security*, ACM, 2007, pp. 286–296.

[16] NIST. Advanced encryption standard (AES). *Federal Information Processing Standards Publication (FIPS)*, volume 197, 2001.

[17] Schwartz, J. TECHNOLOGY; U.S. Selects a New Encryption Technique. 2000. Available from: `http://www.nytimes.com/2000/10/03/business/technology-us-selects-a-new-encryption-technique.html`

[18] CNSS. National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information. 2003. Available from: `http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf`

[19] Kevorkian, C.; Tanenbaum, J. Advanced Cryptographic Power Analysis. Available from: `https://www.wpi.edu/Pubs/E-project/Available/E-project-122211-215512/unrestricted/Power_analysis.pdf`

[20] van Woudenberg, J. G.; Witteman, M. F.; Bakker, B. Improving differential power analysis by elastic alignment. In *Topics in Cryptology–CT-RSA 2011*, Springer, 2011, pp. 104–119.

Appendix A

Acronyms

SCA Side-channel attack

SPA Simple power analysis

DPA Differential power analysis

CPA Correlation power analysis

HW Hamming weight

HD Hamming distance

AES Advanced Encryption Standard

NIST National Institute of Standards and Technology

GF Galois field

LSB Least significant bit

MSB Most significant bit

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `cpa` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format