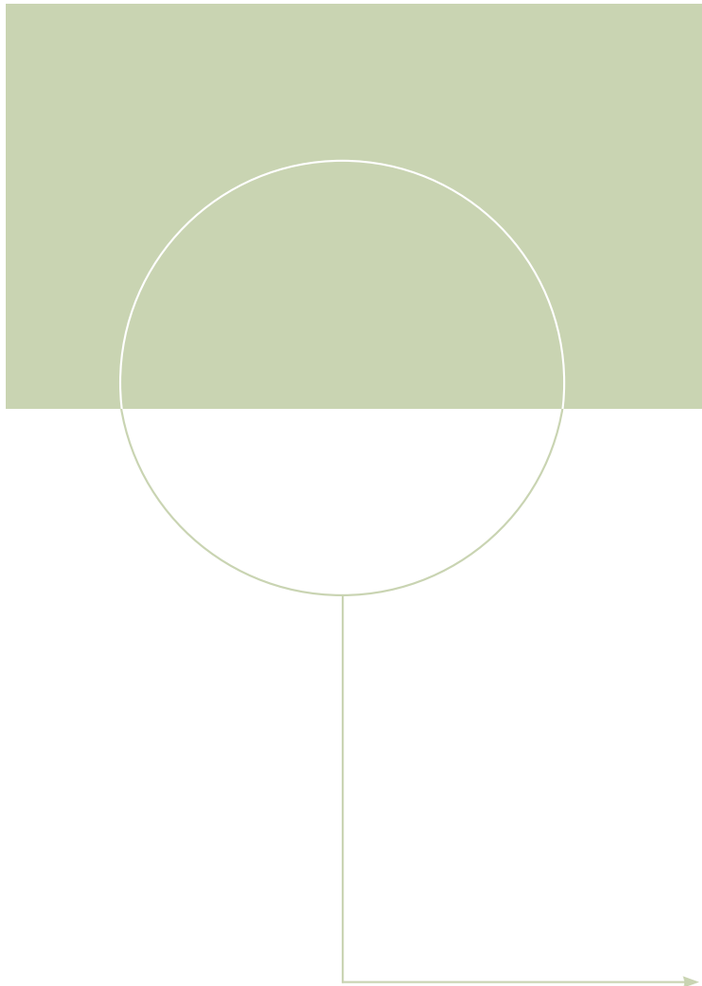


Master's thesis

2021

Sigurd Jordal

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Mathematical Sciences

Sigurd Jordal

Success-rate Estimation for Side Channel Analysis

December 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Success-rate Estimation for Side Channel Analysis

Sigurd Jordal

Master of Science in Mathematical Sciences
Submission date: December 2021
Supervisor: Kristian Gjøsteen
Co-supervisor: Martijn Stam

Norwegian University of Science and Technology
Department of Mathematical Sciences

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Success-rate Estimation for Side Channel Analysis

Sigurd Jordal

December 6, 2021

Abstract.

English:

This thesis explores common techniques and theory behind side channel attacks and side channel analysis. First we look at some popular forms of attacks and ways to calculate the success rate of these attacks. Specifically we establish a mathematical model for what leakage is, and what the attack scenario is in side channel attacks, namely the distinguisher function. Then two specific distinguisher functions are explored, one based on maximum likelihood, and one based on correlation. This is first done in a first order setting, and then expanded to higher orders, where in the leakage model there are several leakage points for each input stemming from a masked implementation of the cryptographic algorithm. Next a side channel attack attribute called the confusion coefficient is explored, which enables the study of how vulnerable a specific cryptographic algorithm is to side channel attacks. Lastly, two ways to estimate the success-rates of different types of attacks are explored, one based on the statistical distribution of the attack, and one based on Monte Carlo simulation.

Norsk:

Denne masteroppgaven utforsker teori og teknikker innenfor side-kanals-angrep og side-kanals-analyse. Først definerer vi den matematiske modellen for lekkasje, og hva angrep-scenarioet er i side-kanals-angrep i form av skille-funksjonen. Deretter blir to typer skille-funksjoner definert, en basert på maksimal sannsynlighet og en basert på korrelasjon. Dette blir først gjort i en endimensjonal scenario og deretter ekspandert til en flerdimensjonal scenario hvor man antar at lekkasjen kommer fra en maskert kryptografisk implementasjon. Deretter defineres kollisjons-koeffisienten, som gjør at man kan relatere kryptografiske algoritmer til hvor sårbare de er til side-kanals-angep. Til slutt blir måter å estimere suksessraten til forskjellige typer angrep beskrevet.

Acknowledgements

I would like to thank Martijn Stam for making this project possible, joining as a co-advisor for this thesis, and being so generous with his time. Without having that expert voice on side-channel theory I would not have gotten as far as I did in this thesis. I would also like to thank Kristian Gjøsteen, who agreed on advising a project in a field neither of us had much experience in.

Lastly, I'm grateful for the the support from everyone in reading room 395b, which has been a really comfortable place to write this thesis.

Sigurd Jordal
Trondheim, 2021

Contents

CHAPTER 1

Introduction

In this thesis we are going to explore theory and techniques within theoretical side channel analysis(SCA). Side channel analysis involves both cryptographic theory and the study of physical hardware, who's respective communities use different terms, and have different intuition on what security terms and attack scenarios are present. This means it will be useful to establish what setting we are in, and what we mean by different terms to minimize confusion from different readers. Hence the introduction into side channel analysis (3) will be somewhat long as to give a proper explanation of the basics that go into side channel analysis. The goal of this thesis is to give an overview of some of the theory in side channel analysis, especially methods to estimate the success-rate of side channel attacks.

For clarity the notation is first described in Chapter (2). As there are a lot of statistical distribution calculations in the thesis we also go through some statistical results, specifically ways to help calculate the expectation and covariance of random variables.

In cryptographic research, especially provable cryptography, the goal is to show equivalences of cryptographic algorithms to known hard problems. In SCA we have a different goal altogether in that we study how vulnerable a cryptographic algorithm run on a computer chip is to key recovery [14]. Key recovery being the task of finding the correct key used in a cryptographic algorithm out of all keys. Hence one can think of side channel analysis as a measuring how vulnerable a cryptosystem is to a more direct chip measuring attack.

The attributes that side channel attacks exploit to recover keys are that there are dependencies between what a computer chip calculates to measurable attributes of that chip. The mainly used ones being its power draw and electromagnetic radiation [16]. The attributes of the computer chip that are useful for key recovery are called side channels, and they generate leakage.

Side channel analysis attempts to study side channel attacks in a theoretical setting [9]. As side channel attacks are practical attacks where one does real life measurements one has to make assumption about how these measurements

behave. Hence we need a model of leakage that is able to predict what kind of leakage one will probably see for given inputs. However by having a model we have to make some assumption about how the real world behaves [4]. Hence we are also making simplifications, this can create some doubt of how closely the real world side channel vulnerability is captured.

In the general leakage model we assume that the chip the cryptographic algorithm is run on calculates some intermediate value of the cryptographic algorithm. This intermediate variable is dependent on the secret key used and the plaintext. When measuring the chip we assume we receive a noisy function of this intermediate variable [14]. What the noise, the leakage function, and model function are depends on the cryptosystem and measuring method in the specific case. When we have specified the functions the important aspect is that one can make predictions of what the leakage will look like for given input. In other words we can hopefully predict what the leakage will look like for given input.

This prediction ability is the essential part of defining a distinguisher. Namely that we use statistical methods to give a score to each key-guess by estimating the probability of measuring some leakage, given plaintext input. In this thesis we go through two different types of distinguishers. One that uses the correlation coefficient (4) to score keys, and one where we have a profile associated with the leakage and then uses a maximum likelihood technique (5).

As side channel attacks exploit the fact that the secret key is handled on a psychical chip, one effective countermeasure is to obfuscate that handling, in other works never handle the intermediate value in the open on the chip. This is done with masking [9]. Intuitively masking is to split up the intermediate value in such a way that one can do calculation on those parts and get back the original value with a calculation applied. This is explained further in Section (6.1), and results in a different leakage model.

The distinguishers in the general leakage model do not take the cryptographic algorithm into account, as we just calculate on the output of the intermediate calculation. This means that we do not take the attributes of the intermediate calculation into account. In Chapter (6), a new view of analysing side channel attacks is introduced. Namely one that takes the cryptographic algorithm, in other words the properties of the intermediate calculation, into account [5]. The confusion coefficient is defined and shown how it relates to side channel attacks. Then both the correlation and a maximum likelihood distinguishers are looked at again, but now in a confusion coefficient perspective. In order to do the distribution calculation in terms of the confusion coefficient some addition calculations have to be done.

The reason for examining the distinguishers of side channel attacks in a theoretical setting is in order to calculate their success-rate. Namely, is it possible to give a good estimation of how probable a side channel attacker is to recover the correct key, given a number of measurements and the accuracy, i.e the noise parameters of those measurements. This thesis introduces two main ways on estimating this success rate. One is by generating model measurements and doing the attack and seeing wither the key is recovered or not (2). The other is by relating the distinguisher's distribution, that we will calculate in the distinguisher chapters, to the probability of the correct key being the highest scoring key-guess(1).

CHAPTER 2

Preliminaries

1. Notation

In this section we go through the notation conventions that is used in the thesis.

By bold $\mathbf{x}$, we mean a vector $\mathbf{x} = (x_i)_i = (x_1, x_2, \ldots, x_n)$ with length n and index x_i to denote the i-th element. Similarly by big bold letter $\mathbf{X}$, we mean a matrix

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{bmatrix},$$

in this case we have a (m, n)-dimensional matrix, though in the thesis this is specifies for each case. For element $x_{i,j}$ of the matrix we write $\mathbf{X}_{i,j}$. Expanding on this to get a row, or a column of a matrix we write a dash, for example to denote the first row we write $\mathbf{X}_{1,-}$.

For a $(n \times n)$- square matrix $\mathbf{X}$, $\mathrm{Tr}(\mathbf{X}) := \sum_{i=1}^{n} \mathbf{A}_{i,i}$. It is called the *Trace* of matrix $\mathbf{A}$.

We will often work with function from sets to sets, for example $f : \mathcal{S} \to \mathcal{S}'$, which denotes a function f from set $\mathcal{S}$ to set $\mathcal{S}'$. Sometimes we also work with vectors with elements from a set, then the space will be denoted as $\mathcal{S}^n$ for n copies of $\mathcal{S}$.

2. Statistical results

The main results of this thesis involves probability distributions. Therefore defining some basic definitions and results is useful. These results can be found in most basic statistics books, for example Larsen et al. [8].

Definition 2.1. By $X \sim \mathcal{N}(\mu, \sigma^2)$ we mean a random variable X with the the univariate Gaussian distribution with mean μ and variance σ^2.

DEFINITION 2.2. By $X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ we mean a random variable X with the multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ.

In this paper there will be many distribution calculations, hence it will be useful to have some distributions results to make the calculation simpler. For samples $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ from a distribution X we write the estimated mean as $\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

LEMMA 2.3. *For random variables $X_1, X_2, \ldots, X_n$, we have that if they are independent then*

$$\mathrm{E}[X_1 X_2 \ldots X_n] = \mathrm{E}[X_1]\mathrm{E}[X_2]\ldots\mathrm{E}[X_n]$$

LEMMA 2.4. *For a random variable X we have that* $\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$

LEMMA 2.5. *For two random variables X and Y we have that* $\mathrm{Cov}[X, Y] = \mathrm{E}[XY] - \mathrm{E}[X]\mathrm{E}[Y]$

LEMMA 2.6. *Adding a constant c or k to the variance of a random variable X or to the covariance of two random variables X and Y does not change it, more formally:*

$$\mathrm{Var}[X + c] = \mathrm{Var}[X]$$
$$\mathrm{Cov}[X + c, Y + k] = \mathrm{Cov}[X, Y]$$

LEMMA 2.7 (Bilinearity of covariance). *For random variables X_1, X_2, X_3 we have that:*

$$\mathrm{Cov}[X_1 + X_2, X_3] = \mathrm{Cov}[X_1, X_3] + \mathrm{Cov}[X_2, X_3]$$

COROLLARY 2.8. *For random variables X_1, X_2, X_3, X_4 we can write:*

$$\mathrm{Cov}[X_1 + X_2, X_3 + X_4] = \mathrm{Cov}[X_1, X_3] + \mathrm{Cov}[X_1, X_4] + \mathrm{Cov}[X_2, X_3] + \mathrm{Cov}[X_3, X_4]$$

For some of the distribution calculation we need to calculate the expectation of the multivariate Gaussian variables. The Isserlis theorem [6] simplifies these calculations:

THEOREM 2.9 (Isserlis theorem). *For a multivariate Gaussian distribution $X_1, X_2, \ldots, X_n$ with zero mean, i.e that $\mathrm{E}[X_1, X_2, \ldots, X_n] = (0, 0, \ldots, 0)$ the expectation of the product is:*

$$\mathrm{E}[X_1 X_2 \ldots X_n] = \sum_{p \in P_n^2} \prod_{\{i,j\} \in p} \mathrm{E}[X_i X_j]$$

Where p denotes all perfect pairings, which is what are summed over.

COROLLARY 2.10 (Isserlis theorem for odd cases). *For a multivariate Gaussian distribution* $X_1, X_2, \ldots, X_n$ *with zero mean, i.e that* $\mathrm{E}[X_1, X_2, \ldots, X_n] = (0, 0, \ldots, 0)$*, and* n *is a odd number, the expectation is:*

$$\mathrm{E}[X_1 X_2 \ldots X_n] = 0$$

Some of our results are based on the Monte Carlo simulation, hence it is useful to know what it is in general to see how we use it.

DEFINITION 2.11 (The Monte Carlo simulation). Also called Gaussian simulation, Monte Carlo Simulation is a method for estimating probabilities by simulating experiments. The idea behind it is that if one performs an experiment under the same conditions many times, one can estimate the probabilities of the experiment. This method works as an alternative to calculating the probability distribution which might require a lot of work.

An example of Monte Carlo simulation is a model of a coin-toss. We have a computer program that uniformly randomly picks either 0 or 1, simulating the 2 sides of the coin. Then one does the "coin-toss" many times, and calculates the average of getting a 1. With enough samples this should become 0.5 and hence we have simulated the probability. This example is quite simple so it would be easier to just calculate the probability.In this thesis however, some of the distribution calculations will be quite complicated, so to calculate the distributions we will make some simplification assumptions. Hence it is useful to use Monte Carlo simulation results as a baseline.

CHAPTER 3

What is side channel analysis?

When we study cryptographic systems we usually think of the cryptographic algorithm as a black box. This means that we only consider the plaintext, the related ciphertext, and the knowledge about the algorithm run. In side channel analysis(SCA) we use the fact that cryptographic algorithms are run on computer chips. This enables us to look at additional attributes of the encryption/decryption process. Namely we exploit the fact that a computer chip has, among other attributes, a varied power draw, and electromagnetic radiation depending on what is calculated.

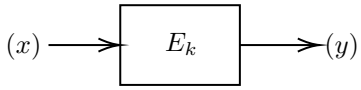
These are the main types of attributes of what we will call the *side channels* of the cryptographic implementation. We will utilize the dependency between the secret key of a cryptographic algorithm and these side channels in order to recover that key. In this way we are not only in a different model than a usual cryptographer, but we also have a different goal. Instead of showing indistinguishably attributes of the algorithm, we here attempt to recover the actual key used in encryption/decryption.

In order to get an overview of what is involved in a side channel attack we will take a closer look at the different aspect of such attacks. Namely we will describe what the goal of an attack is, what physical properties we exploit in the attack, and how we model these physical properties. With these three concepts described we can then define a general attack method.

1. The goal of side channel attacks

A cryptographic algorithm encrypts and decrypts something using a secret. Without having this secret the encrypted information should be difficult to reveal. It should also be difficult to find this secret by looking at the input, called the plaintext, and the output, called the ciphertext. If somebody gets access to the secret they can encrypt and decrypt as they want. So if the secret key is possible to find the implementation is compromised and naturally no longer secure either.

In side channel analysis secrets like these are what we are trying to recover. They usually take the from of some kind of secret key, like a bit-string. When we analyse cryptosystems we usually think of either an encryption or decryption algorithm that inputs a plaintext or a ciphertext and outputs the other. In side channel analysis we look at a general cryptographic algorithm, lets call this E_k, where $k \in \mathcal{K}_{FK}$ is the secret key. Here $\mathcal{K}_{FK}$ is the entire keyspace of the cryptosystem. The information we are given (either the plaintext or the ciphertext) is given as $x \in \mathcal{X}$, in its related(either plaintext or ciphertext space), $y \in \mathcal{Y}$ is defined the same way. More visually this can be seen as:



DEFINITION 3.1. *Key recovery* is the goal to retrieve the key used in some cryptographic process. More specifically given some information related to E_k, the goal is to return k, the key used, out of the set of keys $\mathcal{K} = \{k_1, k_2, ...k_{N_k}\}$, where N_k is the number of different possible key candidates.

The reason we specified above that $\mathcal{K}_{FK}$ is the full keyspace is that we will usually work in a different smaller keyspace. What we want to end up with is a method to score likely key-guesses in order to recover the correct one. A naive attempt may be to try to score the full key, however one quickly realizes that this is usually infeasible. If we want to make any algorithm that needs to do a calculation on every key option in order to find a likely key candidate, then we might as well check if those key-guesses decrypt/encrypt correctly. Hence we will look at specific guessable parts of keys, where the search space is much smaller and our search can be worthwhile. We will call these sub-keys the *guessable part of the key* denoted $k \in \mathcal{K}$. Where $\mathcal{K}$ is the keyspace of the guessable parts, and a part of $\mathcal{K}_{FK}$.

