Signal Theoretical Methods in Differential Side Channel Cryptanalysis

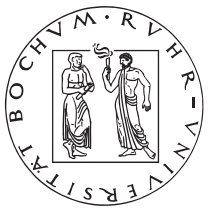
Diplomarbeit

by

Benedikt Gierlichs

gierlichs@crypto.rub.de

Advisor: Dipl. Phys. K. Lemke



Ruhr-University Bochum
Faculty of Electrical Engineering
and Information Technology
Chair for Communication Security

Advisor: Dr. H. Handschuh, Dr. P. Paillier

Gemplus SA
Research and Development
Security Technologies Department

Supervisor: Prof. Dr.-Ing. Christof Paar

Beginning: September 29, 2005
Filing: March 29, 2006

Erklärung

Hiermit versichere ich, daß ich meine Diplomarbeit selbst verfasst und keine anderen als angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

I hereby certify that the work presented in this thesis is my own work and that to the best of my knowledge it is original except where indicated by reference to other authors.

Bochum, den 29. März 2006

Benedikt Gierlichs

Abstract

This work presents the application of Signal Theoretical Methods to Side Channel leakage of Cryptographic Devices, particularly with regard to power consumption and electromagnetic radiation. We profoundly analyse the Template Attack and the Stochastic Model and compare their efficiencies in various parameter settings. Finally, we suggest and verify improvements of both attacks which yield success probabilities increased by a factor of up to 5.

Acknowledgments

This work has been accomplished at the Security Technologies Department of Gemplus SA in Paris and the Side Channel Lab of the Communication Security Group at Ruhr-University Bochum. Thankful acknowledgment is made to many people for their support, feedback, and suggestions. In particular I would like to say:
Danke to Christof Paar and Merci to David Naccache for paving the way of this project. Danke an Kerstin für die Betreuung und Unterstützung. Merci à Helena pour nos discussions sur le bruit. Merci à Pascal de m'avoir présenté à Mathematica ...

Un vachement gros merci à la familie Brière, qui m'a reçu cordialement et particulièrement à Mélanie, qui m'a soutenue toujours.
Nicht zuletzt ein großer Dank an meine Freunde in Bochum, die mich durch dieses Studium begleitet haben und an alle anderen (et à tout les autres), deren Namen ich hier vergessen habe (que j'ai oublié dans cette liste)!

Contents

List of Figures

List of Tables

1 Introduction, Overview

"Cryptanalysis is the study of mathematical techniques for attempting to defeat cryptographic techniques, and, more generally, information security services" [8, p.15].
"Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, [...]" [8, p.4].

Together, Cryptography and Cryptanalysis comprise the science Cryptology wherein they can be illustrated as oppositions. Cryptanalysis has a strong impact on Cryptography as it poses as evaluation. Continuous attempts to thwart protection of information provided by cryptographic means expose their weaknesses and enhance knowledge about them. This leads to the development of improved cryptographic tools whose security, in turn, will be questioned. Simultaneously, knowledge gained from cryptanalytic approaches leads to enhanced analytical methods. This phenomenon is known as the continuous competition between designers and analysts, or more familiar, the cat-and-mouse-game. It motivates progress on both sides and so far, none of the opponents has been able to do the ultimate move that ends "the game".
In (classical) Cryptanalysis, the security level of a cryptographic technique is determined purely theoretically. Therefore, an algorithm that describes the operation of the technique is considered. Basically, the complexity of an attack against an algorithm is determined from only looking at the underlying logical structures and is given by a workload estimation. The complexity of the most efficient attack against an algorithm defines its security level.
But in the last decade, the field of Cryptanalysis has experienced major changes. Cryptanalysts do not only look at abstract algorithms anymore but consider their concrete implementations in electronic devices, too. Since the discovery of implementation attacks, cryptographers have to watch cryptanalysts coming up with attacks which easily defeat cryptographic protections of implemented algorithms which are regarded as secure from a classic point of view. The difference is crucial: an effective implementation attack does

not affect the security level of the cryptographic algorithm but the security provided by its implementation. Hence, there are many algorithms that are still considered as secure, although there exist effective attacks against non-protected implementations.

Implementation attacks expand cryptanalysis into the world of physics. Real devices that execute cryptographic operations deliver much more information than only the intended output of the algorithm. The term *Side Channel* abstracts all unintended information leakage, e.g. power consumption of the device. Attacks based on this information are *Side Channel Attacks*. Side Channel Attacks have also raised new problems within Cryptanalysis, inaccuracy has entered the field. While the efficiency of a classic attack is mostly expressed by computational complexity and therefore comparable to that of other attacks against the same algorithm, the situation is somewhat different for Side Channel Attacks. They process measured data of physical observables to achieve their goal and physics does not only know `0` and `1` but often prefers numbers like `68,17469`. Amongst other factors, the efficiency (or complexity) of a Side Channel Attack significantly depends on the *quality* of the side channel information which in turn is influenced by numerous sources. These coherences let strong statements on efficiency appear a quite daunting task and in fact, many publications in this area evade precision when an attack's efficiency is — estimated (cp. [16, 14]). So far there exists no measure for side channel *quality* (or the resulting complexity) which for the moment abstracts environmental, device specific, implementation specific, and measurement specific influences on an attack's efficiency. A comparison of two Side Channel Attacks which obviously requires not only that the same cryptographic algorithm is attacked, but as well that the underlying side channel complexity is considered, practically means a comparison under identical physical conditions.

In this context, we[1] regard further investigation of known side channel attacks as valuable. We think that understanding more detailed *how*, *why*, and under *which* circumstances a certain attack works (better than another one) will lead to more precise conclusions and to progress in Side Channel

[1] Although I prefer to write "we" than "I", this thesis presents my personal work.

Cryptanalysis.

Therefore, the primary goal of this diploma thesis is a profound analysis and comparison of Template Attacks and the Stochastic Model. In addition to a comparison of key disclosure success rates, we aim at understanding the results in order to learn more about each attack's nature.
Finally, it turned out that we learned enough about the attacks to suggest and verify improvements for both of them.

This document is structured as follows. Section 2 briefly introduces Cryptanalysis and Side Channel Cryptanalysis. Section 3 provides the theoretical fundamentals used in this thesis, while Sections 4 and 5 give insights into the practical work which was performed and present the obtained results. Section 6 comprises the analysis of the results and our suggestion of improvements, whose revised results are given in Section 7. Section 8 covers our work on EM and Multichannel attacks. Our conclusion and further research topics are given in Section 9.

2 Classical and Side Channel Cryptanalysis

In this section, a transition from classical cryptanalytic to recent side channel techniques is given. After an introduction to classical Cryptanalysis and the general idea of side channel attacks, the latter ones will be explored more detailed. The state of the art is presented and recent problems are outlined.

2.1 Classical Cryptanalysis

From a historical point of view, Cryptanalysis is related to the analysis of employed cryptographic algorithms in order to find and exploit weaknesses within their logical structure. Concrete attacks which apply such knowledge are referred to as *Logical Attacks* or *Cryptanalytic Attacks* in literature.
Cryptanalytic approaches are always embedded into a model, often referred to as *attack scenario*, that defines a framework, in particular the attackers abilities and goals. The general approach of classical Cryptanalysis is depicted in Figure 1. In this setting, the attacker of a cryptographic primitive,

Figure 1: Model for classical Cryptanalysis

e.g. encryption, has the following abilities: he knows the cryptographic algorithm, he can choose inputs to the algorithm at his will, and he can observe its output. For example, in the case of encryption (resp. decryption) an adversary can observe the output that was computed from chosen input by a known algorithm using unknown key data. His task is to deduce the unknown key with the help of all available information. There are many variations of this attack scenario, which constrict the amount or the nature of usable data. With the attack framework being defined, the security provided by the cryptographic primitive can be evaluated under several security models. In the style of varying attack scenarios, security models (un-)limit the attackers computational resources. A cryptographic primitive is said to be secure un-

der a certain model, if it resists an adversary with the appropriate powers. We refer the interested reader to [8, pp.41], which is a rich source for further details.

For the reason of this introduction, the attention is restricted to the main characteristic of classical Cryptanalysis: The actual device performing the cryptographic operation is regarded as a black box. It generates output (e.g. ciphertext) from the corresponding given input (e.g. plaintext) using the known employed algorithm and (secret) key data. No further properties of the black box, in particular its internal operating mode, are known or considered[2].

A classical example for Cryptanalysis of a message that has been encrypted with a monoalphabetical substitution cipher (e.g. the Caesar Cipher [8, p.239]) is the observation of the frequency of letters' occurrence. Since the substitution is monoalphabetic, the plaintext's characteristic in terms of frequency distribution of the alphabet, which is characterisitic for many languages, remains intact. Hence it is a reasonable approach to assume that, for a text of reasonable size, the most common letter in the ciphertext corresponds to the most common letter in the plaintext. Then, the key can be concluded by determining the offset by which a letter gets displaced in the alphabet.

Two other well-known and more recent examples for (classical) Cryptanalysis of an employed algorithm are Linear Cryptanalysis and Differential Cryptanalysis. Both of them arose in the context of the Data Encryption Standard (DES) [3].
Linear Cryptanalysis was discovered by Mitsuru Matsui in 1992, although the premisies of its principle were initiated by Henri Gilbert. One year later he published "Linear cryptanalysis method for the DES cipher" [1], which was

[2]Note that for reduced versions of algorithms, which in a way make use of intermediate states, we might create a "smaller" black-box performing only the reduced algorithm such that the image holds.

the first successful Cryptanalysis of the cipher reported in the open community. In the broadest sense, it analyses the non-linearity of a given algorithm and comes up with a linear approximation. Matsui discovered that one of the eight Sboxes used in DES was less balanced than the others which made it possible for him to mount his attack.
The discovery of Differential Cryptanalysis in the late 1980s is attributed to Eli Biham and Adi Shamir. In 1991 they published the results of their analysis of DES in "Differential Cryptanalysis of the full 16-Round DES" [2]. Simplified, an adversary creates pairs of plaintexts comprising a certain difference and observes the difference in the corresponding ciphertexts after encryption. Statistical means are then used to detect patterns in the distribution of the differences.
Since their discovery, both attacks are a basic concern for cryptographers and newly designed ciphers are practically required to be provably resistant to them, as is for example DES' successor, the Advanced Encryption Standard (see Section 3.1).

2.2 Side Channel Cryptanalysis

Side Channel Cryptanalysis is another step in the continuous competition between designers and analysts. But it is not only an attack that successfully operates where prior attacks are ineffective. Side Channel Cryptanalysis (abbr.: SCC) is an entire new field within cryptanalytic research which has potential for various attacks and even attack styles.

SCC is based on *side channel information* which abstracts all information preservable from the cryptographic device that is not covered in the attack scenarios of (classical) Cryptanalysis. In other words, it is information which is observable additionally to the intended output of the cryptographic algorithm. These leakages carry valuable information about the device's internal state. Furthermore it is known that every electronic device[3] is not only influenced by an intended input but as well by other factors as for example

[3] which is not especially protected

external environmental conditions or physical phenomenons in the device. Figure 2 shows a model that gives consideration to these facts by newly introduced dimensions. In this model, the device carrying out a cryptographic

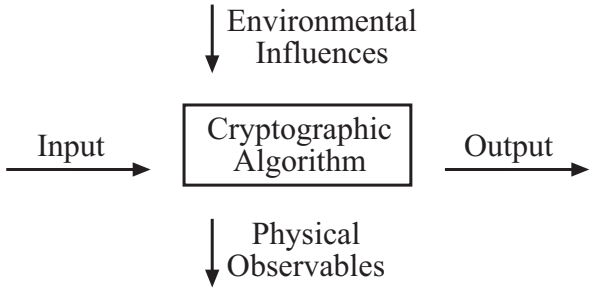


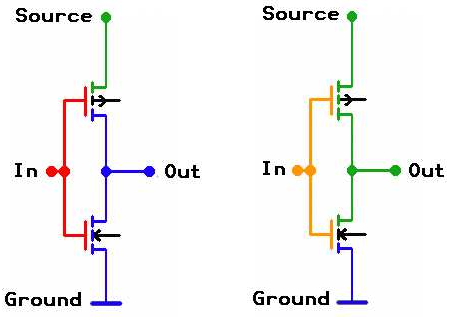
Figure 2: Model for Side Channel Cryptanalysis

operation is no longer a disregarded black-box. An algorithm's tangible implementation on the physical device is the central point. The common basis of all Side Channel Attacks (abbr.: SCAs) is to determine the device's internal state from measurements of physical observables in order to deduce the data which is processed by the device.

The first attack based on side channel information was published in 1995 by Paul Kocher. He showed, how timing information of cryptographic operations can be used to break implementations of several cryptosystems [13]. In 1998, Kocher et al. published "Simple and Differential Power Analysis" [14], two attacks that use measurements of the cryptographic device's power consumption to disclose secret key material. Electromagnetic emanation was introduced as a side channel in 2001. Quisquater and Samyde as well as Gandolfi, Mourtel and Olivier published fundamental works [15, 16].

The following paragraph exemplary shows why the power consumption of a standard digital circuit [4] carries valuable side channel information.
Almost all digital circuits are build in *Complementary Metal Oxide Semiconductor* (CMOS) technology, because it is cheap and efficient. But circuits

[4] and hence of non-protected devices



In	Out
0	Source
1	Ground

Figure 3: Logic inverter in CMOS technology and truth table

build from CMOS gates also have properties, that can be regarded as disadvantageous. The power consumption of logical gates in CMOS technology is directly correlated to their state. More precisely, the power consumption of a CMOS gate is directly correlated to its state change. Figure 3 depicts the simplest logical gate in CMOS technology: a logic inverter. For a constant input, one of the transistors is insulating and the other is conductive. In this state, the power consumption of the inverter is negligible as current cannot flow from source to ground. If the input changes, the conductivity of both transistors is inverted and there is a small time frame where both of them are conductive. During this short period of time, current can flow from source to ground which results in power consumption that is obviously correlated to the input value's alteration.

SCAs do not attack cryptographic algorithms but "only" their implementations. One must not conclude any relation between the security of an algorithm and a success probability of a SCA against one of its implementations, or vice versa. Hence, in general all implementations of cryptographic algorithms are considered to be vulnerable to SCAs, if they are not expressly protected.
For completeness it shall be mentioned that SCAs are only a subset of passive attacks against implementations of cryptographic algorithms. Note in particular that no intentional influence is exerted on the device. [12] provides detailed information on implementation attacks.

In the following it is important to distinguish two styles of SCAs because their approaches fundamentally differ and hence have different requirements. Sections 2.3 and 2.4 therefore introduce One- and Two-Step Side Channel Attacks. Section 2.5 compares representatives of both classes with respect to requirements and applicability.