By guessable part we mean the part of the key we are making an attempt to recover using side channel techniques, for now we can think of it as a part of the full key. By looking at guessable parts of the key, we can make a method to recover the full key piece-wise, recovering the parts one by one. Checking that the guessable keys are correctly recovered is problematic, but for the full key we can usually check that we have the correct key with a couple of known plaintext ciphertext pairs and then check if the recovered full key encrypts/decrypts correctly.

EXAMPLE 3.2. Throughout this thesis we will use AES-128 as the example cryptosystem for side channel analysis. Hence it can be useful to define what a key looks like in this setting. The full key is 128-bits, hence the name. If we

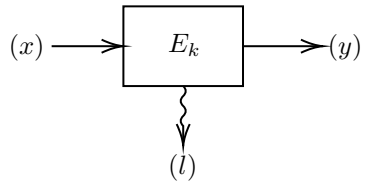
were to attempt to score the full key we would have to score 2^{128} different key candidates, which is infeasible. However during encryption and decryption the algorithm splits the key up into 8-bit round keys for the S-box permutation. In the case of AES-128 the guessable part of the key will thus be the 8-bit round key, which only has 256 key candidates. These round keys can be thought of as a number between 0 and 255. Hence the a successful guessable key recovery is one where we can find the round key k used in one S-box permutation. The for a successful recovery of the full key, one has to do this for all the different round keys.

The goal of side channel attacks is hence to recover the key of a cryptographic algorithm. Side channel attacks are not unique in this goal, though the method it goes about recovering these keys is what gives this attack method the name side channel.

2. Defining side channels

The essential part of side channel analysis is the study of the physical properties of the chip that the cryptographic algorithms are run on. We will call the parts of these physical properties that are useful for key recovery *side channels*. In full generality these side channels could be based on the timing of occurrences [2] or on the information that was on the chip before.

In this setting we are investigating the subset of side channels where it is directly dependent on the input of the algorithm. These side channels give out information about what is going on in the cryptographic algorithm, in other words they leak information. Hence we define *leakage* in this setting as all unintended information the physical device that calculates the cryptographic algorithm exposes. We can go back to the cryptosystem diagram to make this clearer:



The setting is still the same with an cryptographic algorithm E with a guessable secret key k that we want to recover. In addition we have a side channel, denoted by the squiggly arrow, and this side channel generates some leakage l for each input of the algorithm. We have not yet assigned any properties for how this side channel arrow behaves, but we will see that the addition information can be used to recover the secret key k used in the algorithm. The general idea is that for different secret keys k the resulting leakage l will behave differently.

The diagram is supposed to represent a side channel scenario in an abstract setting, but a side channel attack is an attack based on collecting data of the chip the algorithm in run on. Hence it can be useful to also look at an example of a real side channel attack, and what attributes it uses. Side channels in many cases takes the form of the power consumption, or the electromagnetic radiation coming from the chip of a cryptographic device. Hence in these cases the leakage is the measured current and the measured electromagnetic radiation respectively. All electronic devices has a power consumption and radiate electromagnetic fields, and this consumption and radiation also fluctuate depending on what is happening in the chip. How these things fluctuate indicates what calculations are happening within the chip, and is what side channel attacks exploit [14].

The assumption that is made in side channel analysis is that that leakage is dependent on the guessable secret key used in the cryptographic algorithm, and hence it is possible to use statistical attacks to estimate likely keys candidates.

Example 3.3. Power consumption

Depending on what a computer chip does it uses different amounts of power. A good intuitive understanding of how an attackers can exploit this can be gained from looking at the power trace of RSA. In RSA encryption and decryption one can use the square and multiply method to calculate a big exponent quickly. This gives the implementation a weakness in the side channel setting, namely that for a computer chip multiplying requires more energy than squaring[7]. Hence by just looking at the power trace of an RSA decryption in Figure 1, one can read out where the algorithm has squared and multiplied and in that way easily find the secret key used.

Example 3.4. AES-128

Now that we have a concept of what leakage can take the form as, we can look at it in the context of AES-128. As before we are in the setting of the S-box permutation and the round key. From Ors et al. [11], we have that this S-box permutation is an operation where one can reliably measure the power consumption on the chip. In the terms of this thesis that means that it generates leakage. Hence we can attempt to recover the different round keys, where there are only 256 key candidates.

3. Modelling leakage

Now that we have gotten an impression of what leakage is physically we need a way to model it mathematically in order to use it for a side channel attack. Specifically we need a leakage model that can predict how leakage would probably

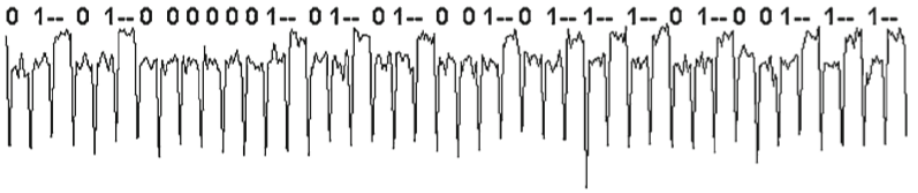


FIGURE 1. The Figure from Kocher et al. [7], shows the power consumption over time from a RSA encryption round. Due to the difference in power consumption for multiplication and squaring, one can easily determine what the secret key is in binary form. This is indicated by the binary number above the power trace.

look for a cryptographic algorithm, given some input and key. In order to do that we need to make certain assumptions on how what is happening within a side channel and how leakage is related to the input.

As described before we are in the setting of a general cryptographic algorithm E_k where k is the secret key. Looking at $E_k(x) \longrightarrow y$ from a leakage perspective, what we receive as leakage information is a function of what happens during the computation. In other words we can think of it as an intermediate calculation that we assume is dependent on the input x and the guessable part of the secret key k.

DEFINITION 3.5. An *intermediate value function* is a function $\varphi : \mathcal{X} \times \mathcal{K} \longrightarrow \mathcal{S}$ that takes in a plaintext or ciphertext and a guessable key value and returns an *intermediate value* s also called the *signal*.

What the specific intermediate value function is depends on what cryptographic algorithm we are attacking. The set $\mathcal{S}$ is the space of the intermediate calculations, and is what we assume we would see if we had a perfect view of the intermediate calculation on the chip where the algorithm is processed. The intermediate calculation is deterministic when we know the (x, k) pair of $\varphi(x, k)$. However when we look at the intermediate value in a leakage setting we do not know the correct key k. This means that we will considered it as a random variable over $\mathcal{S}$ when scoring the different keys, as we cannot determine what the intermediate calculation value will be, only based on the input of (x) without knowing k.

In the real world we need to measure the intermediate value on a physical device. As a result of the physical measuring we do not have a perfect view of the intermediate calculation, it first has to be processed through the side channel, and the way we measure it might be different from the intermediate calculation. Hence the leakage stemming from the intermediate calculation will be some model of that intermediate calculation.

DEFINITION 3.6. A *leakage model* is a function $f : \mathcal{S} \to \mathcal{L}$, from the set of intermediate values to the leakage set.

What this leakage model function is defined as is dependent on what kind of side channel we are exploiting. When measuring the power draw or electromagnetic radiation of a chip one usually models it as the hamming weight of the intermediate value. Although this does not necessarily reflect real life measurements completely, it is common for the side channel literature, and useful for the purpose of this thesis where we estimate success-rates.

EXAMPLE 3.7 (Power consumption of load operations). A smart card is a credit card sized integrated circuit often used for identification. It is usually a simple circuit which also leads the power consumption of it to be highly related to what is happening on the data bus. Depending on the data that is loaded different amounts of voltage is used. Hence the more bits that change for the when loading something into memory a higher voltage is used, and this enables us to relate the hamming weight of what is loaded to the power consumption.

In order to show this Messerges et al. [10] measured the power consumption used by an 8-bit smart-card, when loading different data from the memory onto the bus. One can see this difference in Figure 2. This fact leads to the reason that the Hamming weight is often used as the leakage model in Side Channel literature. Although according to Costes and Stam [4], it was only the load operation where the hamming weight model was proper to use.

In addition to the side channel transforming the intermediate calculation, the actual measurement is not perfect either, there will be some noise i.e errors involved in the leakage we record. In this sense we will look at leakage $\mathbf{l} \in \mathcal{L}$ plus some noise, instead of only the leakage itself. To express this $\mathbf{l}$ we assume that we measure the leakage model function of the intermediate value with some normally distributed noise. This is called the *Gaussian assumption* of side channel analysis. Hence $\mathbf{l}$ is the addition of two random variables, namely $\mathbf{l}(x, k) = f(\varphi(x, k)) + n$. In this assumption n is a sample of a random variable and denotes the noise of the side channel. This noise is assumed to be independent from the model values

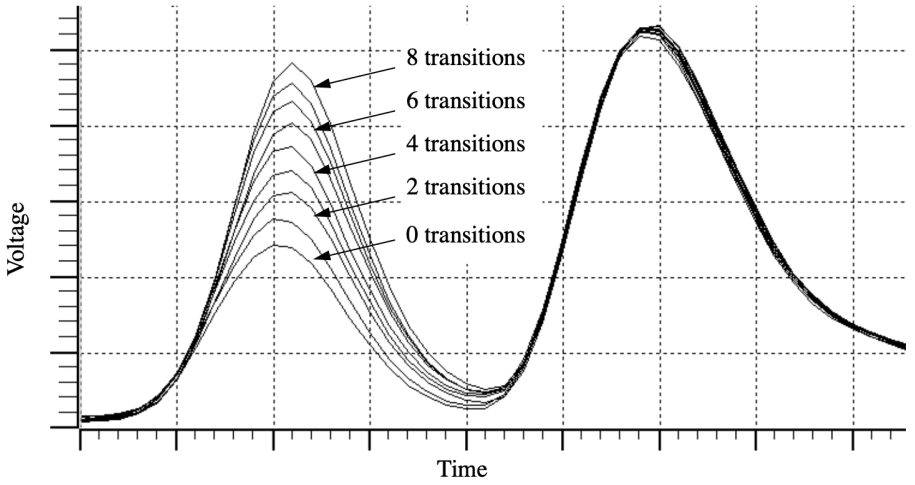
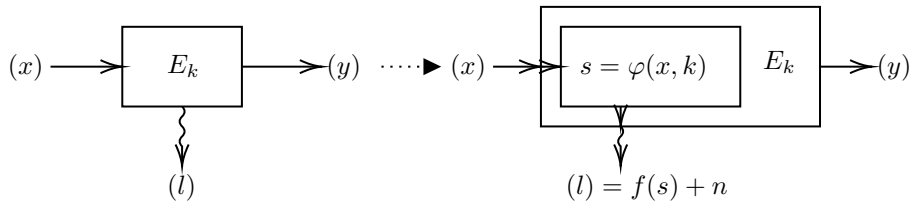


FIGURE 2. In the Figure taken from Messerges et al. [10] one can see the difference in power consumption between the number of bit-transition the 8-bit smartcard. This enables us to relate load operation to the hamming weight, the transition distance is about 6.5 mV.

and has distribution $\mathcal{N}(0, \Sigma)$. The function φ as mentioned is the specific intermediate calculation that takes in an input x and a key k, and returns a signal $s \in \mathcal{S}$. We have now described more attributes of how the side channel setting looks like, the differences can be seen in the figure below:



The essential part of modelling leakage is that we have a way of estimating how the leakage of the same input x will look under different secret keys k. By having such a model this enables us to have an estimation of what leakage should look like with a given key and input x.

4. Defining a distinguisher

The basis of the success of side channel attacks is the assumption that the leakage is dependent on the secret key used in the calculation. So how do we use this

dependence in order to get at the key. There are only a limited number of guessable keys, and with the dependence between the leakage and the input we can create a function called a *distinguisher*.

DEFINITION 3.8. A *distinguisher* is an algorithm $D : \mathcal{X}^n \times \mathcal{L}^n \to \mathbb{R}^{|\mathcal{K}|}$.

The algorithm D inputs the known values $\mathbf{x} = (x_i)_i$ with the corresponding leakage $(l_i)_i$ and outputs distinguishing scores $(d_1, d_2, ..., d_{N_k})$. In an actual attack the leakage vector $\mathbf{l} = (l_i)_i$ is coming from an implementation of some cryptographic algorithm E_k. The index d_i indicates the key-guess i among the number of all guessable keys $N_k = |\mathcal{K}|$. In other words a score for each possible key based on the input and resulting leakage. For $k \in \mathcal{K}$ we can write a single key score as d_k. Hence a distinguisher takes in the input vector $\mathbf{x}$, and the corresponding leakage vector $\mathbf{l}$, and returns a vector of real numbers each corresponding to the score of that key. We will see that the intermediate value function is essential to create distinguishers as they enable us to predict what leakage we would get based on the input and key-guess. Then we can compare the output of the model to the actual leakage we have measured, and attempt to give find what key was probably used in the actual measurements.

DEFINITION 3.9. For a distinguisher D with input $\mathbf{x}$ and leakage $\mathbf{l}$ generated with key k^* one can express a *successful attack* as when

$$k^* = \arg\max_{k \in \mathcal{K}} d_k.$$

This means that the highest scoring key given the input $(x_i)_i$ and the leakage $(l_i)_i$ is the correct one. We also want to express the probability of success, this can be written as the probability of the correct key being the biggest one for distinguisher D given input and leakage. More formally we write this as

$$\text{Succ}^D_{\boldsymbol{x},k^*} = P\left[(l_i \leftarrow \mathcal{L}(\varphi(k^*, x_i)))_i; \boldsymbol{d} \leftarrow D(\boldsymbol{x}, \boldsymbol{l}) : k^* = \arg\max_{k \in \mathcal{K}} d_k\right].$$

This might not be completely sufficient to describe the success probabilities of an side channel attack though. We could image a case where the correct key is not the biggest d_k but in the top hundred. For an attacker it is still quite simple to check a big number of high scoring keys to see if the correct one is there. To remedy this we introduce the concept of o-th order success, in other words the chance of the correct key guess being the o-th highest distinguisher score or higher. We will therefore denote $\arg\max\text{-o}_{k \in \mathcal{K}} d_k$ as the set of the o biggest key-scores. Hence we can write the probability of an o-th order success as

$$\text{Succ-o}^D_{\boldsymbol{x},k^*} = P\left[k^* \in \arg\max_{k \in \mathcal{K}}\text{-o}\, d_k \middle| (l_i \leftarrow \mathcal{L}(\varphi(k^*, x_i)))_i; \boldsymbol{d} \leftarrow D(\boldsymbol{x}, \boldsymbol{l})\right].$$

A distinguisher gives a score to each key-guess, however it is not strictly the size of these scores that are essential. What we care about is the order of those scores, hence if two distinguishers have different scores but keeps the same order, in the view of side channel analysis they are equivalent, this gives rise to the definition:

DEFINITION 3.10. Two distinguishers d and $\dot{d}$ are considered *equivalent* if the order of the key-guess scores are the same for the same input, and we denote it as $d \equiv \dot{d}$.

If two distinguishers are equivalent and one is simpler or faster to calculate than the other, we will naturally prefer the faster one. Simpler equivalent distinguishers will also be useful when calculating success rates.

As we have seen, a distinguisher takes in the plaintext $\boldsymbol{x}$ and its corresponding trace $\boldsymbol{l}$ in order to score the keys, it would however be useful to have a distinguisher that can score on a trace by trace basis and then just add the scores together. This is the specification of an additive distinguisher:

DEFINITION 3.11. A distinguisher d is called *additive* if given plaintext $(x_1, x_2, ...x_N)$ we can express the key scoring $d_k = \frac{1}{N}\sum_{i=1}^{N} g_{x_i,k}(l_i)$ for each trace l_i and for family of functions $g_{x_i,k} : \mathcal{L} \to \mathbb{R}; (x, k) \in \mathcal{X} \times \mathcal{K}$

In this definition of an additive distinguisher we are given the full plaintext vector, however to get a scoring where we only use each plaintext leakage pair by itself we need a slightly different definition:

DEFINITION 3.12. A distinguisher d is called *properly additive* if we can express the key scoring $d_k = \frac{1}{N}\sum_{i=1}^{N} g_k(x_i, l_i)$ for each plaintext, trace pair x_i, l_i and for family of functions $g_k : \mathcal{L} \to \mathbb{R}; k \in \mathcal{K}$

5. Relating the distribution of leakage to distinguishers

In Section 3 the statistical model for leakage was introduced. Here the leakage is assumed to be a function of a intermediate variable with Gaussian noise. Hence there is a distribution associated to the leakage. Due to the fact that distinguishers has this leakage as input, it is also possible to calculate a related distribution to a distinguisher. This is particularly useful as it is possible to relate the score of the correct key to a probability. This probability will be one of the way to calculate the success rates of distinguishers. But in order to do this the distribution of the distinguisher is needed. For additive and properly additive distinguisher it will therefore be useful to relate the distribution of the

distinguisher to the distribution of the family of function $g_{x_i,k}(l_i)$ that adds up to the distinguisher.

PROPOSITION 3.13. *The distribution of an additive distinguisher* $(d_k)_{k\in\mathcal{K}} = \left(\frac{1}{N}\sum_{i=1}^{N} g_{x_i,k}(l_i)\right)_{k\in\mathcal{K}}$ *satisfies*

$$\mathrm{E}[d_k] = \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}[g_{x_i,k}(x_i)], \tag{1}$$

$$\mathrm{Cov}[d_{k_1}, d_{k_2}] = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Cov}[g_{x_i,k_1}(x_i), g_{x_i,k_1}(x_i)]. \tag{2}$$

for every $k, k_1, k_2 \in \mathcal{K}$.