2.3 One-Step Side Channel Attacks

One-Step SCAs are directly mounted against the device under attack. All side channel information or meta-information that is used by the attack is obtained from exactly the one device under attack and during this one attack.

Simple { Power ‖ ElectroMagnetic } Analysis Simple Power Analysis (SPA) and Simple ElectroMagnetic Analysis (SEMA) are known plaintext attacks. The adversary needs passive physical access to the device to obtain instantaneous measurement data. He deduces information about the processed data by e.g. the Hamming Weight leakage model which considerably reduces the brute force search space. SPA/SEMA is particularly of interest if key bits are processed sequentially by the implemented algorithm, as for example in modular exponentiation with secret exponents. However, a disadvantage of this approach is that it requires detailed knowledge about the implementation.

Differential { Power ‖ ElectroMagnetic } Analysis Differential Power Analysis (DPA) and Differential ElectroMagnetic Analysis (DEMA) require samples that represent well-spread[5] plaintexts and a fixed key k. Hence, they are known-plaintext attacks if this distribution may be assumed and chosen-plaintext attacks if not. The adversary needs passive physical access to the device under attack to obtain many[6] samples. Based on a sub-key hypothesis $k' \in \{0,1\}^n$, the adversary computes the value of a chosen key-dependent intermediate result $r \in \{0,1\}^n$ for each sample and sorts the samples to

[5] approximately equally likely distributed

[6] the exact number of required samples, usually 1000 samples should suffice, heavily depends on several factors which we summarize to side channel information quality

2^n piles with respect to r. Next, the adversary computes the average avg_r of each pile and then the sum of pairwise differences between all averages, that is $\sum_{i=0,j>i}^{r} \text{avg}_i - \text{avg}_j$. The height of the peaks in the differential trace quantifies the correlation between the key hypothesis and the correct key, hence the adversary decides for the hypothesis with maximum correlation. The higher complexity of these attacks compared to SPA/SEMA faces the advantages that superposed noise is eliminated due to the averaging process and no knowledge about the implementation is required. It is common sense that DPA/DEMA are more powerful than SPA/SEMA in the context of block ciphers, while the relation is rather inverse in the context of Public-Key techniques.

2.4 Two-Step Side Channel Attacks

Two-Step SCAs consist of two constitutive steps. The first step which will be referred to as the profiling step requires access to a training device A, which is programmable to the adversary's will and identical to the device under attack B. Note that "identical" should not be interpreted too strictly. It is common sense that a device A which fulfills the same specifications, e.g. that comes from the same production as B, suffices. At least for the case of attacks against several block-ciphers, long-term access to a non-programmable device A, this could even be device B for instance, substitutes the need of a programmable device A, see Remark 1 in Section 5.1.1.
The second step which will be referred to as the classification step involves device B in either case. Two-Step SCAs require well-spread inputs to the cryptographic algorithm in the profiling step.

Profiling Step During the profiling step, an adversary applies differential and statistical techniques to a large number of side channel samples from device A to determine the characteristics of the algorithm's implementation. In other words: he generates key-dependent profiles of the device's side channel leakage.

Classification Step During the classification step, a single or a few side channel samples[7] from device B are used to compute, for each profile, the probability that the samples resemble this profile. The profile, respectively the key hypothesis, which yields the maximum probability is the best candidate and selected.

Inferential Power Analysis Attack [9], published by P.N. Fahn and P.K. Pearson at CHES 1999, is, to the best of our knowledge, the first attack complying to our definition of Two-Step SCAs reported in the open community. Two more recent representatives of this class of SCAs are introduced in Section 3 and investigated in this thesis.

2.5 One-Step vs. Two-Step attacks

In this section, the requirements and the applicability of One- and Two-Step SCAs are confronted. Table 1 illustrates a general overview. If a training

	Training device not available	Training device available	Implementation known
many measurements from device B	DPA/DEMA	(Amplified) Two-Step Attacks	unimportant
one measurement from device B	SPA/SEMA	Two-Step Attacks	yes
		Two-Step Attacks	no

Table 1: Requirements and applicability of One- and Two-Step SCAs

device is not available, the choice of a Single-Step SCA depends only on the number of available curves. Under the reasonable assumption, that a training device is available, the range of selectable attacks is wider.

Two-Step SCAs gain relevance in consequence of successfully performing under circumstances that render most One-Step SCAs inoperative.
Consider the following reasonable assumptions for an attack scenario: an attacker might be limited in the number of samples which he can obtain from

[7] This depends on the attack's nature.

device B. Reasons for this include but are not limited to: limited access to the device or implemented techniques within the device such as non-linear key updates. In the worst case scenario, this turns into access to only a single sample. Under this assumption, DPA/DEMA style attacks obviously turn out to be unmountable.
Furthermore, the implementation of the cryptographic algorithm on the device might be unknown. One obvious reason for this circumstance is a vendor's concern in his Intellectual Property. This assumption at least considerably complicates SPA/SEMA style attacks and, in practice, takes them from the range of choice.
Two-Step SCAs perform well, even if both is assumed simultaneously.

3 Theory

This chapter provides the theoretical basis that later chapters will rely on. In Section 3.1 we introduce the Advanced Encryption Standard since knowledge of certain properties of the algorithm is required for this thesis. Section 3.2 gives a short review of all relevant statistical measures and in Section 3.3 and 3.4 the Template Attack and the Stochastic Model are introduced.

3.1 Advanced Encryption Standard

In 1997 the National Institute of Standards and Technology (NIST) invited the cryptographic community to submit proposals for a new encryption standard [4]. The new standard would be the successor of the Data Encryption Standard (DES) which was in place since 1977 and outdated in terms of the provided security level. At the end of the selection process, during that proposed ciphers were judged not only by their security and efficiency properties [4], the Rijndael algorithm [5] was chosen and standardized as the Advanced Encryption Standard (AES) [6] in November 2001. In fact, the AES only provides a subset of Rijndael's options. This is due to the fact that NIST changed the requirements for proposed ciphers during the selection process when Rijndael's basics had already been designed. [7] contains the final submission paper of Rijndael while [5] is a rich source for design strategies and detailed insights.

Because the AES resists all known forms of classical cryptanalysis and is considered secure, its implementations are widespread and a basic module in almost every application that deals with information security. This makes it an interesting object of studies for SCC but as well a good "tester" for the efficiency (or complexity) of SCAs.

The following description strictly follows [6] and provides additional information where necessary. The AES is a symmetric block cipher that processes data blocks of 128 bits. Cryptographic keys of 128, 192, and 256 bits in length can be used, where each key length leads to a specific number of rounds (10,

12, 14) and may be indicated by naming the algorithm AES-128, AES-192, or AES-256, respectively. During our experiments we only used AES in *encryption mode* and with a key length of 128 bits (10 rounds). Therefore the remainder of this document will focus on this variant.

3.1.1 Mathematical preliminaries

AES operates on bytes as its basic unit. Within a byte, single bits are identified by their index value in the following order: $b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$. Hence b_0 stands for the least significant bit, for example. Bytes are interpreted as finite field elements using a polynomial representation in GF(2^8):

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^{7} b_ix^i \qquad (1)$$

In case of the AES, operations over a GF(2^8) are defined by the *irreducible polynomial*

$$m(x) = x^8 + x^4 + x^3 + x + 1. \qquad (2)$$

Addition of two field elements can be achieved by consecutively adding coefficients of corresponding powers in the two polynomials modulo 2, since they are elements of the prime field, thus $\in \{0, 1\}$. Addition modulo 2 is equivalent to the *XOR* operation, denoted by $\oplus$. Furthermore, addition modulo 2 is equivalent to subtraction modulo 2, which implies that the same relation is true for the polynomials $\in GF(2^8)$.

Multiplication of two field elements in polynomial representation corresponds to multiplication of two polynomials modulo the *irreducible polynomial* $m(x)$. The modular reduction ensures that the result will be a polynomial of degree less than 8, hence an element of $GF(2^8)$ and representable by a byte.

The *multiplicative inverse* element $a(x)$ of any non-zero element $b(x)$ is

defined by

$$a(x) \cdot b(x) \equiv 1 \mod m(x) \Rightarrow a^{-1}(x) \equiv b(x) \mod m(x). \tag{3}$$

Further mathematical preliminaries with reference to the AES can be found in [6], for a wider overview we refer to [8].
We will mostly use hexadecimal notation to present byte values, e.g. {1A} = 26, but might change to other notations where necessary.

3.1.2 The State Array

AES' operations are performed on a two-dimensional array of 16 bytes, arranged in four rows and four columns, called *State* and denoted by s. In the beginning of the algorithm, the input data bytes are copied into the State. After all operations have been performed, the State is copied into the output, see figure 4 for details. Note that this notation is used both for encryption and decryption.

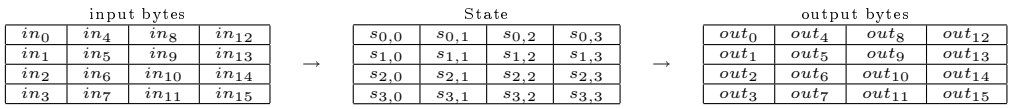


Figure 4: AES: Input, State, and Output

3.1.3 Cipher

As mentioned above, all transformations are performed on the *State*. The cipher begins with an initial Round Key addition, after which the *State* is transformed by 9 iterations of a round function. In the end, a slightly modified final round is applied once.
The round function of the AES algorithm is composed of four byte-wise transformations. In encryption mode, their order is: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The final round is identical besides the missing MixColumns transformation. Figure 5 shows the overall processing order of an AES encryption.

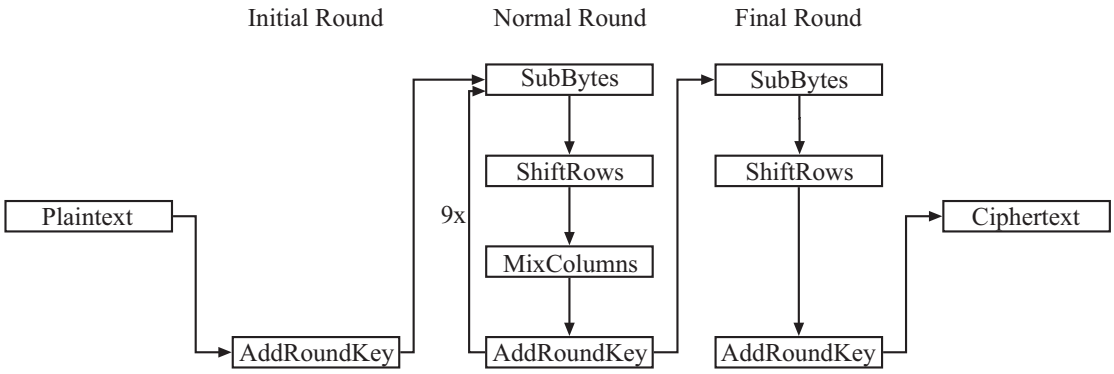


Figure 5: Overall processing order of an AES encryption

SubBytes This is a non-linear, invertible byte substitution using a substitution table (S-box). For each byte of the *State*, the following two transformations are performed:

1. The byte is substituted by its multiplicative inverse element[8] in $GF(2^8)$.

2. The affine transformation:

$$b'_i = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i \quad (4)$$

   is applied over $GF(2)$ for $0 \leq i < 8$, where b_i and c_i are the i$^{\text{th}}$ bits of the byte b and c, respectively, and c = $\{63\} = 01100011_2$.

The affine transformation can be written in matrix form as shown in Figure 6. The byte-wise effect of SubBytes is illustrated in Figure 7, see [6] for the complete S-box substitution table.

ShiftRows The ShiftRows transformation cyclically left-shifts each row of bytes within the *State* by a certain offset. The offset for each row is given by its index, e.g. row_0 is not shifted since the offset is 0. Figure 8 shows this procedure.

[8]The zero element is mapped to itself

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 6: Affine transformation in SubBytes

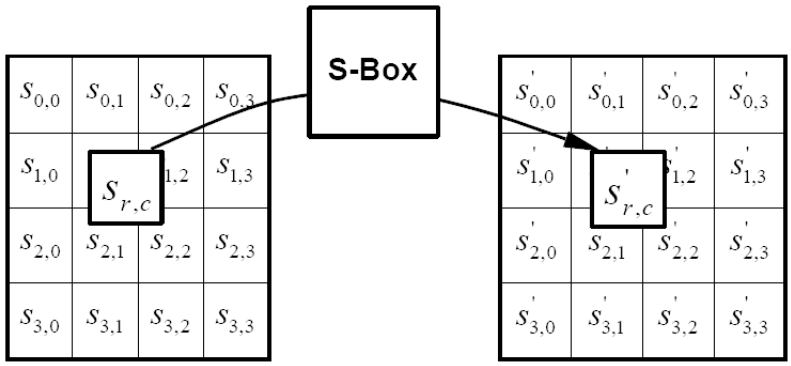


Figure 7: S-box substitution for each byte of the *State* [6]

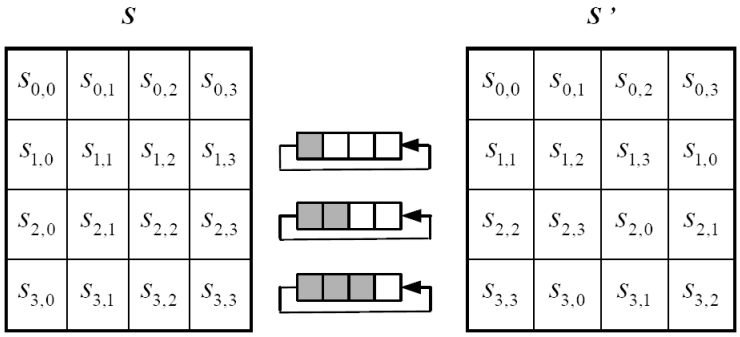


Figure 8: ShiftRows, a cyclic left-shift [6]

MixColumns MixColumns operates on the four columns of the *State*, one at a time, as can be seen in Figure 10. The four bytes of one column are treated as coefficients of a four-term polynomial over GF(256^4). This poly-

nomial is multiplied modulo $x^4 + 1$ (denoted by $\otimes$) with a fixed polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. Again, the transformation can be written in matrix notation, see Figure 9. Let $s'(x) = a(x) \otimes s(x)$:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \leq c < 3.$$