PROOF. The expectation is easy to see as $\mathrm{E}[X_1+X_2+\ldots+X_m] = \mathrm{E}[X_1]+\mathrm{E}[X_2]+\ldots+\mathrm{E}[X_m]$, hence the expectation of the distinguishers is just the average of the expectation $g_{x_i,k}(x_i)$ for each trace as described.

In order to calculate the covariance we have to use the assumptions that all the samples are mutually independent. Hence

$$\mathrm{Cov}[d_{k_1}, d_{k_2}] = \mathrm{Cov}[\frac{1}{N}\sum_{i=1}^{N} g_{x_i,k_1}(l_i), \frac{1}{N}\sum_{i=1}^{N} g_{x_i,k_1}(l_i)]$$

$$= \frac{1}{N^2}\mathrm{Cov}[\sum_{i=1}^{N} g_{x_i,k_1}(l_i), \sum_{i=1}^{N} g_{x_i,k_1}(l_i)]$$

Since we assumed all the samples are independent the covariance is only non-zero when it is the same sample, hence we can write it as $\frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Cov}[g_{x_i,k_1}(l_i), g_{x_i,k_1}(l_i)]$, which is what we wanted. □

6. Countermeasures against side channel attacks

Since side channel attacks are an efficient form of attack, there has been many proposals for how to protect implementations from such attacks. Most of the early countermeasures were more ad-hoc, and therefore also not difficult to circumvent [3]. However one countermeasure that has remained a good method to prevent side channel attacks, is the use of masking. In a masked implementation of a cryptographic algorithm the intermediate value $s = \varphi(x, k)$ is never handled in the open but split into several shares. We can write the relationship between these shares and the actual value as $s_0 \oplus s_2 \oplus ... \oplus s_d = s$ for some operation $\oplus$.

We will often consider the case of Boolean masking so it could be useful with an example of how this looks like:

Example 3.14. Boolean Masking

In Boolean masking $\oplus$ is defined to be the XOR operation, we pick $d-1$ random shares in the correct size as the value we want to mask. In the case of the AES-128 leakage model this is a byte. Hence we pick $d-1$ random bytes $s_0, s_1, ..., s_{d-1}$ then calculate:

$$s \oplus s_0 \oplus s_1, ... \oplus s_{d-1} = s_d$$

As indicated the result of this calculation will be the final share. One can see that this satisfies the purpose of a masking technique, namely that we can recover the original s by XOR-ing all the shares:

$$s_0 \oplus s_1 \oplus ... \oplus s_{d-1} \oplus s_d = s$$

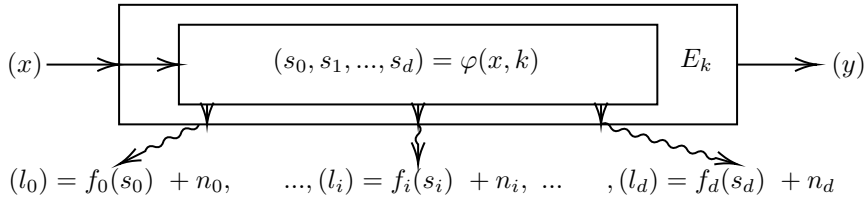
In this context we can see that masking is basically a different word for of secret sharing [15], but in this setting the goal is not to distribute a secret between several parties, but to obfuscate the handling of the cryptographic algorithm on the chip.

The hope for masking is that it will require more measurements to reach the same success rate in a masked implementation as opposed to a standard implementation. In [3] and [12], they discuss the general security of a masked implementation compared to a standard one. They show that a masked implementation of a cryptographic algorithm requires exponentially many more leakage measurements, where the exponent is the number of masks, to reach the same success rate. Using masking has several consequences for how we model leakage. So we will not only need to change the model, but also how the distinguishers is defined.

6.1. Modelling leakage in a masked implementation. Masking makes key-recovery more difficult as the behaviour of the leakage is more complex, and the intermediate value $s = \varphi(x, k)$ is never handled in the open. In this section we will use the leakage model of Lomné et al. [9] adapted into this paper's notation. The leakage setting is similar as before as we also have a leakage vector $\mathbf{l}$ with traces, but these traces are no longer just a single value but vectors in their own right. This can be written as $\mathbf{l} = (\mathbf{l}_1, \mathbf{l}_2, ..., \mathbf{l}_N)$, where $\mathbf{l}_i = (l_{i,1}, l_{i,2}, ..., l_{i,d})$, and d is the masking order, i.e the number of masks or shares that's in the implementation. This means that for each plaintext x_i we have an associated leakage vector $\mathbf{l}_i$ and not just a value. Due to this fact we will call a masked cryptographic implementation, a higher order model, and a non masked implementation a first-order model.

Higher order leakage models are, similarly to first-order leakage models, also assumed to be related to some intermediate calculation φ, however since we are in a higher order setting the intermediate value is now a vector of length d, hence $\varphi(x, k) = \mathbf{s} = (s_0, s_1, ..., s_d) \in \mathcal{S}^{d+1}$, where $\mathcal{S}$ is the space of intermediate calculation for each share. The intermediate calculation function could also be described on each single share, but for the sake of simplicity it can be described as outputting the intermediate calculation of each share.

Similarly as in the unmasked case, it is assumed that what is measured as leakage is a noisy function of this intermediate calculation. This noise is assumed to be Gaussian for each mask/share measurement, and is as before called the *Gaussian assumption*. A visual representation of the masked leakage model can be seen in the figure below:



There is also the need to make an assumption about the relationship between the different leakage measurements of the shares. For the masking technique we constructed them in a way such that all the masks are mutually independent. This is also assumed for the leakage measurements, namely that one mask as a random variable l_i is mutually independent from the other shares l_j. This is called the *independent noise assumptions* from Lomné et al. [9].

A distinguisher can still be defined in the same way as before, but the actual distinguisher method needs to be changed in order to account for $\mathbf{l}_i$ being a vector and not just a value. We will go further into detail on how these higher order distinguishers work in chapter 4 and 5. Now that there is two different leakage scenarios there will also be two different success-rate estimations. Namely one for the unmasked first order setting, and one for the masked higher order setting. Here one can also estimate how a masked implementation changes the success rate and hopefully requires more traces to reach the same success-rate.

CHAPTER 4

A correlation based distinguisher

We introduced the notion of distinguishers in the last chapter, and now we will go further by defining specific ones, namely a correlation based distinguisher. As in the theoretic introduction of distinguishers these will take in all plaintext and leakage/trace pairs and give a score to each key-guess, if the highest scoring key-guess is the correct one we consider it an successful attack.

Correlation, also called differential, based distinguishers exploit the assumption that the leakage is correlated to some function of the plaintext. We will do a key dependent transformation of the plaintext, then calculate a correlation coefficient between this transformed plaintext vector and the trace vector. The most common correlation coefficient that is used is the Pearson coefficient, and is also claimed to be the most efficient [1]. The basic technique for the different coefficient types are however not very different. The general formula for Pearson coefficient for vectors $(\mathbf{v}, \mathbf{w})$ can we written as:

$$\rho(\mathbf{v}, \mathbf{w}) := \frac{\mathrm{Cov}[\mathbf{v}, \mathbf{w}]}{\sigma_{\mathbf{v}} \sigma_{\mathbf{w}}}$$

Where $\sigma_{\mathbf{w}}$ and $\sigma_{\mathbf{v}}$ are the standard deviation of $\mathbf{w}$ and $\mathbf{v}$ respectively.

1. Constructing an unmasked first order correlation distinguisher

In this scenario we assume we are given plaintext vector $\mathbf{x} = (x_0, x_1, ..., x_N)$ and accompanying leakage vector $\mathbf{l} = (l_0, l_1, ..., l_N)$. When constructing the distinguisher and calculating the distribution we follow the theory from Rivain in [14]. We cannot directly calculate the correlation between these two vectors, so we need a key dependent *model function* $\mathrm{m} : \mathcal{X} \times \mathcal{K} \to \mathbb{R}$ with $\mathrm{m}(x, k)$ being linearly related to the theoretical leakage $L(\varphi(x, k))$. In order to calculate the correlation we calculate $\mathrm{m}(\mathbf{x}, k) = (\mathrm{m}(x_0, k), \mathrm{m}(x_1, k), ..., \mathrm{m}(x_N, k))$ for key-guess k then calculate the correlation $\rho(\mathrm{m}(\mathbf{x}, k), \mathbf{l})$. In other words, for each key-guess we use that key's model function, then look at the correlation coefficient. After having calculated all these, we find the key-guess with the maximum correlation and use

this as our guess for the correct key. More rigorously for plaintext and traces $(\mathbf{x}, \mathbf{l})$, a key-guess will look like

$$\rho_k(\mathbf{x}, \mathbf{l}) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)\left(l_i - \frac{1}{N}\sum_{j=1}^{N} l_j\right)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)^2}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(l_i - \frac{1}{N}\sum_{j=1}^{N} l_j\right)^2}}.$$

In addition to making a model specific correlation calculation there is also the need to estimate the standard deviation of the modelled leakage and the actual leakage. So instead of the theoretical $\sigma_{\mathrm{m}(\mathbf{x},k)}$ and $\sigma_{\mathbf{l}}$, it is estimated by the usual standard deviation estimation. Now we can properly define the CPA distinguisher vector $\mathbf{d}$ by calculating the correlation coefficient for each key-guess, i.e $\mathbf{d}_{\mathrm{corr}} = (\rho_0(\mathbf{x}, \mathbf{l}), \rho_1(\mathbf{x}, \mathbf{l}), ..., \rho_{N_k}(\mathbf{x}, \mathbf{l}))$. Here N_k denotes the total number of guessable keys.

From the calculation of the correlation we can see that we need all traces $\mathbf{l}$ for each trace score, hence it is neither additive or properly additive. Note however that in the formula for ρ, the standard deviation leakage vector is not an essential part of the different key-guesses k, as it is constant for all the different key-guesses. This enables us to define a different distinguisher that does not involve the need for all traces when calculating the score for one plaintext trace pair. Hence we disregard the standard deviation of the leakage vector which results in a new distinguisher,

$$\dot{\rho}_k(\mathbf{x}, \mathbf{l}) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left(\mathrm{m}(x_i,k) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)l_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)^2}} = \frac{1}{N}\sum_{i=1}^{N} g_{(x_i,k)}(l_i).$$

We can see that $\dot{\rho}$ is additive, as we can define it as the average of all the trace scores $g_{(x_i,k)}(l_i)$. It is still not properly additive as we have to calculate the average $\overline{\mathrm{m}}_k = \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)$, for scoring each plaintext, leakage pair (x_i, l_i), and here we need the full plaintext vector. We will now show that $\rho \equiv \dot{\rho}$, i.e that the two distinguishers are rank equivalent. To clean up the calculation somewhat we will denote the average of the model vector and leakage vector as $\overline{\mathrm{m}}_k$ and $\bar{\mathbf{l}}$ respectively. If they were not rank equivalent it would not be interesting to look at $\dot{\rho}$ by itself. One way to show that two distinguishers are rank equivalent it to show that $\frac{\dot{\rho}}{\rho} = c$, where c is a constant.

LEMMA 4.1. $\rho \equiv \dot{\rho}$

PROOF.

$$\frac{\dot{\rho}_k(\mathbf{x},\mathbf{l})}{\rho_k(\mathbf{x},\mathbf{l})} = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)l_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)^2}} \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)^2}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(l_i-\bar{\mathbf{l}}\right)^2}}{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)\left(l_i-\bar{\mathbf{l}}\right)}$$

$$= \frac{\frac{1}{N}\sum_{i=1}^{N}\left(\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)l_i\right)\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(l_i-\bar{\mathbf{l}}\right)^2}}{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)\left(l_i-\bar{\mathbf{l}}\right)}$$

$$= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(l_i-\bar{\mathbf{l}}\right)^2}\frac{\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)l_i}{\sum_{i=1}^{N}\left(\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)l_i\right)-\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)\bar{\mathbf{l}}}$$

$$= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(l_i-\bar{\mathbf{l}}\right)^2} = \sigma_{\mathbf{l}}$$

The jump to the last line is due to the fact that

$$\sum_{i=1}^{N}\left(\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right)\bar{\mathbf{l}}\right) = \bar{\mathbf{l}}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\right) =$$

$$\bar{\mathbf{l}}\left(\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)\right)-N\overline{\mathrm{m}}_k\right) = \bar{\mathbf{l}}\left(\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)\right)-\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)\right)\right) = 0.$$

□

Hence we have that $\frac{\dot{\rho}}{\rho} = \sigma_{\mathbf{l}}$, the standard deviation the traces, which is constant for all key-guesses. This means that the order of the key-scores will be the same in ρ and $\dot{\rho}$, so $\rho \equiv \dot{\rho}$. Now we have defined a simplified first order distinguisher, which enables to perform side channel attacks. However this distinguisher cannot handle a masked cryptographic algorithm, hence we will need to go back to the construction in order to define a masked distinguisher.

2. Constructing a masked higher order correlation distinguisher

In section 6, chapter 3 we introduced the concept of a masked cryptographic algorithm. The result of this is that for each plaintext x_i we have a trace vector $\mathbf{l}_i$, and not a single point. The construction and distribution calculation of this

higher order distinguisher takes its theory from Lomné et al. in [9]. To still use the Pearson coefficient we need a way to change this leakage vector to something manageable. This is solved by using a *combining function*, $\mathrm{C} : \mathcal{L} \to \mathbb{R}$. This is basically a function that takes the leakage vector to a real number in such a way that the leakage and the modified plaintext is still linearly related. With this combining function we can calculate the Pearson coefficient as usual, with the traces transformed using the combining function and the plaintext with the model function, more rigorously this looks like:

$$\rho_k(\mathbf{x}, \mathbf{l}) = \rho(m(\mathbf{x}, k), \mathrm{C}(\mathbf{l})) = \frac{\frac{1}{N}\sum_{i=1}^{N} \left(\mathrm{m}(x_i, k) - \overline{\mathrm{m}_k}\right)\left(\mathrm{C}(\mathbf{l}_i) - \overline{\mathrm{C}(\mathbf{l})}\right)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(\mathrm{m}(x_i, k) - \overline{\mathrm{m}_k}\right)^2}\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(\mathrm{C}(\mathbf{l}_i) - \overline{\mathrm{C}(\mathbf{l})}\right)^2}}$$

There are several choices of combining functions and the function is related to the distribution for success rate calculation. Now we will look at a equivalent distinguisher that has the added property of being additive. This distinguisher is identical as $\dot{\rho}$ except with the combining function of the leakage vector instead of just the trace:

$$\mathbf{d}_{\mathrm{cor}} = (\dot{\rho}_k(\mathbf{x}, \mathbf{l}))_{k \in \mathcal{K}} = \left(\frac{1}{N} \sum_{i=1}^{N} \frac{\left(\mathrm{m}(x_i, k) - \frac{1}{N}\sum_{j=1}^{N} \mathrm{m}(x_j, k)\right)\mathrm{C}(\mathbf{l}_i)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(\mathrm{m}(x_i, k) - \frac{1}{N}\sum_{j=1}^{N} \mathrm{m}(x_j, k)\right)^2}} \right)_{k \in \mathcal{K}}$$

We have preciously shown that $\dot{\rho}_k \equiv \rho_k$, and this also applies to when we use it in conjunction with the combining function. To calculate the success-rate distribution of a higher-order correlation distinguisher a combining function needs to be picked. Here one example of such a function is presented, but others are also possible.

Definition 4.2. The *centering product* is a function $\mathrm{C} : \mathbb{R}^d \to \mathbb{R}$ defined as $\mathrm{C}(\mathbf{l}_i) = \mathrm{C}(l_0, l_1, \ldots, l_d) = \prod_{j=0}^{j=d} (l_j - \mu_j)$ where $\mu_j = \mathrm{E}[L_j]$, i.e the expected value of the leakage of the "j-th" mask.

In an real-life attack the mean $\mathrm{E}[L_j]$ will need to be estimated with the traces. This complicates the calculation of the success-rate somewhat, as we need to take

the precision of the mean estimation into account when calculating it. However in [13], they argue that the number of traces needed to have a successful attack is big enough to get a precise estimation of $\mathrm{E}[L_j]$. This still leads to some uncertainty about the success-rate of low trace number scenarios. With the centering product the calculation of the distinguishing score for key guess k will look like

$$\dot{\rho}_k(\mathrm{m}(\mathbf{x}, \mathrm{C}(\mathbf{l})) = \frac{1}{N}\sum_{i=1}^{N} \frac{\left(\mathrm{m}(x_i,k) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)\left(\prod_{j=0}^{j=d}(l_{i,j}-\mu_j)\right)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)^2}}$$

Now that the distinguisher is fully described we have all the tools needed to calculate the distribution of the scores.

3. First order correlation distribution calculation

In order to find the success rate of different attacks we need to calculate the distribution of the key-scores of different key-guesses. Here we need to find the expected value of a general key-guess, and the covariance between two key-guesses, i.e: $\mathrm{E}[d_k], \mathrm{Cov}[d_{k_1}, d_{k_2}]$.

Since $\rho \equiv \dot{\rho}$ we can calculate the distribution of the simpler $\dot{\rho}$ and still have the same success-rate. This is first done in the unmasked setting, then in the masked higher order setting. How these distribution are used in order to reach theoretical success rates is explained in chapter 7.