Figure 9: MixColumns in matrix notation

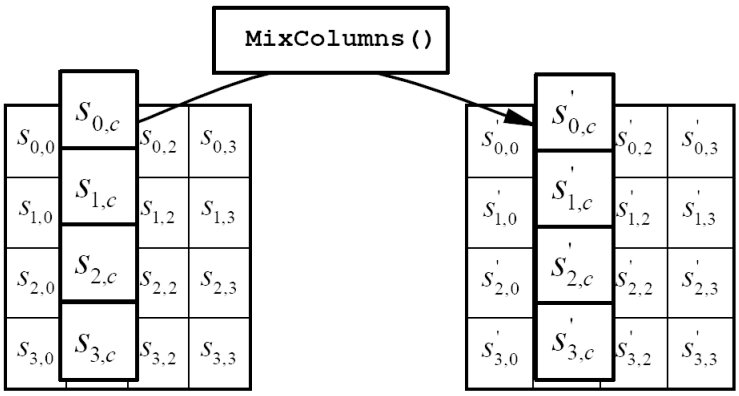


Figure 10: MixColumns processes *State* columns one-by-one [6]

AddRoundKey This operation adds a 128-bit *RoundKey*, that is generated by the key schedule (see next paragraph), to the *State*. As addition here means *XOR*, AddRoundKey can be denoted as $State = State \oplus RoundKey$. Note that during the initial Round Key addition, the originally supplied key data is used. In the subsequent 9 + 1 rounds, AddRoundKey adds derived round keys.

KeyExpansion This routine generates 11 RoundKeys that are necessary for a complete AES encryption or decryption operation. The RoundKeys are iteratively derived from the supplied key data according to the pseudo code in Figure 11. Note that in encryption mode, the derived RoundKey for the

initial Round Key addition is the supplied key.
SubWord() operates on a four-byte word and applies the SubBytes() transformation to each of the four bytes. RotWord() transforms a four-byte array (a_0, a_1, a_2, a_3) into (a_1, a_2, a_3, a_0), thus it is a cyclic left shift. The round constant Rcon[i] contains the following four bytes (x^{i-1}, {00}, {00}, {00}), that include powers of $x = \{02\}$.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
word temp
i = 0
while (i < Nk)
w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
i = i+1
end while
i = Nk
while (i < Nb * (Nr+1)]
temp = w[i-1]
if (i mod Nk = 0)
temp = SubWord(RotWord(temp)) ⊕ Rcon[i/Nk]
else if (Nk > 6 and i mod Nk = 4)
temp = SubWord(temp)
end if
w[i] = w[i-Nk] xor temp
i = i + 1
end while
end
```

Figure 11: Pseudo code for key scheduling algorithm

3.2 Statistics

All statistical measures we use in the course of this thesis are standard and well described in virtually every introduction to statistics or complete math reference book, e.g. [30, 31, 32]. Nevertheless, we give a short review of the measures, for completeness.
Let n denote the number of realisations x_i ($i = 1, \ldots, n$) of a random variable X.

3.2.1 Measures of central tendency

Arithmetic mean (Average) $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1+x_2+...+x_n}{n}$
Note that the arithmetic mean converges to the expectation value $E(X)$ (law of large numbers).

Median If n is odd and $n = 2k+1$, then $M = x_{k+1}$, thus the middle element that appears in a sorted list of all x_i.
If n is even and $n = 2k$, then $M = \frac{x_k+x_{k+1}}{2}$, thus the arithmetic mean of the two middle elements of a sorted list of all x_i.

3.2.2 Measures of dispersion

Variance $\sigma^2 = E(X-\overline{x})^2$
is a measure of the dispersion of X from its mean $\overline{x}$. If the probability distribution of X is unknown, the sample variance $\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2$ can be computed from realisations x_i to estimate σ^2.

Covariance $cov_{xy} = E\left((X-\overline{x})(Y-\overline{y})\right)$
is a measure for the linear dependency between X and Y. A positive (resp. negative) covariance indicates that if X increases Y tends to increase (resp. decrease). If the probability distributions of X and Y are unknown, the sample covariance $\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})$ can be computed from realisations x_i and y_i to estimate cov_{xy}.

Correlation $\rho(X,Y) = \frac{cov(X,Y)}{\sigma_X\sigma_Y}$
is the covariance normalized to be in the range $[-1,1]$. One advantage of the correlation measure is that it allows an interpretation of the "strength" of the linear dependecy. One has $\rho(X,X) = 1$.

Covariance matrix Let $X = (X_1, X_2, \ldots, X_m)$ be a random vector.
$cov(X) := (cov(X_i, X_j)) \quad \in \mathbb{R}^{m\times m}$ with $i,j = 1,\ldots,m$
The covariance matrix comprises all pairwise covariances of the random vec-

tor's elements. For example: let $A = (X, Y, Z)$, then

$$cov(A) = \begin{pmatrix} \sigma_X^2 & cov(X,Y) & cov(X,Z) \\ cov(Y,X) & \sigma_Y^2 & cov(Y,Z) \\ cov(Z,X) & cov(Z,Y) & \sigma_Z^2 \end{pmatrix}$$

3.2.3 Measures for the difference of two sets

T-Test The T-Test is a measure for the statistical difference of means of two random variables. It is an advanced tool to compare two random variables as it does not only consider the distance of their averages but as well their dispersion. Let X, Y be two random variables with n_x and n_y known realizations, then

$$t = \frac{\overline{x} - \overline{y}}{\sqrt{\frac{\sigma_X^2}{n_x} + \frac{\sigma_Y^2}{n_y}}}$$

3.2.4 Selected distributions

Gaussian distribution Let $\sigma > 0$. X has a Gaussian (normal) distribution with parameters $\overline{x}$ and σ^2 if X has density $f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\overline{x})^2}{2\sigma^2}\right)$. Figure 12 shows Gaussian distributions for $\overline{x} = 0$ and several choices of σ^2 (1; 1,5; 2; 3).

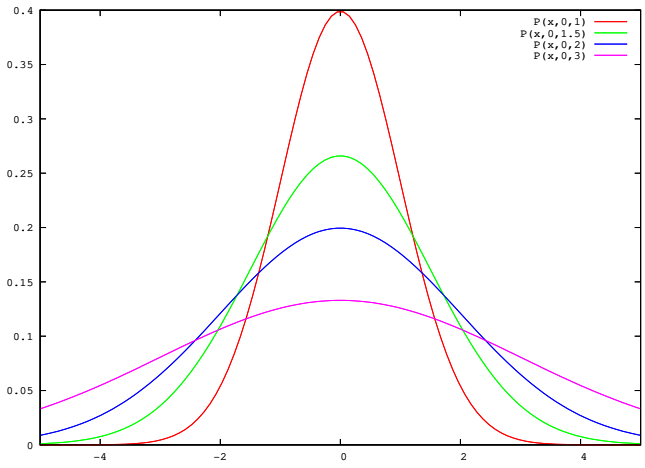


Figure 12: Selected Gaussian distributions

Multivariate Gaussian distribution Let $X = (X_1, X_2, \ldots, X_m)$ be a vector of m jointly normally distributed random variables with the vector of means $\overline{X}$ and covariance matrix $\sum$. $|\sum|$ denotes the determinant of $\sum$. The joint probability density of X's elements is given by

$$f(X) = \frac{1}{\sqrt{(2\pi)^m |\sum|}} \exp\left(-\frac{1}{2}(X - \overline{X})^T \sum^{-1} (X - \overline{X})\right)$$

Figure 13 shows the probability densities of two jointly normally distributed random variables. In a) they are not correlated at all and in b) they are correlated with $\rho = -1$.

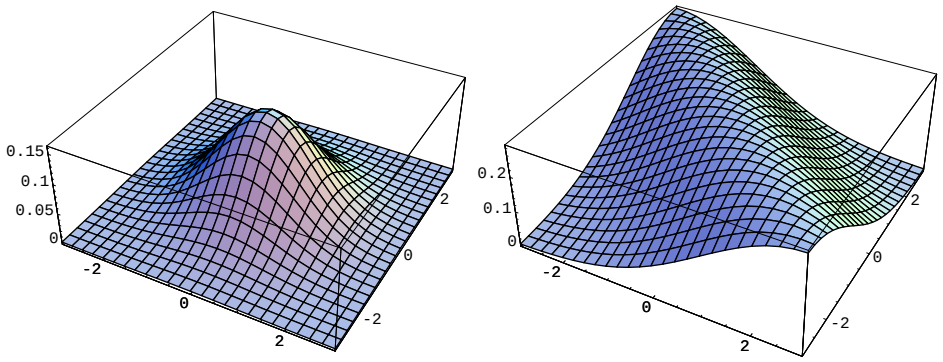


Figure 13: a) uncorrelated and b) correlated Multivariate Gaussian densities

3.3 Template Attacks

Template Attacks [10] were introduced by S. Chari, J.R. Rao, and P. Rohatgi at CHES 2002. Clearly, the Template Attack complies with our definition of Two-Step SCAs.

In this chapter we describe Template Attacks closely to the original paper. Commencing with the attack's elementary idea, we give a rough review of its procedure in Section 3.3.1, underlying assumptions on the side channel in Section 3.3.2 and a step-by-step explanation with detailed information in Sections 3.3.3 and 3.3.4.

Note that we omit the expand and prune strategy that is part of the original publication as it is more related to the field of stream ciphers. It was not required for the side channel cryptanalysis of the AES. Furthermore, a unique key-dependent computation of a cryptographic device will be referred to as an *operation* in the remainder of this chapter in order to be consistent with [10]. In the context of stream ciphers where the Template Attack was originally motivated, the authors gave the example of executing the same code for different values of key bits to elucidate the word operation. For our experiments in the context of the AES block cipher, we identify an operation by the value of the AES State array after the initial Round Key addition, that is $x \oplus k$. The motivation of this decision is provided on page 31, see "Equal Images under different Subkeys".

Unlike One-Step SCAs that use some hundred side channel samples to eliminate the noise contained in each sample by computing averages (DPA, DEMA), the Template Attack extracts and (exclusively) uses the noise to learn about the implementations characteristics. More precisely: the template attack uses precise multivariate characterizations of the (deterministic component of the) noise and precise estimations of the intrinsic signal within side channel samples from device A to classify given samples from device B. The authors argue that especially for CMOS devices these characterizations are an extremely powerful tool (cf. [10]).

3.3.1 Template Attacks in a nutshell

In the profiling step, a training device A is used to generate representations of the signal and multivariate characterizations of the occurring noise in side channel measurements for all possible operations of the device. A pair of models for signal and noise is referred to as the *template* of the operation. In the classification step, the maximum-likelihood approach is used to compare the noise within one sample from device B to these templates in order to deduce the performed operation. To successfully determine the underlying operation is equivalent to key disclosure, because the operation is $x \oplus k$ and

x is known.

3.3.2 Model for side channel observables

The observable, i.e. side channel information, is modeled as a combination of an intrinsic signal, intrinsically generated noise and ambient noise. When side channel samples of several invocations of the same operation are considered, their signal component is the same whereas the noise component is best modeled as a random sample from a noise probability distribution that depends on the environment and operational conditions.
Obviously, an attack's success rate is limited to some bounds by the implementation of an algorithm on a particular device. A *perfect* model of the noise probability distribution would lead to a success rate of the Template Attack that meets these bounds in theory. Nevertheless, approximations such as the multivariate Gaussian model are ought to perform well in practice (cf. [10]).

3.3.3 The profiling step

For each of the K possible operations[9] of the device a large number L (e.g. one thousand) of side channel samples has to be obtained using device A. The subsequent steps determine the K templates from these samples, one for each operation.

Intrinsic signal The first part of each template is a precise representation of the intrinsic signal that can be observed during an invocation of the corresponding operation. The Template Attack's empiric approach to generate this representation is to suppress the noise within the appropriate samples and use the remaining signals to determine the typical signal. Both is achieved at the same time by computing the average M_i from the L samples that correspond to operation O_i for all $i = 1, \ldots, K$ operations. In ideal case, M_i contains in fact a very precise estimation of the intrinsic signal as the noise components average out at 0 and the remaining average signal is a