3.1. First order distribution.

PROPOSITION 4.3. *For distinguisher* $\mathbf{d} = (\dot{\rho}_k)_k$ *has expectation and covariance satisfying:*

$$\mathrm{E}[\dot{\rho}_k] = \frac{1}{N\sigma_k}\sum_{i=1}^{N}\left(m(x_i,k) - \frac{1}{N}\sum_{j=1}^{N} m(x_j,k)\right)m_{x_i,k^*} \tag{3}$$

$$\mathrm{Cov}[\dot{\rho}_{k_1}, \dot{\rho}_{k_2}] = \frac{1}{N^2\sigma_{k_1}\sigma_{k_2}}\sum_{i=1}^{N}\left(m(x_i,k_1) - \frac{1}{N}\sum_{j=1}^{N} m(x_j,k_1)\right)\left(m(x_i,k_2) - \frac{1}{N}\sum_{j=1}^{N} m(x_j,k_2)\right)\sigma^2_{x_i,k^*} \tag{4}$$

PROOF. The expectation calculation of the variables that do not have a distribution associated with them except is just the variable itself. This is true for all the

variables except for l_i which has distribution $\sim N(m_{x_i,k^*}, \sigma_{x_i,k^*})$. Here k^* is the key that generated the trace, i.e the correct key-guess key. Hence the expectation of $\dot{\rho}_k$ is the same expression but with the expectation of $E[l_i] = m_{x_i,k^*}$ replacing the trace. This can be expressed as:

$$
\begin{aligned}
E\left[\dot{\rho}_k(\mathbf{x},\mathbf{l})\right] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{\left(\mathrm{m}(x_i,k)-\frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)l_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)^2}}\right] \\
&= \frac{1}{N}\sum_{i=1}^{N}\frac{\left(\mathrm{m}(x_i,k)-\frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)E[l_i]}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)^2}} \\
&= \frac{1}{N\sigma_k}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)m_{x_i,k^*}.
\end{aligned}
$$

Where $\sigma_k = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{m}(x_i,k)-\frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k)\right)^2}$. Now that we have the expression for the expectation of a key-guess we can more easily calculate the covariance between two keys. Namely we use that for the covariance calculation we have the identity $\mathrm{Cov}[\dot{\rho}_{k_1}, \dot{\rho}_{k_2}] = E[\dot{\rho}_{k_1}\dot{\rho}_{k_2}] - E[\dot{\rho}_{k_1}]E[\dot{\rho}_{k_2}]$. In our leakage model

this becomes:

$$\begin{aligned}
&\mathrm{Cov}[\dot{\rho}_{k_1}, \dot{\rho}_{k_2}] = \\
&E\left[\frac{1}{N\sigma_{k_1}}\sum_{i=1}^{N}\big(\mathrm{m}(x_i,k_1) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k_1)\big)l_i \cdot \frac{1}{N\sigma_{k_2}}\sum_{i=1}^{N}\big(\mathrm{m}(x_i,k_2) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k_2)\big)l_i\right] \\
&- \left(\frac{1}{N\sigma_{k_1}}\sum_{i=1}^{N}\big(\mathrm{m}(x_i,k_1) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k_1)\big)E[l_i]\right) \\
&\quad\left(\frac{1}{N\sigma_{k_2}}\sum_{i=1}^{N}\big(\mathrm{m}(x_i,k_2) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k_2)\big)E[l_i]\right) \\
&= \frac{1}{N^2\sigma_{k_1}\sigma_{k_1}}\sum_{i=1}^{N}\big(\mathrm{m}(x_i,k_1) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k_1)\big) \\
&\quad\big(\mathrm{m}(x_i,k_2) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k_2)\big)(E[l_i^2] - E[l_i]E[l_i])
\end{aligned}$$

Now we can use the identity from 2.4 to get

$$= \frac{1}{N^2\sigma_{k_1}\sigma_{k_1}}\sum_{i=1}^{N}\big(\mathrm{m}(x_i,k_1) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k_1)\big)\big(\mathrm{m}(x_i,k_2) - \frac{1}{N}\sum_{j=1}^{N}\mathrm{m}(x_j,k_2)\big)\mathrm{Var}(l_i).$$

We know the variance of l_i, namely that $\mathrm{Var}[l_i] = \sigma^2_{x_i,k^*}$. Hence we are done with the calculation, and we get what we wanted. □

4. Higher order correlation distribution calculation

In the same manner we have calculated the distribution for a first order correlation distinguisher, we want to do it for a higher order distinguisher. One can see in the formulas that the formula for the leakage in quite similar except for the centered product combining function $\mathrm{C} : \mathcal{L} = \mathbb{R}^{d+1} \to \mathbb{R}$. Therefore in order to calculate the distribution of the distinguisher it will be useful to know the distribution of C.

LEMMA 4.4. *The distribution of the centered product $C(\mathcal{L}_{x,k^*})$ where $\mathcal{L}_{x,k^*} = (f_0(s_0) + n_0, f_1(s_1) + n_1), \ldots, f_d(s_d) + n_d)$, i.e the leakage generated by*

input x and key k^*, has distribution satisfying:

$$\mathrm{E}[C(\mathcal{L}_{x,k^*})] = \frac{1}{|\mathcal{S}|^d} \sum_{s_0 \in \mathcal{S}_0} \sum_{s_1 \in \mathcal{S}_1} \cdots \sum_{s_d \in \mathcal{S}_d} \prod_{j=0}^{d} \left(m_{j,s_j} - \mu_j\right) \tag{5}$$

$$\begin{aligned} \mathrm{Var}[C(\mathcal{L}_{x,k^*})] &= \prod_{j=0}^{d} \left(\mathrm{E}[(f_j(s_j) - \mu_j)^2] + \mathrm{E}\left[n_j^2\right]\right) \\ &- \prod_{j=0}^{d} \mathrm{E}[(f_j(s_j) - \mu_j)^2] + \mathrm{E}\left[\prod_{j=0}^{d} (f_j(s_j) - \mu_j)^2\right] - \mathrm{E}\left[C(\mathcal{L}_{x,k^*})\right]^2 \end{aligned} \tag{6}$$

Proof. In order to calculate the expected value of the centering product over the random variable $(\mathcal{L}_{x,k^*})$ we take the average of the leakage functions of all the signals $(s_0, s_1, \ldots, s_d) = \varphi(x, k^*)$:

$$\mathrm{E}[\mathrm{C}\left(f_0(s_0) + n_0, f_1(s_1) + n_1), \ldots, f_d(s_d) + n_d)\right] = \mathrm{E}\left[\prod_{j=0}^{d} \left(f_j(s_j) + n_j - \mu_j\right)\right]$$

Since all the noise variables are independent we can take the average of all signals:

$$= \frac{1}{|\mathcal{S}|^d} \sum_{s_0 \in \mathcal{S}_0} \sum_{s_1 \in \mathcal{S}_1} \cdots \sum_{s_d \in \mathcal{S}_d} \left(\prod_{j=0}^{d} \mathrm{E}\left[f_j(s_j) + n_j - \mu_j\right]\right)$$

we have that $f_j(s_j) = m_{j,s_j}$ and $\mathrm{E}[n_j] = 0 \Rightarrow$

$$\mathrm{E}[\mathrm{C}\left(\mathcal{L}_{x,k^*}\right)] = \frac{1}{|\mathcal{S}|^d} \sum_{s_0 \in \mathcal{S}_0} \sum_{s_1 \in \mathcal{S}_1} \cdots \sum_{s_d \in \mathcal{S}_d} \left(\prod_{j=0}^{d} \left(\mathrm{m}_{j,s_j} - \mu_j\right)\right)$$

In order to calculate the variance we use the identity $\mathrm{Var}\left[\mathrm{C}\left(\mathcal{L}_{x,k^*}\right)\right] = \mathrm{E}\left[\mathrm{C}\left(\mathcal{L}_{x,k^*}\right)^2\right] - \mathrm{E}\left[\mathrm{C}\left(\mathcal{L}_{x,k^*}\right)\right]^2$, as we already have the formula for $\mathrm{E}\left[\mathrm{C}\left(\mathcal{L}_{x,k^*}\right)\right]$, we show the calculation of $\mathrm{E}\left[\mathrm{C}\left(\mathcal{L}_{x,k^*}\right)^2\right]$:

$$\mathrm{E}[\mathrm{C}\left(f_0(s_0)+n_0, f_1(s_1)+n_1), \ldots, f_d(s_d)+n_d\right)^2] = \mathrm{E}\left[\prod_{j=0}^{d}\left(f_j(s_j)+n_j-\mu_j\right)^2\right]$$

$$= \mathrm{E}\left[\prod_{j=0}^{d}\left(f_j(s_j)^2 - 2f_j(s_j)\mu_j + \mu_j^2 + 2f_j(s_j)n_j - 2\mu_j n_j + n_j^2\right)\right],$$

$$\mathrm{E}[n_j] = 0 \text{ and } \mathrm{E}[n_i n_j] = \mathrm{E}[n_j]\mathrm{E}[n_i] \Rightarrow$$

$$= \mathrm{E}\left[\prod_{j=0}^{d}\left(f_j(s_j)^2 - 2f_j(s_j)\mu_j + \mu_j^2 + n_j^2\right)\right] = \mathrm{E}\left[\prod_{j=0}^{d}\left((f_j(s_j)-\mu_j)^2 + n_j^2\right)\right]$$

We want to get the expectation on the inside of the product instead of on the outside. For independent variables we have that the expectation of the product of variables, is equal to the product of the expectation. More formally $\mathrm{E}[X_1 X_2 \ldots X_n] = \mathrm{E}[X_1]\mathrm{E}[X_2]\ldots\mathrm{E}[X_n]$ when all the variables are mutually independent. In the leakage model we assumed all the noise variables n_j, representing the noise of the masks, were independent from each other. In addition they are also mutually independent to the leakage model $f_j(s_j)$. When introducing masking in 6, we also made the assumption that all the masks are mutually independent from each other.

This however has an important exception, the masks are all mutually independent except when they are all together. This fact creates and issue when we want to get the expectation inside the product, if we calculate the product alone one of the factors of the product will have all of the masks in the expression. We can show this as

$$\prod_{j=0}^{d}\left((f_j(s_j)-\mu_j)^2 + n_j^2\right) = \left(\prod_{j=0}^{d}(f_j(s_j)-\mu_j)^2 + \ldots + \prod_{j=0}^{d} n_j^2\right).$$

This is the only term that involves all the masks so it is also the only term where the variables are not mutually independent. Hence in order to simplify the expression we will add the expression $\prod_{j=0}^{d}\mathrm{E}[(f_j(s_j)-\mu_j)^2]$ inside the full product, then subtract it on the outside of the product. In doing this we all together add 0, we are however still left with the expectation of the product of all the masks,

which will need to be added as well. To make this clearer we can write is as:

$$\mathrm{E}\left[\prod_{j=0}^{d}\left((f_j(s_j)-\mu_j)^2+n_j^2\right)\right]=\mathrm{E}\left[\left(\prod_{j=0}^{d}(f_j(s_j)-\mu_j)^2+\ldots+\prod_{j=0}^{d}n_j^2\right)\right]$$

$$=\left(\mathrm{E}\left[\prod_{j=0}^{d}(f_j(s_j)-\mu_j)^2\right]+\ldots+\prod_{j=0}^{d}\mathrm{E}\left[n_j^2\right]\right)$$

$$+\prod_{j=0}^{d}\mathrm{E}[(f_j(s_j)-\mu_j)^2]-\prod_{j=0}^{d}\mathrm{E}[(f_j(s_j)-\mu_j)^2]$$

Now we can switch the expectation expressions

$$=\left(\prod_{j=0}^{d}\mathrm{E}[(f_j(s_j)-\mu_j)^2]+\ldots+\prod_{j=0}^{d}\mathrm{E}\left[n_j^2\right]\right)$$

$$-\prod_{j=0}^{d}\mathrm{E}[(f_j(s_j)-\mu_j)^2]+\mathrm{E}\left[\prod_{j=0}^{d}(f_j(s_j)-\mu_j)^2\right]$$

$$=\prod_{j=0}^{d}\left(\mathrm{E}[(f_j(s_j)-\mu_j)^2]+\mathrm{E}\left[n_j^2\right]\right)-\prod_{j=0}^{d}\mathrm{E}[(f_j(s_j)-\mu_j)^2]+\mathrm{E}\left[\prod_{j=0}^{d}(f_j(s_j)-\mu_j)^2\right]$$

□

We now have an expression for the distribution of the additive function of the higher order correlation distribution. This enables us to use proposition 3.13 in order to get the distribution of the full distinguisher.

COROLLARY 4.5. *The distribution of the higher order correlation coefficient* $(\dot{\rho}_k(\mathbf{x},\mathbf{l}))$*, where* σ_k *and* $\overline{m}_k$ *denotes the standard deviation and mean of* $m(\mathbf{x},k)$ *respectively, satisfies:*

$$\mathrm{E}[\dot{\rho}_k(\mathbf{x},\mathbf{l})]=\frac{1}{\sigma_k N}\sum_{i=0}^{N}\left(m(x_i,k)-\overline{m}_k\right)\mathrm{E}[C(\mathbf{l}_i)] \tag{7}$$

$$\mathrm{Cov}[\dot{\rho}_{k_1}(\mathbf{x},\mathbf{l}),\dot{\rho}_{k_2}(\mathbf{x},\mathbf{l})]$$

$$=\frac{1}{\sigma_{k_1}\sigma_{k_2}N^2}\sum_{i=0}^{N}\left(\left(m(x_i,k_1)-\overline{m}_{k_1}\right)\left(m(x_i,k_2)-\overline{m}_{k_2}\right)\mathrm{Var}[C(\mathcal{L}_{x_i,k^*})]\right) \tag{8}$$

PROOF. According to proposition 3.13, we can relate the distribution of the distinguisher to the full distribution. Hence we can use that:

$$\mathrm{E}[\dot{\rho}_k(\mathbf{x},\mathbf{l})] = \mathrm{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{\big(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\big)\mathrm{C}(\mathbf{l}_i)}{\sigma_k}\right] = \frac{1}{N\sigma_k}\sum_{i=1}^{N}\big(\mathrm{m}(x_i,k)-\overline{\mathrm{m}}_k\big)\mathrm{E}[\mathrm{C}(\mathbf{l}_i)]$$

The covariance calculation is a little more involved, but again using the fact from proposition 3.13, that for distinguisher d_k, $\mathrm{Cov}[d_{k_1}, d_{k_2}] = \frac{1}{N^2}\sum_{i=0}^{N}\mathrm{Cov}[g_{x_i,k_1}, g_{x_i,k_2}]$. For the correlation coefficient, this becomes:

$$\begin{aligned}
&\mathrm{Cov}[\dot{\rho}_{k_1}(\mathbf{x},\mathbf{l}),\dot{\rho}_{k_2}(\mathbf{x},\mathbf{l})]\\
&= \frac{1}{N^2}\sum_{i=0}^{N}\Big(\mathrm{Cov}[\big(\mathrm{m}(x_i,k_1)-\overline{\mathrm{m}}_{k_1}\big)\mathrm{C}(\mathbf{l}_i),\big(\mathrm{m}(x_i,k_2)-\overline{\mathrm{m}}_{k_2}\big)\mathrm{C}(\mathbf{l}_i)]\Big)\\
&= \frac{1}{N^2}\sum_{i=0}^{N}\big((\mathrm{m}(x_i,k_1)-\overline{\mathrm{m}}_{k_1})(\mathrm{m}(x_i,k_2)-\overline{\mathrm{m}}_{k_2})\mathrm{Var}[\mathrm{C}(\mathbf{l}_i)]\big)
\end{aligned}$$

We already calculated the expectation and variance of the combining function, so we are done with the distribution calculation. □

Now that we have calculated the distribution of both an masked an unmasked implementation of the correlation attack we will be able to estimate the success-rate given the number of traces. How this is done is described in chapter 7.

CHAPTER 5

A maximum likelihood distinguisher

A profiling attack, sometimes called a template attack, uses a maximum likelihood technique to score keys. Here we assume we have some probability distribution of the leakage according to each possible intermediate calculation value. For a plaintext leakage pair (x, l) and a key-guess k we can write the probability as $P[L = l|S = \varphi(x, k)]$. For plaintext-leakage vector pair $\mathbf{x} = (x_0, x_1, ..., x_N)$,$\mathbf{l} = (l_0, l_1, ..., l_N)$ we just find the product of the probability evaluations of the leakage points. Formally we can write this as

$$P[\mathcal{K} = k|(\mathbf{x}, \mathbf{l})] = \alpha \prod_{i=1}^{N} P[L = l_i|S = \varphi(x_i, k)]$$

An important distinction between the profiling distinguishers and the correlation distinguishers in the last chapter is, as the name implies, that in profiling attack we have a profile of how the distribution of the leakage will look like for each plaintext and key-guess. This profile is here assumed to be given, as we are in a theoretical setting. In reality it would require a profiling of the device, or a device similar to, that one tries to attack. This profiling involves estimating the distribution of each plaintext and key-guess, in order to have an associated pdf which again is used to calculate the above expression. Hence for the distribution calculation, the profile distributions will also be involved, as we will see in the next section.