[9] recall that an operation is defined by $x \oplus k$ in the AES context

very good estimator in the absence of outliers[10].

The next step is optional but highly advisable in practice because it significantly reduces the attack's costs (processing time, storage) with only a small loss of accuracy. It is almost sure that not all moments covered by the side channel samples are of interest to an attacker, thus this step deals with identification and selection of *interesting* points in time.
Computing pairwise differences between the average signals M_i yields a curve that shows large spikes at points where the underlying signals (and thus operations) differ. Only these points are of interest to an attacker. The Gaussian model applies to W points (P_1, ... ,P_W) that were chosen along the spikes. The original publication does not declare how exactly these points should be chosen. Our insights on this issue are given in Section 5.1.2, Step 4.

Multivariate noise model The second part of each template is a precise characterization of the noise that can be observed during an invocation of the corresponding operation. The Template Attack assumes that the noise approximately has a multivariate Gaussian distribution, hence the covariance matrix $\sum_{N_i}$ describing the probability density of the noise is computed consecutively for all operations in this step.
First of all, the noise within the samples has to be extracted. For each operation O_i all L noise vectors $N_i(\cdot)$ of the samples need to be computed. Thereby one W-dimensional noise vector $N_i(T)$ of sample T is the difference of the sample T and the average signal M_i at the chosen W instants. More formally:

$$N_i(T) = (T(P_1) - M_i(P_1), \ldots, T(P_W) - M_i(P_W)) \tag{5}$$

Then the noise covariance matrix $\sum_{N_i}$ can be computed using the L noise vectors $N_i(T)$ for each operation O_i [11]. The elements of the covariance matrix

[10] The *Median* might be used to gain better results while its computation is more costly.
[11] Recall from 3.2.2 that a covariance matrix consists of the pairwise covariances of a random vector's elements.

$\sum_{N_i}$ are defined as:

$$\sum\nolimits_{N_i}[u,v] = cov(N_i(P_u), N_i(P_v)), \tag{6}$$

where u and v denominate two of the W chosen points in time, e.g. the pair $\{P_u, P_v\}$. Note that computation of $\sum_{N_i}[u,v]$ for $u \leq v$ suffices because $\sum_{N_i}$ is a symmetric matrix.
After this step all K templates $(M_i, \sum_{N_i})$ are computed. The expected signal for operation O_i is M_i and the noise probability distribution is given by the W-dimensional multivariate Gaussian distribution $p_{N_i}(\cdot)$. The probability of observing a noise vector $\mathbf{n}$ is:

$$p_{N_i}(\mathbf{n}) = \frac{1}{\sqrt{(2\pi)^W |\sum_{N_i}|}} \exp\left(-\frac{1}{2}\mathbf{n}^T \sum\nolimits_{N_i}^{-1} \mathbf{n}\right), \quad \mathbf{n} \in \mathbb{R}^W \tag{7}$$

where $|\sum_{N_i}|$ denotes the determinant of $\sum_{N_i}$ and $\sum_{N_i}^{-1}$ its inverse.

3.3.4 The classification step

The situation of the classification step is as follows: an attacker obtains one side channel sample S from the device under attack (B) and wants to find out which of the K possible operations it descends from.
This step primarily comprises a maximum likelihood hypothesis test, hence it is less costly in computational efforts. For each operation O_i the probability of observing S if indeed it originated from O_i is computed. To do so, first the W-dimensional noise vector $\mathbf{n}$ within S has to be extracted by subtracting M_i from S at the W selected instants (M_i is part 1 of template$_i$). Then equation (7) can be evaluated for $\mathbf{n}$ using $\sum_{N_i}$ (part 2 of template$_i$) to get the actual probability. Finally, the operation O_i that yields maximum probability is selected.
As one is rather interested in a ranking of the candidates O_i than in the actual probabilities, the formula can be simplified by disregard of constant

terms in (7). If so, the operation that *minimises*

$$\ln\left(|\sum\nolimits_{N_i}|\right) + \mathbf{n}^T \sum\nolimits_{N_i}^{-1} \mathbf{n}, \quad \mathbf{n} \in \mathbb{R}^W \tag{8}$$

is selected.

3.3.5 Use of Template Attacks against AES

In the original paper the authors describe an "expand and prune" strategy that is particularly useful when analyzing side channel samples of stream ciphers. If the attacker uses this strategy, profiling and classification build a recurring cycle which means in particular that the vast effort of the profiling step cannot be precomputed.
In contrast, if the attacked key is known to be sufficiently small or assailable in such blocks (e.g. this is true for all block ciphers with the property that each block of the first roundkey only depends on one original key block) the profiling can be done independently before or after obtaining S from the device under attack. This might be of importance for such cases where the period between obtaining the sample S and key recovery is a critical factor. For example: to recover an 128-bit AES key in the way we present in this thesis an attacker has to compute "only" $2^8 \cdot 16 = 4096$ instead of 2^{128} templates, which would be clearly infeasible. The attacker can precompute all these templates and - after obtaining S - immediately start the classification step which takes only a few seconds, even on an ordinary home computer.

3.4 Stochastic Model

The Stochastic Model [11] was published by W. Schindler, K. Lemke, and C. Paar at CHES 2005. It is the third attack in the class of Two-Step SCAs since it definitely shows the necessary properties.

In this section we present the Stochastic Model close to the original contribution. As in the previous section we begin with the attack's fundamental idea after what we give a short review of its overall procedure in Section 3.4.1

and explain the mathematical model in Section 3.4.2. Detailed information on the approach is then given in Sections 3.4.3 and 3.4.4.

The Stochastic Model, as the name leads one to assume, is a fairly sophisticated approach that uses several statistical methods and is based on a well defined, elaborated mathematical model. However, for the sake of comprehensibleness we will skip all formal proofs and theoretic considerations that we find unnecessary for the reader to understand the attacks's concept. Therefore we refer the interested reader to [11] for proofs, details and deeper understanding.
Furthermore, several aspects that we bring forward might sound redundant, like repetitions from Section 3.3. We do so anyway, rather than pointing to the Template Attack, in order to give a complete review of the Stochastic Model that can be read on its own. On the other hand we omit the minimum-principle approach at key extraction because it was already expected and experimentally proven to be less efficient in the original publication. The Stochastic Model aims at block ciphers, its adaptability to stream ciphers is unknown.

The Stochastic Model extracts and uses the noise contained within side channel samples to disclose secret information. This stands in sharp contrast to all known One-Step SCAs which see noise as a hindrance. More precisely: the Stochastic Model uses **one** precise multivariate characterization of the (nondeterministic) noise in conjunction with an approximation of the deterministic signal in a chosen vector subspace to classify given samples. The authors argue that due to approximation of the deterministic signal the Stochastic Model's success rate is bounded upwards by the Template Attack which estimates the signal as good as possible. On the other hand, far less[12] measurements would be required in the profiling step. Our investigation of efficiency differences and explanatory approaches are provided in Section 5 and thereafter.

[12] savings in the dimension of up to 100 are mentioned in the case of AES

3.4.1 Stochastic Model in a nutshell

In the profiling step, a training device A is used to approximate its real side channel leakage function in a chosen vector subspace and to generate a multivariate characterization of the occurring noise. The training curves are assumed to represent all key dependencies uniformly distributed, in the concrete case of AES that is they are uniformly distributed for $x \oplus k$. In the classification step, the maximum likelihood approach is used to compare the sample(s) from device B to the approximated leakage function in order to deduce its key dependency.

3.4.2 The mathematical model

The model's underlying setting is as follows: an attacker has access to side channel samples (e.g. of an encryption) and a part of the corresponding plaintext[13] $x \in \{0,1\}^p$. His task is to disclose a subkey $k \in \{0,1\}^s$.
For any given instant t (covered by the samples) the measurement is regarded as a realization of the random variable

$$I_t(x,k) = h_t(x,k) + R_t \tag{9}$$

that is composed of two parts. The first part $h_t(x,k)$ denotes the portion of the sample that depends on x and k and will be referred to as the deterministic part. The second part R_t denotes the portion that does not depend on x and k and will be referred to as the random part. Since both portions (thus the entire sample) additionally depend on the instant t the random variable could be expanded to the discrete function $I(x,k,t) = h(x,k,t) + R(t)$ to cover this fact. Nevertheless, to be consistent with [11] we will stick to the notation in (9) and consider single instants where not indicated differently.