1. Constructing an unmasked first order profiling distinguisher

In order to define the profile distinguisher we need a way to calculate the probability of seeing a leakage trace according to a key-guess. The specific evaluation of the pdf for a first order Gaussian distribution $\sim \mathcal{N}(m, \Sigma)$ can be written as:

$$\phi_{\Sigma,m}(l) = \frac{e^{-\frac{1}{2}(l-m)^T \Sigma^{-1}(x-m)}}{\sqrt{(2\pi)|\Sigma|}}$$

Usually we take the logarithm of this for computational reasons, this does not change the order of the size of the key-scores so it results in an equivalent distinguisher. Taking the logarithm also means that the product becomes a sum and the pdf evaluation that can be expressed as:

$$\begin{aligned}\mathcal{L}_k &= \frac{1}{N}\log(P[\mathcal{K}=k|(\mathbf{x},\mathbf{l})]/\alpha) \\ &= \frac{1}{2N}\sum_{i=1}^{N}\left(-\log((2\pi)|\widehat{\Sigma}_{x_i,k}|) - (l_i-\widehat{m}_{x_i,k})^T\widehat{\Sigma}^{-1}_{x_i,k}(l_i-\widehat{m}_{x_i,k})\right)\end{aligned}$$

Now that we have the expression for the score of a key-guess we can define a distinguisher by taking this score over all keys. Thus the profile distinguisher $\mathbf{d}_{\text{prof}} = (\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_{N_k})$. This distinguisher is not only additive but properly additive, as we can evaluate each single trace, plaintext pair separately and calculate the average. This can be written as:

$$\mathcal{L}_k = \frac{1}{N}\sum_{i=1}^{N} g_k(x_i, l_i)$$

$$\text{where } g_k(x_i,l_i) = -\log((2\pi)|\widehat{\Sigma}_{x_i,k}|) - (l_i-\widehat{m}_{x_i,k})^T\widehat{\Sigma}^{-1}_{x_i,k}(l_i-\widehat{m}_{x_i,k})$$

2. Constructing a masked higher order profiling distinguisher

Masking, as seen before is a countermeasure to side channel attacks, but the profiling distinguisher can also be modified in order to handle higher order leakage. As before the traces, i.e the leakage $\mathbf{l} = (\mathbf{l}_i)_i$, no longer consists of real number but vectors of real numbers $\mathbf{l}_i = (l_0, l_1, \ldots, l_d)$ for each $\mathbf{l}_i$. Similarly to the first order distinguisher we also need a model of expected leakage given the input x, k, i.e a pdf evaluation. With the Gaussian assumption of leakage the pdf evaluation of one higher order trace can be expressed as

$$p_s = \frac{1}{|\mathcal{S}|^d}\sum_{s_0\in\mathcal{S}_0}\sum_{s_1\in\mathcal{S}_1}\cdots\sum_{s_d\in\mathcal{S}_d}\left(\prod_{j=0}^{d}\phi_{m_{j,s_j},\Sigma_j}(l_j)\right).$$

Where $s_j = \varphi(x,k)$ is the signal from the j-th coming from the intermediate calculation, m_{j,s_j} the model-function for mask j with input s_j, and covariance matrix Σ_j for mask j. One can see here that the profiling step of getting the expectation and covariance for all inputs and masks is more extensive than the first order attack.

As in the previous section the distinguisher is defined as the log of the likelihood of getting the leakage vector with given input, the only difference is that instead of the standard pdf we use this new one for masked leakage vector. Since we only use the the single input and corresponding leakage this distinguisher is properly

additive as the first order version is. The distinguisher score for key guess k can be expressed as

$$\mathcal{L}_k = \frac{1}{N}\sum_{i=1}^{N} \log(p_{\varphi(x_i,k)}).$$

3. First order profile distribution calculation

Before calculating the expectation and covariance of the log-likelihood vector $\mathcal{L}_k$, we need to establish two norms:

$\|\cdot\|$ is the euclidean norm, i.e for a d-size vector X, $\|X\| = \sqrt{\sum_{i=1}^{d}(X_i)^2}$

$\|\cdot\|_{hs}$ is the Hilbert-Schmidt matrix norm, i.e for a $d \times d$ matrix A, $\|A\|_{hs} = \sqrt{\sum_{i=1}^{d}\sum_{j=1}^{d}(A_{i,j})^2}$

LEMMA 5.1. *X is a d-sized random vector with 0-mean, and $d \times d$ covariance matrix Σ. A_1, A_2 are both $d \times d$ matrices, and m_1, m_2 are two d-size vectors. The expectation and covariance matrix of the quadratic form $Q_j = (X + m_j)^T A_j^T A_j (X + m_j)$ satisfies:*

$$\mathrm{E}[Q_j] = \|A_j m_j\|^2 + Tr(A_j \Sigma A_j^T) \tag{9}$$

$$\mathrm{Cov}[Q_1, Q_2] = 2\left\|A_1 \Sigma A_2^T\right\|_{hs}^2 + m_1^T A_1^T A_1 \Sigma \tag{10}$$

We will first calculate the expectation then calculate the covariance.

PROOF. We start with $\mathrm{E}[Q_j]$, to calculate this we need to write $(X+m_j)^T A_j^T A_j((X+m_j)$ on a different form:

$$(X+m_j)^T A_j^T A_j (X+m_j) = (A_j(X+m_j))^T (A_j(X+m_j)) = \sum_{i=1}^{d}((A_j(X+m_j))_i)^2$$

Now we can use the formula $\mathrm{Var}(X) = \mathrm{E}[X^2] - \mathrm{E}[X]^2 \implies \mathrm{E}[X^2] = \mathrm{Var}(X) + \mathrm{E}[X]^2$. Hence we get:

$$\mathrm{E}[Q_j] = \sum_{i=1}^{d}\mathrm{E}[(A_j(X+m_j))_i)^2] = \sum_{i=1}^{d}\left(\mathrm{E}[(A_j(X+m_j))_i]^2 + \mathrm{Var}((A_j(X+m_j))_i)\right)$$

From this we can calculate the expected value, since X is a random vector with distribution $X \sim \mathcal{N}(0, \Sigma)$ we get that $A_j(X + m_j) \sim \mathcal{N}(A_j m_j, A_j \Sigma A_j^T)$. The variance of $(A_j(X + m_j))_i$ is just the $i - th$ diagonal of $A_j \Sigma A_j^T$. So we end up with:

$$\sum_{i=1}^{d} \left(\mathrm{E}[A_j(X + m_j)]^2 + \mathrm{Var}((A_j(X + m_j))_i) \right) =$$
$$\sum_{i=1}^{d} \left(((A_j m_j)_i)^2 + (A_j \Sigma A_j^T)_{i,i} \right) = \|A_j m_j\|^2 + \mathrm{Tr}(A_j \Sigma A_j^T)$$

Hence we have that the expectation $\mathrm{E}[Q_j] = \|A_j m_j\|^2 + \mathrm{Tr}(A_j \Sigma A_j^T)$, and the first part of the lemma is finished.

Now we want to calculate the covariance matrix $\mathrm{Cov}[Q_1, Q_2]$, in order to do this we again need to express Q_j on a different form:

$$\begin{aligned}(A_j(X + m_j))^T (A_j(X + m_j)) &= (A_j X + A_j m_j)^T (A_j X + A_j m_j) \\ &= \sum_{i=1}^{d} (A_{j_{*,i}} X + A_{j_{*,i}} m_j)^T (A_{j_{*,i}} X + A_{j_{*,i}} m_j) \\ &= \sum_{i=1}^{d} (X^T A_{j_{*,i}}^T + m_j^T A_{j_{*,i}}^T)(A_{j_{*,i}} X + A_{j_{*,i}} m_j)\end{aligned}$$
$$= \sum_{i=1}^{d} \left(X^T A_{j_{*,i}}^T A_{j_{*,i}} X + X^T A_{j_{*,i}}^T A_{j_{*,i}} m_j + m_j^T A_{j_{*,i}}^T A_{j_{*,i}} X + m_j^T A_{j_{*,i}}^T A_{j_{*,i}} m_j \right)$$

We can see that $X^T A_{j_{*,i}}^T A_{j_{*,i}} m_j = m_j^T A_{j_{*,i}}^T A_{j_{*,i}} X$, as the different coordinates of X_i and m_{j_i} coincide in the sum. We can hence write Q_j on the form:

$$\begin{aligned}Q_j =& \sum_{i=1}^{d} (X^T A_{j_{*,i}}^T A_{j_{*,i}} X + m_j^T A_{j_{*,i}}^T A_{j_{*,i}} m_j + 2 m_j^T A_{j_{*,i}}^T A_{j_{*,i}} X) \\ =& (A_j X)^T (A_j X) + (A_j m_j)^T (A_j m_j) + 2(m_j^T A_j^T A_j X)\end{aligned}$$

So to calculate the covariance now we need to find:

$$\begin{aligned}\mathrm{Cov}[Q_1, Q_2] = \mathrm{Cov}[&(A_1X)^T(A_1X) + (A_1m_1)^T(A_1m_1) + 2(m_1^TA_1^TA_1X),\\ &(A_2X)^T(A_2X) + (A_2m_2)^T(A_2m_2) + 2(m_2^TA_2^TA_2X)]\end{aligned}$$

Adding a constant to the covariance does not change the result so we can disregard $(A_1m_1)^T(A_1m_1)$ in the calculation, from the biliniarity of covarience we get:

$$\begin{aligned}&\mathrm{Cov}[(A_1X)^T(A_1X) + 2(m_1^TA_1^TA_1X), (A_2X)^T(A_2X) + 2(m_2^TA_2^TA_2X)] =\\ &\mathrm{Cov}[(A_1X)^T(A_1X), (A_2X)^T(A_2X)] + \mathrm{Cov}[(A_1X)^T(A_1X), 2(m_2^TA_2^TA_2X)]+\\ &\mathrm{Cov}[2(m_1^TA_1^TA_1X), (A_2X)^T(A_2X)] + \mathrm{Cov}[2(m_1^TA_1^TA_1X), 2(m_2^TA_2^TA_2X)]\end{aligned}$$

Factoring out constants the expression simplifies to:

$$\begin{aligned}&\mathrm{Cov}[Q_1, Q_2] =\\ &\mathrm{Cov}[(A_1X)^T(A_1X), (A_2X)^T(A_2X)] + 2\mathrm{Cov}[(A_1X)^T(A_1X), (m_2^TA_2^TA_2X)]\\ &+2\mathrm{Cov}[(m_1^TA_1^TA_1X), (A_2X)^T(A_2X)] + 4\mathrm{Cov}[(m_1^TA_1^TA_1X), (m_2^TA_2^TA_2X)]\end{aligned}$$

We have now simplified the expression and what is left to do is showing that the four covariance expressions satisfy:

$$\mathrm{Cov}[(A_1X)^T(A_1X), (A_2X)^T(A_2X)] = 2\left\|A_1\Sigma A_2^T\right\|_{hs}^2 \tag{11}$$

$$\mathrm{Cov}[(A_1X)^T(A_1X), (m_2^TA_2^TA_2X)] = 0 \tag{12}$$

$$\mathrm{Cov}[(m_1^TA_1^TA_1X), (A_2X)^T(A_2X)] = 0 \tag{13}$$

$$\mathrm{Cov}[(m_1^TA_1^TA_1X), (m_2^TA_2^TA_2X)] = m_1^TA_1^TA_1\Sigma A_2^TA_2m_2 \tag{14}$$

Equation (14) follows from standard covariance rules. We begin with equation (11). From before we have that $(A_jX)^T(A_jX)$ can be written as $((A_jX)_i)^2$ summed over i, then by bilinearity we can express the covariance as:

$$\mathrm{Cov}[(A_1X)^T(A_1X), (A_2X)^T(A_2X)] = \sum_{i=1}^{d}\sum_{j=1}^{d}\mathrm{Cov}[((A_1X)_i)^2, ((A_2X)_j)^2]$$

With the standard formula for covariance $\mathrm{Cov}[Y,Z] = \mathrm{E}[YZ^T] - \mathrm{E}[Y]\mathrm{E}[Z]$, and the fact that $(A_2X)^T(A_2X)$ is symmetric, we can write this as:

$$\sum_{i=1}^{d}\sum_{j=1}^{d}\left(\mathrm{E}[((A_1X)_i)^2((A_2X)_j)^2] - \mathrm{E}[((A_1X)_i)^2]\mathrm{E}[((A_2X)_j)^2]\right)$$

To calculate the first part of the sum we can use Isserlis theorem 2.9, to get that:

$$\begin{aligned}
\mathrm{E}[((A_1X)_i)^2((A_2X)_j)^2] &= \mathrm{E}[((A_1X)_i)^2]\mathrm{E}[((A_2X)_j)^2] \\
&\quad + \mathrm{E}[(A_1X)_i(A_2X)_j]^2 + \mathrm{E}[(A_2X)_j(A_1X)_i]^2 \\
&= \mathrm{E}[((A_1X)_i)^2]\mathrm{E}[((A_2X)_j)^2 + 2\mathrm{E}[(A_1X)_i(A_2X)_j]^2 \\
&= \mathrm{E}[((A_1X)_i)^2]\mathrm{E}[((A_2X)_j)^2 + 2\big(\mathrm{Cov}[(A_1X)_i,(A_2X)_j] + \mathrm{E}[(A_1X)_i]\mathrm{E}[(A_2X)_j]\big)^2 \\
&= \mathrm{E}[((A_1X)_i)^2]\mathrm{E}[((A_2X)_j)^2 + 2\mathrm{Cov}[(A_1X)_i,(A_2X)_j]^2
\end{aligned}$$

Looking at back the sum we can see that the first term cancels with the last one in the original sum, hence:

$$\mathrm{Cov}[(A_1X)^T(A_1X),(A_2X)^T(A_2X)] = \sum_{i=1,j=1}^{d,d} 2\mathrm{Cov}[(A_1X)_i,(A_2X)_j]^2 = 2\left\|A_1\Sigma A_2^T\right\|_{hs}^2$$

Calculating equation (12) and (13) is easier as:

$$\mathrm{Cov}[(A_1X)^T(A_1X),(m_2^TA_2^TA_2X)] = \sum_{i=1}^{d}\mathrm{Cov}[((A_1X)_i)^2,(m_2^TA_2^TA_2X)]$$

By using the formula for covariance:

$$\mathrm{Cov}[((A_1X)_i)^2,(m_2^TA_2^TA_2X)] = \mathrm{E}[((A_1X)_i)^2(m_2^TA_2^TA_2X)] - \mathrm{E}[((A_1X)_i)^2]\mathrm{E}[(m_2^TA_2^TA_2X)]$$

As $\mathrm{E}(X) = 0$, and Isserlis theorem for odd cases 2.10 says that the expectation is 0, we have that this sum is 0. Similarly the covariance equation (13) is also 0.

□

Now that we have an expression for expectation and covariance of quadratic forms from lemma 5.1 we want to use this to find the expectation and covariance of the log-likelihood calculation.

PROPOSITION 5.2. *The expectation of the log-likelihood calculation* $\mathcal{L}_k = \frac{1}{2N}\sum_{i=1}^{N}\Big(-\log((2\pi)^d|\widehat{\Sigma}_{x_i,k}|) - (l_i - \widehat{m}_{x_i,k})^T\widehat{\Sigma}^{-1}_{x_i,k}(l_i - \widehat{m}_{x_i,k})\Big)$ *satisfies:*

$$\mathrm{E}[\mathcal{L}_k] = \frac{1}{2}\sum_{x\in\mathcal{X}}\tau_x\Big(-\log((2\pi)^d|\widehat{\Sigma}_{x_i,k}|) - \Big\|(\widehat{\Sigma}^{-1_L}_{x_i,k})(m_{i,k^*} - \widehat{m}_{x_i,k})\Big\|^2 - Tr((\widehat{\Sigma}^{-1_L}_{x_i,k})\Sigma(\widehat{\Sigma}^{-1_L}_{x_i,k})^T)\Big) \tag{15}$$

And for two keys $\mathcal{L}_{k_1}, \mathcal{L}_{k_2}$*, the covariance satisfies:*

$$\mathrm{Cov}[\mathcal{L}_{k_1}, \mathcal{L}_{k_2}] = \frac{1}{N}\sum_{i=1}^{N}\tau_x\Big(\frac{1}{2}\Big\|(\widehat{\Sigma}^{-1_L}_{x_i,k_1})\Sigma(\widehat{\Sigma}^{-1_L}_{x_i,k_2})^T\Big\|_{hs} + (m_{i,k^*} - \widehat{m}_{x_i,k_1})^T(\widehat{\Sigma}^{-1}_{x_i,k_1})\Sigma(\widehat{\Sigma}^{-1}_{x_i,k_2})(m_{i,k^*} - \widehat{m}_{x_i,k_2})\Big) \tag{16}$$

The expectation is calculated first, then the covariance.

PROOF. From 5.1, we have that a quadratic form $Q_j = (X + m_j)^T A_j^T A_j (X + m_j)$ has covariance, $\mathrm{Cov}[Q_1, Q_2] = 2\big\|A_1\Sigma A_2^T\big\|_{hs}^2 + 4m_1^T A_1^T A_1 \Sigma A_2^T A_2 m_2$, and expectation $\mathrm{E}(Q_j) = \|A_j m_j\|^2 + \mathrm{Tr}(A_j\Sigma A_j^T)$.

We use this to calculate the expectation $\mathrm{E}[\mathcal{L}_k]$:

$$\begin{aligned}\mathrm{E}[\mathcal{L}_k] =& \mathrm{E}[\frac{1}{2N}\sum_{i=1}^{N}\Big(-\log((2\pi)^d|\widehat{\Sigma}_{x_i,k}|) - (l_i - \widehat{m}_{x,k})^T\widehat{\Sigma}^{-1}_{x_i,k}(l_i - \widehat{m}_{x_i,k})\Big)]\\ =& \frac{1}{2N}\sum_{i=1}^{N}\mathrm{E}[-\log((2\pi)^d|\widehat{\Sigma}_{x_i,k}|] - \mathrm{E}[(l_i - \widehat{m}_{x,k})^T\widehat{\Sigma}^{-1}_{x_i,k}(l_i - \widehat{m}_{x_i,k})]\end{aligned}$$

The first part of the sum has no random variables so the expectation is just itself. The second sum is almost as the setup we have in lemma 5.1, except that l_i has expectation m_{i,k^*}, we can solve this by looking at the leakage differently, specifically $l_i = x_i + m_{i,k^*}$, where x_i is a random vector with expectation 0. Hence the vector $(l_i - \widehat{m}_{x,k})$ is translated to $(x_i + m_{i,k^*} - \widehat{m}_{x_i,k})$. In addition to this we need to have $\widehat{\Sigma}^{-1}_{x_i,k}$ on the form of the product of two matrices as in lemma 5.1, this is the Cholesky decomposition and will be denoted as $\widehat{\Sigma}^{-1}_{x_i,k} = (\widehat{\Sigma}^{-1_L}_{x_i,k})(\widehat{\Sigma}^{-1_L}_{x_i,k})^T$

$$\mathrm{E}[\mathcal{L}_k] = \frac{1}{2N}\sum_{i=1}^{N}\Big(-\log((2\pi)^T|\widehat{\Sigma}_{x_i,k}|) - \Big\|(\widehat{\Sigma}^{-1_L}_{x_i,k})(m_{i,k^*} - \widehat{m}_{x_i,k})\Big\|^2 - Tr((\widehat{\Sigma}^{-1_L}_{x_i,k})\Sigma(\widehat{\Sigma}^{-1_L}_{x_i,k})^T)\Big)$$

Now we switch index from over observation i to ratio of inputs x with τ_x, doing this over the sum we get an extra factor N:

$$\mathrm{E}[\mathcal{L}_k] = \tfrac{1}{2}\sum_{x\in\mathcal{X}}\tau_x\left(-\log((2\pi)^T|\widehat{\Sigma}_{x_i,k}|) - \left\|(\widehat{\Sigma}^{-1_L}_{x_i,k})(m_{i,k^*}-\widehat{m}_{x_i,k})\right\|^2 - Tr((\widehat{\Sigma}^{-1_L}_{x_i,k})\Sigma(\widehat{\Sigma}^{-1_L}_{x_i,k})^T\right)$$

When calculating the covariance we can use the same argument as the expectation, the first part of the sum does not have any random variables in it, and can hence be disregarded in the covariance calculation. With the same arguments about the expectation of l_i, and the Cholesky decomposition we get covariance on the form:

$$\begin{aligned}
&\mathrm{Cov}[\mathcal{L}_{k_1},\mathcal{L}_{k_2}] = \\
&\mathrm{Cov}[\frac{1}{2N}\sum_{i=1}^{N}\left(-(l_i-\widehat{m}_{x,k_1})^T\widehat{\Sigma}^{-1}_{x_i,k_1}(l_i-\widehat{m}_{x_i,k_1})\right),\frac{1}{2N}\sum_{i=1}^{N}\left(-(l_i-\widehat{m}_{x,k_2})^T\widehat{\Sigma}^{-1}_{x_i,k_2}(l_i-\widehat{m}_{x_i,k_2})\right)] \\
&= \frac{1}{4N^2}\sum_{i=1}^{N}\Big(2\left\|(\widehat{\Sigma}^{-1_L}_{x_i,k_1})\Sigma(\widehat{\Sigma}^{-1_L}_{x_i,k_2})^T\right\|_{hs} \\
&+ 4(m_{i,k^*}-\widehat{m}_{x_i,k_1})^T(\widehat{\Sigma}^{-1_L}_{x_i,k_1})^T(\widehat{\Sigma}^{-1_L}_{x_i,k_1})\Sigma(\widehat{\Sigma}^{-1_L}_{x_i,k_2})^T(\widehat{\Sigma}^{-1_L}_{x_i,k_2})(m_{i,k^*}-\widehat{m}_{x_i,k_2})\Big)
\end{aligned}$$

Switching to the ratio of x and taking out a factor 4 from the sum, we finally get the result:

$$\begin{aligned}
&\mathrm{Cov}[\mathcal{L}_{k_1},\mathcal{L}_{k_2}] = \\
&\frac{1}{N}\sum_{i=1}^{N}\tau_x\Big(\frac{1}{2}\left\|(\widehat{\Sigma}^{-1_L}_{x_i,k_1})\Sigma(\widehat{\Sigma}^{-1_L}_{x_i,k_2})^T\right\|_{hs} + (m_{i,k^*}-\widehat{m}_{x_i,k_1})^T(\widehat{\Sigma}^{-1}_{x_i,k_1})\Sigma(\widehat{\Sigma}^{-1}_{x_i,k_2})(m_{i,k^*}-\widehat{m}_{x_i,k_2})\Big)
\end{aligned}$$

□

4. Higher order profile distribution calculation

The technique to find the success-rate of the higher order profiling distinguisher is the same as in the first order one. However there is one fact that complicates the calculation. In the first order setting taking the log of the pdf of the model simplifies the expression somewhat. That is not the case in the higher order setting where the $\phi_{m_{j,s_j},\Sigma_j}$ is preceded by a sum over all the masks, hence not making the log expression simpler. This leads to the calculation of the expectation and the covariance somewhat more complicated. In order to make the calculation easier it is useful to find the expectation of the additive part of the function, in this higher order case that is $\log(p_s)$.

To help with the expectation and covariance calculation we introduce the two functions, namely

$$\lambda(s_1, s_2) := \int_{\mathbf{l}\in\mathcal{L}} \log(p_{s_1}(\mathbf{l}))p_{s_2}(\mathbf{l})\, d\mathbf{l}, \tag{17}$$

$$\psi(s_1, s_2, s_3) := \int_{\mathbf{l}\in\mathcal{L}} \log(p_{s_1}(\mathbf{l}))\log(p_{s_2}(\mathbf{l}))p_{s_3}(\mathbf{l})\, d\mathbf{l}. \tag{18}$$

We can relate this function to the expectation of $g_s(\mathbf{l})$ for $\mathbf{l}$ generated by $s = \varphi(x, k^*)$:

LEMMA 5.3. *For trace* $\mathbf{l}$ *generated by signal* $s = \varphi(x, k^*)$ *we can describe the expectation of the keyscores as:*

$$\mathrm{E}[g_{x,k}(\mathbf{l})] = \lambda(\varphi(x, k), \varphi(x, k^*))$$
$$\mathrm{E}[g_{x,k_1}(\mathbf{l})g_{x,k_2}(\mathbf{l})] = \psi(\varphi(x, k_1), \varphi(x, k_2), \varphi(x, k^*))$$

PROOF. The result is by definition as the expected value $\mathrm{E}[X]$ of some random variable X is integral of all the values X takes multiplied with the chance it occurs. We already have an expression for the pdf of leakage generated by the correct key k^* namely $p_{\varphi(x,k^*)}$ for given input x. The expected value we want to calculate is of one key-score $g_{x,k}(\mathbf{l})$, and two key-scores multiplied, $g_{x,k_1}(\mathbf{l})g_{x,k_2}(\mathbf{l})$, for given keys k, k_1, k_2. Hence the expected value of these two expressions are just the value of the function, i.e $g_{x,k}(\mathbf{l})$ and $g_{x,k_1}(\mathbf{l})g_{x,k_2}(\mathbf{l})$, multiplied with the probability of that leakage happening, which is $p_{\varphi(x,k^*)}$. Then we integrate over all signals $\mathbf{l}$ and we get the expected value of said expressions. □

Now that the expectation is described for $g_{x_i,k}(\mathbf{l})$ the expectation and covariance of $\mathcal{L}_k$ is more straightforward.

COROLLARY 5.4. *Let key* $k^* \in \mathcal{K}$*, input* $\mathbf{x} = (x_0, x_1, \ldots, x_N)$*, and leakage* $\mathbf{l}_i \leftarrow^{\$} \mathbf{L}_{x_i,k^*}$ *for each* x_i*. Then the key-guess* $\mathcal{L}$ *has distribution satisfying:*

$$\mathrm{E}[\mathcal{L}_k] = \frac{1}{N}\sum_{i=0}^{N} \lambda\left(\varphi(x_i, k), \varphi(x_i, k^*)\right)$$

$$\mathrm{Cov}[\mathcal{L}_{k_1}, \mathcal{L}_{k_2}] =$$

$$\frac{1}{N^2}\sum_{i=0}^{N} \psi(\varphi(x, k_1), \varphi(x, k_2), \varphi(x, k^*)) - \lambda\left(\varphi(x_i, k_1), \varphi(x_i, k^*)\right)\lambda\left(\varphi(x_i, k_2), \varphi(x_i, k^*)\right)$$

PROOF. The expectation calculation is straightforward as $\mathrm{E}[\mathcal{L}_k] = \frac{1}{N}\sum_{i=0}^{N}\mathrm{E}[g_{x_i,k}] = \frac{1}{N}\sum_{i=0}^{N}\lambda\left(\varphi(x_i, k), \varphi(x_i, k^*)\right)$. The last equality is from lemma 5.3.

For the covariance calculation we use proposition 3.13 to relate the additive function to the distinguisher, then we can use the same identity that we have used before, namely that $\mathrm{Cov}[X, Y] = \mathrm{E}[XY] - \mathrm{E}[X]\mathrm{E}[Y]$, hence we get:

$$\begin{aligned}\mathrm{Cov}[\mathcal{L}_{k_1}, \mathcal{L}_{k_2}] &= \frac{1}{N^2}\sum_{i=0}^{N}\left(\mathrm{Cov}[g_{x,k_1}(\mathbf{l}), g_{x,k_2}(\mathbf{l})]\right) \\ &= \frac{1}{N^2}\sum_{i=0}^{N}\left(\mathrm{E}[g_{x_i,k_1}(\mathbf{l}_i)g_{x_i,k_2}(\mathbf{l}_i)] - (\mathrm{E}[g_{x,k_1}(\mathbf{l})]\mathrm{E}[g_{x,k_2}(\mathbf{l})])\right)\end{aligned}$$

$$= \frac{1}{N^2}\sum_{i=0}^{N}\psi(\varphi(x,k_1), \varphi(x,k_2), \varphi(x,k^*)) - \lambda\left(\varphi(x_i,k_1), \varphi(x_i,k^*)\right)\lambda\left(\varphi(x_i,k_2), \varphi(x_i,k^*)\right)$$

□

As in chapter 4 the calculation of the distribution of the masked and unmasked distinguisher are done. This enables us to calculate the success-rate, which we will see in chapter 7.

CHAPTER 6

Confusion analysis

In the previous chapters we have explored ways to build and describe the distribution of general side channel attack techniques. In these chapters we build a distinguisher which takes in an intermediate value in the cryptographic algorithm, and depending on the noise parameters of some function of this intermediate value, and how many number of such traces one has, a distribution is assigned. However what this method does not show, is the vulnerability of the specific cryptographic implementation. In other words, are there cryptosystems that are particularly vulnerable to side channel attacks compared to others? We will explore this by expanding the study of intermediate value function, namely how different the intermediate value looks for different key guesses.

1. The Confusion coefficient

The intermediate value function is a function $\varphi : \mathcal{X} \times \mathcal{K} \to \mathcal{S}$, dependent on the cryptosystem. When leakage was modelled we say assume we receive a noise function of this variable. Namely that for leakage point $l_i = f(\varphi(x_i, k))$ if we disregard the noise. With this definition of how leakage looks like theoretically, the confusion coefficient between two keys is defined as:

DEFINITION 6.1. For keys $k_i, k_j \in \mathcal{K}$, for intermediate value function φ, and leakage model f the *confusion coefficient* κ is defined as:

$$\kappa(k_i, k_j) = \frac{1}{N_x} \sum_{x=0}^{N_x} (f(\varphi(x, k_i) - f(\varphi(x, k_j)))^2$$

N_x is the size of the plaintext space, i.e $N_x = |\mathcal{X}|$. More descriptively, the confusion coefficient measures how different the output of the model leakage is for the same plaintext for two different keys. If the model leakage is a bit the confusion coefficient becomes $\Pr[f(\varphi(x, k_1)) \neq f(\varphi(x, k_2)]$ for a random input x, as the output is either 1 or 0. As mentioned an intuitive understanding of the the confusion coefficient is that is measures how different the theoretical leakage

looks when its generated by different keys. This enables us to get a sense of how easy side channel analysis is with regards to the cryptographic algorithm.

In order to calculate the distribution of an attack involving the confusion coefficient we also need a three way confusion coefficient defined as:

DEFINITION 6.2. For keys $k_h, k_i, k_j \in \mathcal{K}$, for intermediate value function φ, and leakage model f the *confusion coefficient* $\tilde{\kappa}$ is defined as:

$$\tilde{\kappa}(k_h, k_i, k_j) = \frac{1}{N_x} \sum_{x=0}^{N_x} (f(\varphi(x, k_h) - f(\varphi(x, k_i))(f(\varphi(x, k_h) - f(\varphi(x, k_j))$$

We also need

$$\tilde{\kappa}^*(k_h, k_i, k_j) = \frac{1}{N_x} \sum_{x=0}^{N_x} \big(f(\varphi(x, k_h) - f(\varphi(x, k_i))^2 (f(\varphi(x, k_h) - f(\varphi(x, k_j)\big)^2$$

Lastly we need

$$\hat{\kappa}(k_h, k_i, k_j) = \frac{1}{N_x} \sum_{x=0}^{N_x} (f(\varphi(x, k_h)) - \mathrm{E}[f(\varphi(x, k_h))]\big)\big(f(\varphi(x, k_h)) - f(\varphi(x, k_i)))\big) \\ (f(\varphi(x, k_h)) - f(\varphi(x, k_j)))\big)$$

We write

$$\boldsymbol{\kappa}_{k_c} = (\kappa(k_c, k_1), \kappa(k_c, k_2), \ldots, \kappa(k_c, k_{c-1}), \kappa(k_c, k_{c+1}), \ldots, \kappa(k_c, k_{N_k}))$$

DEFINITION 6.3. Similarly we define the $(N_k - 1 \times N_k - 1)$-matrix for key c as

$$\boldsymbol{K}_{k_c}[i, j] = \begin{cases} \tilde{\kappa}(k_c, k_i, k_j) & \text{if } i \neq j \\ \kappa(k_c, k_i) & \text{if } i = j \end{cases}$$

DEFINITION 6.4. Again similarly we define the $(N_k - 1 \times N_k - 1)$-matrix for key c as

$$\boldsymbol{K}^*_{k_c}[i, j] = \begin{cases} \tilde{\kappa}^*(k_c, k_i, k_j) & \text{if } i \neq j \\ \kappa(k_c, k_i) & \text{if } i = j \end{cases}$$

2. Confusion analysis or model functions

If we take the confusion coefficient over all possible key-pairs then Fei et al. [5] claims that we can relate the distribution to how vulnerable that leakage model is to side channel attacks. One can think of this distribution as how different leakage will look like for different keys. This again describes how easy it is to distinguish keys, namely if the confusion coefficient usually has big values, then leakage generated by one key will probably look quite different than leakage

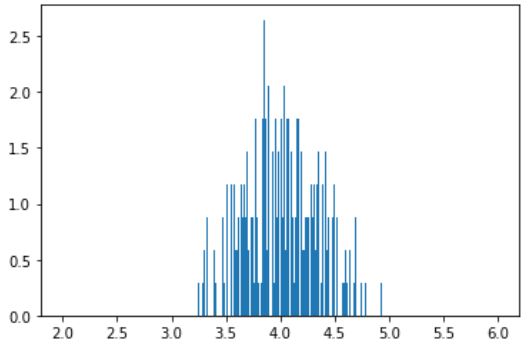


FIGURE 1. A graph showing the byte wise confusion analysis of AES-128. This distribution is reached by calculating the average of all squared differences of the hamming weight of the S-box output for the same plaintext but different keys.

generated by a different key. This would imply that a good cryptosystem for countering side channel attacks should have confusion coefficients as close to zero as possible. Unfortunately it is not that simple, if $(f(\varphi(x,k_1) - f(\varphi(x,k_2)))^2 = 0 \Rightarrow (f(\varphi(x,k_1) = f(\varphi(x,k_2))$. Although this does not directly mean that the two ciphertext outputted are the same, it does have a much bigger probability of being so. Hence if all the confusion coefficient are 0, we loose a lot of the diffusion of the algorithm. Namely that different keys give unrelated plaintext ciphertext pairs. So we have a trade-off between having sufficient diffusion and keys not being easily differentiated in a side channel setting.

REMARK 6.5. The optimal confusion coefficient is when the two variables $(f(\varphi(x,k_1))$ and $f(\varphi(x,k_2))$ look uncorrelated. In the case of AES-128 this will be 4 in the byte model, as this is where the confusion coefficient looks like the difference of two uncorrelated byte hamming weights.

2.1. Analysing bit-wise or byte-wise. When doing confusion analysis we can either be in a lekage model where we attempt to model bit-wise or byte-wise. In terms of the confusion coefficient the bit-wise formulas end up being somewhat simpler, as it is just measuring whether the intermediate value is different or the same. In this paper we will follow the leakage model where the intermediate value is a byte. An interesting example from the bit-analysis though, is the confusion analysis of AES-128.

EXAMPLE 6.6. When looking at the confusion coefficient bit-wise one have to choose which one of the 8 bits of the intermediate value output of AES-128 to

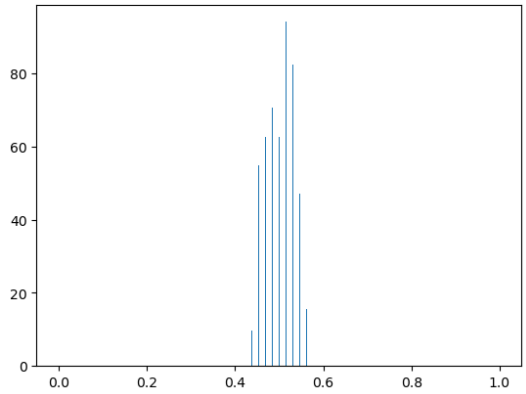


FIGURE 2. A graph showing the bit wise confusion analysis of AES-128, in this case we have to choose what bit to calculate. In this case it is the first bit, but an interesting fact is that it has the same distribution for all 8 bits.

calculate the confusion coefficient for. An interesting attribute of AES-128 is that it does not matter, the distribution of the confusion coefficients are identical for all the bits in AES-128, and has the distribution of figure 2.

3. Leakage model for confusion analysis

When studying leakage of a general cipher we have used the leakage model $l_i = f(\varphi(x_i, k)) + n_i$. This is however not the only setting one can view leakage. In the paper we take the confusion analysis theory from[5], they originally use a seemingly more complicated leakage model. This model involves an ϵ factor of the intermediate calculation, and a constant factor c, hence a leakage trace is modeled as $l_i = \epsilon f(\varphi(x_i, k)) + c + n_i$. This model can be seen as close to what one would actually measure on a chip. We can however use the simpler model without ϵ and c without loosing generality of the success-rate estimation. This is due to fact that the constant c does nothing to the success-rate estimation and the ϵ can be related to the standard deviation of the noise. Intuitively this can be done by having a bigger standard deviation in the simplified model if the ϵ is small, and the opposite in the other case. Hence in this thesis we will continue to use the simpler leakage model.

4. Using the confusion coefficient in the distribution calculation

As in the standard attacks we will also define a maximum likelihood and a correlation based distinguisher. Similarly we will calculate the distribution of these

distinguishers. The difference is that in order to incorporate the confusion coefficient we have to get the distinguisher on a form where we have the difference of two keys. This also changes how the distribution is calculated.

4.1. A maximum likelihood distinguisher with confusion analysis. As in the profiling section we will now look at the distinguisher where we have a probability associated with each input x and leakage point l. In other words we have a probability distribution $P[L = l_i|S = f(\varphi(x_i, k))]$. As before we also take the logarithm of this distribution, with results in the probability evaluation of multiple traces becoming the sum of the logarithm of the pdf evaluation of those traces.

As in the correlation confusion distribution we need to take the difference of the distinguisher score to relate it to the confusion coefficient. Hence we get a 255-dimensional Gaussian distribution, where one takes the difference of the correct key-guess score d_c, and all other scores d_k. Here we do the distribution calculation of one key-score difference for the correct key c and a general key-guess k, and the full distribution is the multi dimensional Gaussian distribution for all non-correct key-guessed k.

THEOREM 6.7. *The $(N_k - 1)$-score vector $(d_{k^*} - d_{k_g})_{k_g \in \mathcal{K}}$ for correct key k^* and key-guesses k_g satisfy the following distribution:*

$$\mu_{k^*} = \frac{1}{2\sigma^2}\boldsymbol{\kappa}_{k^*} \tag{19}$$

$$\Sigma_{k^*} = \frac{1}{N}\left(\frac{1}{\sigma^2}\boldsymbol{K}_{k^*} + \frac{1}{4\sigma^4}(\boldsymbol{K}_{k^*} - \boldsymbol{\kappa}_{k^*}\boldsymbol{\kappa}_{k^*}^T)\right) \tag{20}$$

PROOF. The value $d_{k^*} - d_{k_g}$ is the difference of the logarithm of the pdf evaluation in our model. For a key-guess k_g and trace l with related plaintext x, the log-pdf evaluation can be expressed in our model as:

$$\log(P[L = l|S = f(\varphi(x, k_g))]) = \log(\frac{1}{\sqrt{2\pi}\sigma}) - \frac{(l - f(\varphi(x, k_g)))^2}{2\sigma^2}$$

The score for several traces are just the sum of the single scores, hence for the difference of scores between $d_{k^*} - d_g$ for correct key-guess k^* becomes

$$\log\left(\prod_{i=1}^{N}(P[L = l_i|S = f(\varphi(x_i, k^*))])\right) - \log\left(\prod_{i=1}^{N}(P[L = l_i|S = f(\varphi(x_i, k_g))])\right)$$

$$= \log(\frac{1}{\sqrt{2\pi}\sigma}) - \log(\frac{1}{\sqrt{2\pi}\sigma}) - \sum_{i=1}^{N}\frac{(l_i - f(\varphi(x_i, k_g)))^2}{2\sigma^2} + \sum_{i=1}^{N}\frac{(l_i - f(\varphi(x_i, k^*)))^2}{2\sigma^2}$$

$$= \frac{1}{2\sigma^2}\sum_{i=1}^{N}(l_i - f(\varphi(x_i, k_g)))^2 - (l_i - f(\varphi(x_i, k^*)))^2$$

We have that $l_i = f(\varphi(x_i, k^*)) + n_i$ where k^* is the key that generated the leakage

$$= \frac{1}{2\sigma^2}\sum_{i=1}^{N}(f(\varphi(x_i, k^*)) + n_i - f(\varphi(x_i, k_g)))^2 - (f(\varphi(x_i, k^*)) + n_i - f(\varphi(x_i, k^*)))^2$$

$$= \frac{1}{2\sigma^2}\sum_{i=1}^{N}(f(\varphi(x_i, k^*)) + n_i - f(\varphi(x_i, k_g)))^2 - n_i^2$$

$$= \frac{1}{2\sigma^2}\sum_{i=1}^{N}\left((f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))^2 + 2(f(\varphi(x_i, k^*))) - f(\varphi(x_i, k_g))n_i\right)$$

This expression is what we are finding the distribution for as it is simpler calculation wise. Finding the expectation is quite straight forward.

$$\mathrm{E}[d_{k^*} - d_{k_g}]$$

$$= \mathrm{E}\left[\frac{1}{2N\sigma^2}\sum_{i=1}^{N}\left((f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))^2 + 2(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))n_i\right)\right]$$

$$= \frac{1}{2N\sigma^2}\sum_{i=1}^{N}\left(\mathrm{E}\left[(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))^2\right] + \mathrm{E}\left[2(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))n_i\right]\right)$$

As $\mathrm{E}[n_i] = 0$ we get

$$= \frac{1}{2N\sigma^2}\sum_{i=1}^{N}\left(\mathrm{E}\left[(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))^2\right]\right)$$

The expected value of the difference squared of the model leakage under different keys is exactly the confusion coefficient of those keys, hence we end up with

$$\mathrm{E}[d_{k^*} - d_{k_g}] = \frac{1}{2N\sigma^2}\sum_{i=1}^{N}\kappa(k^*, k_g) = \frac{\kappa(k^*, k_g)}{2\sigma^2}.$$

For the covariance calculation we will begin will use the much used fact from 2.5. Hence we have that that for correct key k^*, and key-guesses k_g, k'_g

$$\mathrm{Cov}[d_{k^*} - d_{k_g}, d_{k^*} - d_{k'_g}] = \mathrm{E}[(d_{k^*} - d_{k_g})(d_{k^*} - d_{k'_g})] - \mathrm{E}[d_{k^*} - d_{k_g}]\mathrm{E}[d_{k^*} - d_{k'_g}].$$

We already have the results of the last two expectations from before so we will focus on the first one.

$$\begin{aligned}
&\mathrm{E}[(d_{k^*} - d_{k_g})(d_{k^*} - d_{k'_g})] \\
&= (\frac{1}{2N\sigma^2})^2 \sum_{i=1}^{N}\sum_{j=1}^{N} \\
&\mathrm{E}[\big((f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g)))^2 + 2(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g)))n_i\big) \\
&\big((f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g)))^2 + 2(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g)))n_j\big)] \\
&= (\frac{1}{2N\sigma^2})^2 \sum_{i=1}^{N}\sum_{j=1}^{N} \\
&\mathrm{E}[(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g)))^2(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g)))^2 \\
&+ 2(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g)))^2(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g)))n_j \\
&+ 2(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g)))n_i(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g)))^2 \\
&+ 4(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g)))n_i(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g)))n_j]
\end{aligned}$$

From the model assumptions we have that $\mathrm{E}[n_i] = 0$ and $\mathrm{E}[n_i n_j] = 0$ when $i \neq j$, hence we get that all the terms that have one n_i factor are 0, and that $n_i n_j$ is only nonzero for $i = j$. Hence we can rewrite the above expression as:

$$\begin{aligned}
&\mathrm{E}[(d_{k^*} - d_{k_g})(d_{k^*} - d_{k'_g})] = (\frac{1}{2N\sigma^2})^2 \\
&\Big(\sum_{i=1}^{N}\sum_{j=1}^{N}\mathrm{E}[(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g)))^2(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g)))^2] \\
&+ 4\sum_{i=1}^{N}\mathrm{E}[(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g)))(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g)))]\mathrm{E}[n_j^2]\Big)
\end{aligned}$$

Again we use an independence assumption of the model, namely that the different traces are independent from each other. Hence we have than when we go over different plaintexts in the double sum, i.e when $i \neq j$, then $\mathrm{E}[XY] = \mathrm{E}[X]\mathrm{E}[Y]$. Therefore we get two sums. We also have that $\mathrm{E}[n_j^2] = \sigma^2$ from equation 2.5 and

the model. We can now write the above expression as:

$$\begin{aligned}
&\mathrm{E}[(d_{k^*} - d_{k_g})(d_{k^*} - d_{k'_g})] = (\frac{1}{2N\sigma^2})^2 \\
&\Big(\sum_{i=1}^{N} \big(\mathrm{E}[(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))^2 (f(\varphi(x_i, k^*)) - f(\varphi(x_i, k'_g)))^2] \\
&+ \sum_{j \neq i}^{N} \mathrm{E}[(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))^2] \mathrm{E}[(f(\varphi(x_j, k^*)) - f(\varphi(x_j, k'_g)))^2] \big) \\
&+ (2\sigma)^2 \sum_{i=1}^{N} \mathrm{E}[(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))(f(\varphi(x_j, k^*)) - f(\varphi(x_j, k'_g)))] \Big)
\end{aligned}$$

Now we have all the expectations on a form where we can express them as confusion coefficients. Hence we use that $\mathrm{E}[(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))^2 (f(\varphi(x_i, k^*)) - f(\varphi(x_i, k'_g)))^2] = \tilde{\kappa}^*(k^*, k_g, k'_g)$, $E[(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))(f(\varphi(x_j, k^*)) - f(\varphi(x_j, k'_g)))] = \tilde{\kappa}(k^*, k_g, k'_g)$, and $\mathrm{E}[(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)))^2] = \kappa(k^*, k_g)$. Then the equation is on the form

$$\begin{aligned}
&\mathrm{E}[(d_{k^*} - d_{k_g})(d_{k^*} - d_{k'_g})] = (\frac{1}{2N\sigma^2})^2 \\
&\Big(\sum_{i=1}^{N} \big(\tilde{\kappa}^*(k^*, k_g, k'_g) + \sum_{j \neq i}^{N} \kappa(k^*, k_g) \kappa(k^*, k'_g) \big) + (2\sigma)^2 \sum_{i=1}^{N} \tilde{\kappa}(k^*, k_g, k'_g) \Big) \\
&= (\frac{1}{2N\sigma^2})^2 \left(N\tilde{\kappa}^*(k^*, k_g, k'_g) + N(N-1)\kappa(k^*, k_g)\kappa(k^*, k'_g) + (2\sigma)^2 N\tilde{\kappa}(k^*, k_g, k'_g) \right).
\end{aligned}$$

If we now look at the full covariance, it is possible to simplify the expression further,

$$\begin{aligned}
&\mathrm{Cov}[d_{k^*} - d_{k_g}, d_{k^*} - d_{k'_g}] = \mathrm{E}[(d_{k^*} - d_{k_g})(d_{k^*} - d_{k'_g})] - \mathrm{E}[d_{k^*} - d_{k_g}] \mathrm{E}[d_{k^*} - d_{k'_g}] \\
&= (\frac{1}{2N\sigma^2})^2 \left(N\tilde{\kappa}^*(k^*, k_g, k'_g) + N(N-1)\kappa(k^*, k_g)\kappa(k^*, k'_g) + (2\sigma)^2 N\tilde{\kappa}(k^*, k_g, k'_g) \right) \\
&- \frac{\kappa(k^*, k_g)}{2\sigma^2} \frac{\kappa(k^*, k'_g)}{2\sigma^2} \\
&= \frac{1}{4N\sigma^4} \tilde{\kappa}^*(k^*, k_g, k'_g) + \frac{N-1-N}{4N\sigma^4} \kappa(k^*, k_g)\kappa(k^*, k'_g) + \frac{1}{N\sigma^2} \tilde{\kappa}(k^*, k_g, k'_g) \\
&= \frac{1}{N} \left(\frac{1}{\sigma^2} \tilde{\kappa}(k^*, k_g, k'_g) + \frac{1}{4\sigma^4} \big(\tilde{\kappa}^*(k^*, k_g, k'_g) + \kappa(k^*, k_g)\kappa(k^*, k'_g) \big) \right)
\end{aligned}$$

This is for each key-guess pair k_g, k'_g, hence the covariance matrix over all $(N-1)$ wrong key-guesses becomes the expression stated. □

4.2. A correlation distinguisher with confusion analysis. In order to get the distributions of a correlation attack we will calculate the expectations and covariance of all the key-guesses individually, this becomes:

THEOREM 6.8. *For leakage generated by key k^*, the expectation $\mathrm{E}[\dot{\rho}_{k^*} - \dot{\rho}_g]$ and covariance $\mathrm{Cov}[\dot{\rho}_{k^*} - \dot{\rho}_{k_g}, \dot{\rho}_{k^*} - \dot{\rho}_{k'_g}]$, where k_g and k'_g are key-guesses, satisfies:*

$$\mathrm{E}[\dot{\rho}_{k^*} - \dot{\rho}_{k_g}] = \frac{\kappa(k^*, k_g)}{2\sigma_{f(\varphi(-,k^*))}} \tag{21}$$

$$\mathrm{Cov}[\dot{\rho}_{k^*} - \dot{\rho}_{k_g}, \dot{\rho}_{k^*} - \dot{\rho}_{k'_g}] = \tag{22}$$

$$\frac{\sigma^2}{N\sigma^2_{f(\varphi(-,k^*))}} \left(\kappa(k^*, k_g, k'_g) + (\frac{1}{\sigma^2}) \left(\kappa(k^*, k_g, k'_g) - \frac{1}{4}\kappa(k^*, k_g)\kappa(k^*, k'_g) \right) \right)$$

In order to prove the distribution of the differences we will need a small results relating to the confusion coefficient using the assumptions in this model.

REMARK 6.9 (Constant expectation assumption). According to Fei et al. [5] we have that for cryptosystems with identical model distribution for all keys, we also have identical expectation for all keys. We can write this as:

$$\mathrm{E}[f(\varphi(x, k_1)] = \mathrm{E}[f(\varphi(x, k_2)] \text{ for all keys } k_1, k_2 \in \mathcal{K}$$

LEMMA 6.10. *For keys k^* and k_g we have that:*

$$\mathrm{E}[f(\varphi(x, k^*))\,(f(\varphi(x, k^*) - f(\varphi(x, k_g))] = \frac{\kappa(k^*, k_g)}{2}$$

PROOF. By definition we have that $\kappa(k^*, k_g) = \mathrm{E}[(f(\varphi(x, k^*) - f(\varphi(x, k_g))^2]$, we can write this out further to get.

$$\kappa(k^*, k_g) = \mathrm{E}[f(\varphi(x, k^*)^2] - 2\mathrm{E}[f(\varphi(x, k^*) f(\varphi(x, k_g)] + \mathrm{E}[f(\varphi(x, k_g)^2].$$

From assumption 6.9, we have that $\mathrm{E}[f(\varphi(x, k^*)^2] = E[f(\varphi(x, k_g)^2]$, hence we can write the expression on the from:

$$\begin{aligned} \kappa(k_c, k_g) &= 2\mathrm{E}[f(\varphi(x, k^*)^2] - 2\mathrm{E}[f(\varphi(x, k^*) f(\varphi(x, k_g)] \\ &= 2\mathrm{E}[f(\varphi(x, k^*)^2 - f(\varphi(x, k^*) f(\varphi(x, k_g)] \\ &= 2\mathrm{E}[(f(\varphi(x, k^*))(f(\varphi(x, k^*) - f(\varphi(x, k_g)] \Rightarrow \\ \frac{\kappa(k^*, k_g)}{2} &= \mathrm{E}[(f(\varphi(x, k^*))(f(\varphi(x, k^*) - f(\varphi(x, k_g)] \end{aligned}$$

□

With this result we can find the success-rate distribution for the CPA distinguisher in terms of the confusion coefficient.

Proof theorem 6.8.

$$b_g = \frac{1}{N\sigma_{f(\varphi(-,k^*))}} \sum_{i=1}^{N} (f(\varphi(x_i, k^*) + n_i) f(\varphi(x_i, k_g)))$$

From Fei et al. [5] we have that finding the probability that $\dot{\rho}_c > \dot{\rho}_g$, is equivalent to showing $b_{k^*} > b_g$. Hence we will calculate the distribution similarly as in theorem 6.7, therefore when the steps are similar we will refer to that proof.

We can write $b_{k^*} - b_g$ in a different for to simplify the calculations.

$$\begin{aligned} b_{k^*} - b_g = \\ \frac{1}{N\sigma_{f(\varphi(-,k^*))}} \sum_{i=1}^{N} (f(\varphi(x_i, k^*) + n_i) f(\varphi(x_i, k^*)) - (f(\varphi(x_i, k^*) + n_i) f(\varphi(x_i, k_g))) \\ \frac{1}{N\sigma_{f(\varphi(-,k^*))}} \sum_{i=1}^{N} \big(f(\varphi(x_i, k^*)(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g))) \\ + n_i \big(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)) \big) \big) \end{aligned}$$

We can now quite easily calculate the expectation as $\mathrm{E}[n_i] = 0$, so we can disregard the second term. From lemma 6.10, we have that $\mathrm{E}[f(\varphi(x_i, k^*)(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)) = \frac{\kappa(k^*, k_g)}{2}$, so we can write the expectation as

$$\mathrm{E}[b_{k^*} - b_g] = \frac{1}{N\sigma_{f(\varphi(-,k^*))}} \sum_{i=1}^{N} \frac{\kappa(k^*, k_g)}{2} = \frac{\kappa(k^*, k_g)}{2\sigma_{f(\varphi(-,k^*))}},$$

which is what we wanted. As mentioned the covariance calculation is similar to the covariance calculation in theorem 6.7, except with a slightly different expression, hence the independence and 0 expectation explanations will not be as detailed. We know that

$$\mathrm{Cov}[b_{k^*} - b_{k_g}, b_{k^*} - b_{k'_g}] = \mathrm{E}[(b_{k^*} - b_{k_g})(b_{k^*} - b_{k'_g})] - \mathrm{E}[b_{k^*} - b_{k_g}]\mathrm{E}[b_{k^*} - b_{k'_g}],$$

and as before we will focus on the first term as we have the formula for the second one.

$$\begin{aligned} \mathrm{E}[(b_{k^*} - b_{k_g})(b_{k^*} - b_{k'_g}) = \mathrm{E}\Big[\frac{1}{N^2 \sigma^2_{f(\varphi(-,k^*))}} \sum_{i=1}^{N} \sum_{j=1}^{N} \\ \big((f(\varphi(x_i, k^*)) + n_i \big) \big(f(\varphi(x_i, k^*)) - f(\varphi(x_i, k_g)) \big) \big) \\ \big((f(\varphi(x_j, k^*)) + n_j \big) \big(f(\varphi(x_j, k^*)) - f(\varphi(x_j, k_{k'_g})) \big) \big) \Big] \end{aligned}$$

This big sum is quite similar to the one in the previous proof. Hence we can use that the expectation of the noise is 0, mutually independent and independent from the intermediate calculation. We then get the expectation on the form

$$\begin{aligned}
&\mathrm{E}[(b_{k^*} - b_{k_g})(b_{k^*} - b_{k'_g})] = \frac{1}{N^2\sigma^2_{f(\varphi(-,k^*))}}\Big(\sum_{i=1}^{N} \\
&\mathrm{E}\big[\big((f(\varphi(x_i,k^*)) + n_i\big)^2\big(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g))\big)\big(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k'_g))\big)\big] \\
&+ \sum_{j\neq i}\mathrm{E}\big[\big((f(\varphi(x_i,k^*)) + n_i\big)\big((f(\varphi(x_j,k^*)) + n_j\big) \\
&\qquad \big(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g))\big)\big(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g))\big)\big]\Big)
\end{aligned}$$

Now we use the independence and 0 expectation results as in 6.7

$$\begin{aligned}
&\frac{1}{N^2\sigma^2_{f(\varphi(-,k^*))}}\Big(\sum_{i=1}^{N} \\
&\mathrm{E}[(f(\varphi(x_i,k^*))^2\big(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g))\big)\big(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k'_g))\big)] \\
&+ \mathrm{E}[n_i^2]\mathrm{E}[\big(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g))\big)\big(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k'_g))\big)] \\
&+ \sum_{j\neq i}\mathrm{E}[\big((f(\varphi(x_i,k^*))(f(\varphi(x_j,k^*)) \\
&\qquad \big(f(\varphi(x_i,k^*)) - f(\varphi(x_i,k_g))\big)\big(f(\varphi(x_j,k^*)) - f(\varphi(x_j,k'_g))\big)]
\end{aligned}$$

By lemma 6.10,the independence of $f(\varphi(x_i,k)$ and $f(\varphi(x_j,k)$, and assumption 6.9 that $\mathrm{E}[f(\varphi(x,k)) = 0$ we have all the different expressions on a form that we can define with confusion coefficients, namely

$$\begin{aligned}
\mathrm{E}[(b_{k^*} - b_{k_g})(b_{k^*} - b_{k'_g})] = \frac{1}{N^2\sigma^2_{f(\varphi(-,k^*))}}\Big(&N\big(\hat{\kappa}(k^*,k_g,k'_g) + \sigma^2\tilde{\kappa}(k^*,k_g,k'_g)\big) \\
&+ N(N-1)\frac{\kappa(k^*,k_g)\kappa(k^*,k'_g)}{4}\Big).
\end{aligned}$$

All together we have that the covariance becomes:

$$\begin{aligned}
&\mathrm{Cov}[(b_{k^*} - b_{k_g}),(b_{k^*} - b_{k'_g})] \\
&= \frac{1}{N^2\sigma^2_{f(\varphi(-,k^*))}}\Big(N\big(\hat{\kappa}(k^*,k_g,k'_g) + \sigma^2\tilde{\kappa}(k^*,k_g,k'_g)\big) \\
&+ N(N-1)\frac{\kappa(k^*,k_g)\kappa(k^*,k'_g)}{4}\Big) - \frac{\kappa(k^*,k_g)\kappa(k^*,k'_g)}{4\sigma^2_{f(\varphi(-,k^*))}} \\
&= \frac{1}{N\sigma^2_{f(\varphi(-,k^*))}}\Big(\hat{\kappa}(k^*,k_g,k'_g) + \sigma^2\tilde{\kappa}(k^*,k_g,k'_g) - \frac{\kappa(k^*,k_g)\kappa(k^*,k'_g)}{4}\Big)
\end{aligned}$$

□

Now that we have the expression for all the coordinates of the covariance matrix and expectation vector we have all we need to estimate the success rate, now expressed with confusion coefficients.

CHAPTER 7

Estimating the success-rate of an attack

One of the main purposes of calculating the probability distributions of different side channel attacks is to estimate the success rate given different noise parameters. The reason for wanting to estimate it rather than actually doing the attacks and seeing the success, is that this is in some cases not possible or prohibitively costly to do. Especially since we can be in the scenario where a success rate estimation is too costly to estimate, but the actual attack is possible for a motivated attacker. Hence in this chapter we will look at how we can estimate the success rate of a side channel attack. One of the methods will be using the distinguisher distributions we calculated in the last chapter. A way to simulate attacks to estimate the success rate will also be introduced.

1. Estimating success-rates from cumulative distribution

With the probability distribution of the different distinguishers we can say what probability the distinguishing vector has of being a certain value. This enables us to calculate a theoretical success-rate by relating the distinguisher vector values that is a successful attack to the distribution. Namely the case that the biggest distinguishing score is the correct key-guess d_{k^*}. We will relate this case to the distributions we have calculated by defining a new distribution where a successful attack is the case where all the coordinates in the vector are positive.

In order to describe the distribution of success we need a way to compare the correct key-score to the other ones. A simple way to this is to take the difference. For correct key k^* with distinguishing score d_{k^*} is scored higher than a different key-guess score d_k if $d_{k^*} - d_k > 0$.

Definition 7.1. A *comparison vector* $\mathbf{c}$ is a $(N_k - 1)$-sized vector defined from (N_k)-sized distinguishing vector $\mathbf{d}$, where $c_i = d_{k^*} - d_i$, for all i except for k^*. k^* is the correct key-guess.

We can describe the comparison vector in terms of a linear transformation $\mathbf{c} = P\mathbf{d}$ with $(N_k) \times (N_k - 1)$ matrix

$$P = \begin{bmatrix} -1 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & -1 \end{bmatrix}.$$

In P the 1-column is at the k^* coordinate, and the -1 is at the same index as the row, except for the k^* row where the index skips one index . The essential attribute of $\mathbf{c}$ is the fact that $\mathbf{d}$ successfully recovers correct key k^* if and only if all entries of $\mathbf{c}$ are positive. Since $\mathbf{d}$ has an associated distribution $\mathbf{d} \sim \mathcal{N}(m_{\mathbf{d}}, \sum_{\mathbf{d}})$, we get that the comparison vector has distribution $\mathbf{c} \sim \mathcal{N}(Pm_{\mathbf{d}}, P\sum_{\mathbf{d}} P^T)$.

A convenient fact of the confusion analysis in 6, is that for the distribution calculation the expression is already on the form above. Namely where the key-guesses are subtracted to the correct key. Hence for the distribution success rate of the confusion coefficient distributions the transformation is not necessary and one can go straight to the cumulative distribution calculation.

Due to the fact that the comparison vector $\mathbf{c}$ is positive for a successful key-guess, and that is has distribution $\mathbf{c} \sim \mathcal{N}(Pm_{\mathbf{d}}, P\sum_{\mathbf{d}} P^T)$ it is now possible to describe a successful key recovery attack:

$$\text{Succ-o}^D_{\boldsymbol{x},k^*} = P[\mathbf{c} > 0] = \Phi_{(Pm_{\mathbf{d}}, P\sum_{\mathbf{d}} P^T)}(\mathbf{0}, \boldsymbol{\infty})$$

By $\mathbf{c} > 0$ we mean that each entry in the comparison vector is bigger than 0. By $\mathbf{0}$ and $\boldsymbol{\infty}$ we mean the $N_k - 1$-sized vector containing that value. With this description of success-rate it is now possible to calculate the distribution of success for specific distinguishers. One point however is that this is a $N_k - 1$ dimensional cdf. As the dimensions get bigger this gets more and more demanding to calculate. For AES-128 the cdf to calculate the success rate is 255 dimensional. Therefore it is useful to also have a different way to estimate the success rate.

2. Estimating success-rates from Monte Carlo simulation

In the Monte Carlo simulation method we perform side channel attacks on traces generated in the model with specific noise parameters. Then one can calculate if the distinguisher recovered the correct key and count it as an success if it does. This is done many times, and the ratio of successes vs non-success becomes the success-rate.

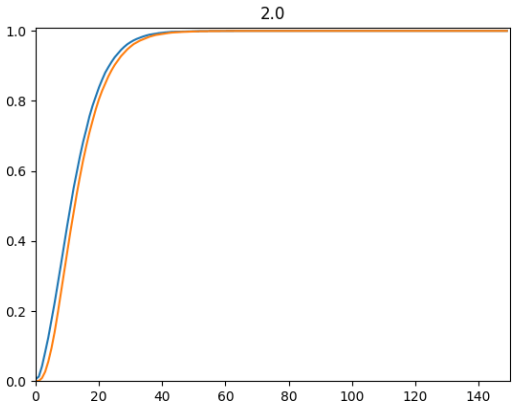


FIGURE 1. The graph shows the success rates on the y-axis and the number of traces on the x-axis. The orange line indicates a simulated correlation attack success rate and the blue line indicates a simulated profiling attack. The standard deviation σ is 2 for the generated traces.

A downside of this method is that it is both computationally heavy as one has to perform the thousands of attacks in order to get a good estimate of the success-rate. One also has to specify the leakage parameters before estimating the success rate. This results in that the variance of the leakage has to be specified for every estimation, as opposed to having a general distribution with a variance parameter, which is then easier to change later.

In the side channel distinguisher case we can do Monte Carlo simulation by generating leakage according to the model with specified noise. Hence one generates random plaintext inputs x_i and create a leakage point with it by sampling from $f(\mathrm{m}(x_i)) + n$ where $n \sim \mathcal{N}(0, \sigma)$, where we have chosen sigma. One can do this repeatedly with randomly chosen x_i each time. Then one simply used the distinguisher and perform the side channel attack and look at the resulting distinguishing vector. If the correct key is the highest scoring one then the experiment is counted as a success. One can do this iterative where traces are added and the score recalculated. Then one can perform this experiment for an appropriate number of times, which depends on the plaintext and key space. In this experiment keys we have $(2^8)^2 = 65536$ in other words the plaintext- and key-space squared.

In Figure 1, we can see how the success-rate changes for the two different first order distinguishers described in Chapter 4 and 5.

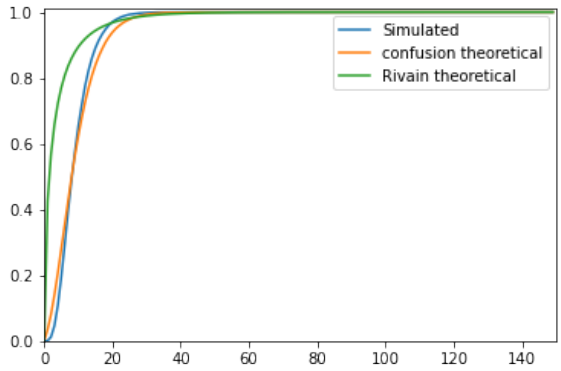


FIGURE 2. The figure shows the comparison of the success-rate vs number of traces with different estimation methods. Namely the Rivain [14] in green, Fei et al. [5] estimation in orange, and the simulated success-rate from Section 2 in blue. This is for a standard deviation of the noise of 1, and for the correlation distinguisher described in 1.

3. Comparing success-rate estimates

In this thesis three techniques of success rates estimation have been introduced. Namely the Rivain [14] distribution estimation described in Chapter 4 and 5. The confusion coefficient distribution estimation based on Fei et al. [5], and described in Chapter 6. Lastly we have the simulated success-rates from the Monte Carlo technique. We can compare these techniques by comparing the success-rates for a specific standard deviation.

In Figure 2, we can see the three estimations compared. The Rivain success-rate estimates a higher probability of success-than the simulated method, this follows the results in [14] as one can see for the profile distinguisher success-rate in Figure 3.

The reasons for the difference in the estimates of the success-rates is an area that is interesting to explore, but is out of scope of this thesis. An important factor for the differences could be the simplification assumptions that are made in the leakage model, and for the distinguishers. This thesis have also focused on the theoretical success-rates which is based on the model of leakage. How these estimations relate to real life attacks is also an interesting area to explore, however here one need to do measurements on a computer chip running the cryptographic algorithm which is also out of scope of this thesis.

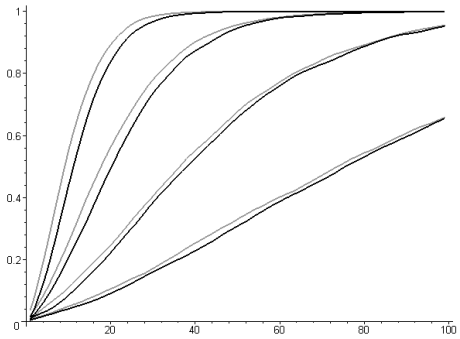


FIGURE 3. The figure shows the comparison of the success-rate vs number of traces with the Rivain estimation [14] in grey, and the simulated success-rate in black. The figure is taken from the same paper.

Hence we have established two techniques for estimating the success-rate of a side channel attack, given the noise parameters of the leakage model. This concludes the exploration into side channel analysis as we now have explored the main fundamental aspects of what goes into estimating the efficiency of a side channel attack in a theoretical setting. Namely establishing a model for how side channels behave and what leakage looks like. Defining the goals for side channel attacks, and defining distinguishers that exploit the leakage model in order to reach this goal. Then the distribution of the leakage model is associated with the distinguisher scores, which is then again associated to the probability of success. Hopefully the reader now has a sense of what attributes of a cryptographic implementation side channel attacks exploits in order to recover the secret key. As well as a sense of how we describe these attributes in a theoretical setting in order to estimate how efficient these types of attacks are.

Bibliography

[1] E. Brier, C. Clavier, and F. Olivier. "Correlation Power Analysis with a Leakage Model". In: *Cryptographic Hardware and Embedded Systems - CHES 2004*. Ed. by M. Joye and J.-J. Quisquater. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 16–29.

[2] D. Brumley and D. Boneh. "Remote Timing Attacks Are Practical". In: *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12*. SSYM'03. Washington, DC: USENIX Association, 2003, p. 1.

[3] S. Chari et al. "Towards Sound Approaches to Counteract Power-Analysis Attacks". In: *Advances in Cryptology — CRYPTO' 99*. Ed. by M. Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 398–412.

[4] N. Costes and M. Stam. "Redundant Code-based Masking Revisited". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.1 (2020), pp. 426–450.

[5] Y. Fei et al. *A Statistics-based Fundamental Model for Side-channel Attack Analysis*. Cryptology ePrint Archive, Report 2014/152. https://ia.cr/2014/152. 2014.

[6] L. Isserlis. "On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables". In: *Biometrika* 12.1-2 (Nov. 1918), pp. 134–139.

[7] P. Kocher et al. "Introduction to differential power analysis". In: *J. Cryptographic Engineering* 1 (Apr. 2011), pp. 5–27.

[8] R. Larsen and M. Marx. *An Introduction to Mathematical Statistics and Its Applications*. Pearson Education, 2011.

[9] V. Lomné et al. "How to Estimate the Success Rate of Higher-Order Side-Channel Attacks". In: *Cryptographic Hardware and Embedded Systems – CHES 2014*. Ed. by L. Batina and M. Robshaw. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 35–54.

[10] T. Messerges, E. Dabbish, and R. Sloan. "Investigations of Power Analysis Attacks on Smartcards". In: (Sept. 1999).

[11] S. Ors et al. “Power-analysis attack on an ASIC AES implementation”. In: *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.* Vol. 2. 2004, 546–552 Vol.2.

[12] E. Prouff and M. Rivain. “Masking against Side-Channel Attacks: A Formal Security Proof”. In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by T. Johansson and P. Q. Nguyen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 142–159.

[13] E. Prouff, M. Rivain, and R. Bevan. “Statistical Analysis of Second Order Differential Power Analysis”. In: *IEEE Transactions on Computers* 58.6 (2009), pp. 799–811.

[14] M. Rivain. “On the Exact Success Rate of Side Channel Analysis in the Gaussian Model”. In: *Selected Areas in Cryptography*. Ed. by R. M. Avanzi, L. Keliher, and F. Sica. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 165–183.

[15] A. Shamir. “How to Share a Secret”. In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613.

[16] F.-X. Standaert. “Introduction to Side-Channel Attacks”. In: Dec. 2010, pp. 27–42.