The deterministic part can be seen as an unknown mapping $h_t : \{0,1\}^p \times \{0,1\}^s \to \mathbb{R}$ that assigns a real value, e.g. power consumption, to each combination of plaintext and key bits. $\mathcal{F} := \{h' : \{0,1\}^p \times \{0,1\}^s \to \mathbb{R}\}$ denotes

[13] adaption to known-ciphertext scenarios is feasible

the infinite set of such mappings. The most precise and costly approach to attack an implementation clearly aims at finding $h' \in \mathcal{F}$ such that $h' = h_t$. In other words, the adversary has to find all 2^{p+s} assignments to uniquely identify $h' = h_t$. In case of an attack against the AES targeting one Sbox at a time, which means $x, k \in \{0,1\}^8$, this leads to 2^{16} mappings[14] per Sbox. The Stochastic Model significantly reduces this effort by approximating $h_t(x,k)$ in a vector subspace and by exploiting an elementary property of the physical observables, if applicable. A full description of these steps is beyond the scope of this thesis, therefore we will skip their derivation and straight provide the results (all details and proofs can be found in [11]).

The idea is as follows: an adversary determines a small u-dimensional vector subspace $\mathcal{F}_{u;t} \subset \mathcal{F}$ which contains a mapping h^* that either is indeed the searched mapping h_t or at least sufficiently close to it. In this subspace, he only has to find u assignments to uniquely identify h^*.
$\mathcal{F}_{u;t}$ is regarded as the set of all mappings $h' \in \mathcal{F}$ that can be expressed in the u-dimensional vector subspace spanned by u known functions $g_{jt} : \{0,1\}^p \times \{0,1\}^s \to \mathbb{R}$. In formal notation:

$$\mathcal{F}_{u;t} := \left\{ h' : \{0,1\}^p \times \{0,1\}^s \to \mathbb{R} \quad | \quad h' = \sum_{j=0}^{u-1} \beta_j \cdot g_{jt} \right\} \text{ with } \beta_j \in \mathbb{R} \tag{10}$$

The success rate of the attack is strongly coupled to the choice of $\mathcal{F}_{u;t}$ thus the functions g_{jt}. Once they are chosen, the coefficients $\beta_0, \ldots, \beta_{u-1}$ can be estimated for each instant t. Apparently, the number of required samples in the profiling step increases with the number of dimensions u, if the same level of precision is aspired for the β_{jt}. One might see this as a trade off problem for a **fixed** number of samples in the profiling step: a small number of dimensions u reduces the searchable space, which might exclude good candidates $h' \in \mathcal{F}$ but gives better estimators for the best h^* still included in $\mathcal{F}_{u;t}$; a large number of dimensions u will more likely include a very good

[14] This is the approach of a naïve Template Attack. However, our Template Attack requires 2^s assignments, see EIS on page 31.

candidate h^* but its estimators will be less precise.

On the choice of the functions g_{jt}: if the physical observables show a certain property (cf. "Equal Images under different Subkeys (EIS)" in [11]), an almost lossless reduction of $\mathcal{F}$ is possible. With *lossless* we address the fact that this reduction decreases the number of candidates h' without "loosing" a single one. This is possible because the nature of the candidates is changed. Consider an arbitrary set V and a (surjective) mapping $\phi(x,k) \to V$ for which the images of $\phi(\{0,1\}^p, k) \subseteq V$ are equal for all subkeys $k \in \{0,1\}^s$. The deterministic portion of the samples $h_t(x,k)$ is said to have the property EIS, if h_t can be expressed as a function of ϕ, i.e. $h_t = \phi \circ \overline{h}_t$ for an appropriate mapping $\overline{h}_t$. If h_t has (or is assumed to have) the invariance property EIS, the authors suggest to select functions g_{jt} that can be expressed as $g_{jt} = \phi \circ \overline{g}_{jt}$ with $\overline{g}_{jt} : V \to \mathbb{R}$. This leads to the following expression for the best estimator h_t^*:

$$h_t^* = \phi \circ \sum_{j=0}^{u-1} \beta_{jt} \cdot \overline{g}_{jt}(y) \quad \beta_{jt} \in \mathbb{R}, y \in V. \tag{11}$$

The gain of exploiting the EIS property can be illustrated as in Figure 14.

$$2^{p+s} \quad \xrightarrow{EIS} \quad V \subseteq 2^p \quad \xrightarrow{loss} \quad u$$

Figure 14: Reduction of the vector space exploiting the EIS property

The decision, whether this property should be assumed or not, can be made considering only the abstract algorithm. Nevertheless, due to lack of a perfect model of the physical device, a proof of the property can only be adduced empirically by experiment.

3.4.3 The profiling step

The number of side channel samples that is necessary for the profiling step is linked to the number of dimensions of the vector space in which the adversary approximates the real leakage function (see above). However, the original

contribution does not quote a specific number but compares several choices in terms of their efficiency at key extraction. Our studies on this are given in Section 6.4.2. Let us assume the adversary decides for a u-dimensional vector space and obtains two sets of N_1 and N_2 side channel samples using device A. The subsequent steps compute the approximators h_t^*, in other words a function $h^*(x, k, t)$, for the deterministic portion of the side channel information and the multivariate characterization of the noise R_t.

Approximation of h_t^* Let $x_j \in \{0,1\}^p$ $(j = 1, \ldots, N_1)$ be the known parts of the plaintext and $i_t(x_j, k)$ be the side channel measurement at instant t that corresponds to x_j.
The approach uses the *Least Squares Method* to find an optimal approximator $h^* \in \mathcal{F}_{u;t}$ of i_t. For any approximator $h' \in \mathcal{F}_{u;t}$ the sum of squared deviations from the real leakage function i_t can be denoted by

$$\sum_{j=1}^{N_1} (i_t(x_j, k) - h'(x_j, k))^2 = \|\mathbf{i}_t - A\mathbf{b}\|^2. \tag{12}$$

As i_t resp. A and $\mathbf{i}_t$ are taken for granted (see below) the optimal approximator h^* that minimises the left hand side of (12) is uniquely identified by any vector $\mathbf{b}$ that minimises the right hand side of (12). $\mathbf{b}$ can be found by evaluating

$$A^T A\mathbf{b} = A^T\mathbf{i}_t \quad \Rightarrow \quad \mathbf{b} = (A^T A)^{-1} A^T \mathbf{i}_t \tag{13}$$

if $A^T A$ is invertible.

The adversary begins with determining the $(N_1 \times u)$ - matrix A. Each matrix element a_{ij} $(i = 1, \ldots, N_1$ and $j = 0, \ldots, u-1)$ is defined as $a_{ij} := g_j(x_i, k)$ resp. $g_j(\phi(x_i, k))$ exploiting EIS.
Hence the adversary traverses all N_1 plaintexts whereat he evaluates the u functions g each time. Then he computes the $(u \times u)$ - matrix $A^T A$ and (if it is regular) the $(u \times N_1)$ - system matrix $S = (A^T A)^{-1} A^T$. The system matrix is time invariant and needs to be computed only once wheres the vector $\mathbf{i}_t$ and hence the vector $\mathbf{b}$ have to be found separately for each instant t.

$$A = \begin{pmatrix} g_1(\phi(x_1,k)) & g_2(\phi(x_1,k)) & \cdots & g_u(\phi(x_1,k)) \\ g_1(\phi(x_2,k)) & g_2(\phi(x_2,k)) & \cdots & g_u(\phi(x_2,k)) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(\phi(x_{N_1},k)) & g_2(\phi(x_{N_1},k)) & \cdots & g_u(\phi(x_{N_1},k)) \end{pmatrix}$$

Figure 15: Design Matrix A exploiting EIS property

The column vector $\mathbf{i}_t$ is defined as $(i_t(x_1,k),\ldots,i_t(x_{N_1},k))^T$. For each instant t, the adversary extracts the measurement for t from all N_1 samples and computes $\mathbf{b} = S \cdot \mathbf{i}_t$. Every $\mathbf{b}$ has dimension u and contains the coefficients $(\beta'_0,\ldots,\beta'_{u-1})$ for the optimal approximator $h_t^* = \sum_{j=0}^{u-1} \beta_j \, g_j(x,k)$.

The next step is optional but highly advisable in practice in order to reduce the computational effort of the attack with only a small loss of accuracy. It is almost sure that only some instants covered by the side channel samples actually carry valuable information, therefore this step deals with the identification and selection of *interesting* points in time $t_1,\ldots,t_m$. The authors do not make a statement on how these points can be found in the theoretic part of the publication. Yet, the experimental analysis part shows several approaches based on the euclidean vector norm $\|(b_{0,t},\ldots,b_{u-1,t})\|$ and compares them in terms of efficiency at key extraction. For the moment we simply go on with the set $t_1,\ldots,t_m$ provided by an oracle. Our experiences in this field are given in Section 5.1.2, Step 4.

Multivariate Characterization of $\mathbf{R}_t$ In the Stochastic Model the noise within the side channel is assumed to be independent of x,k, i.e. non-deterministic, and to roughly show properties of a multivariate Gaussian distribution. The subsequent steps compute the covariance matrix C that characterizes the noise probability density.
Let R_t denote a random vector $(R_{t_1},\ldots,R_{t_m})$ with $t_1,\ldots,t_m$ being the selected instants. The adversary uses the approximators h_t^* to extract the noise within the $j = 1,\ldots,N_2$ side channel samples $i_t(x_j,k)$ from the second set. More precisely, he computes N_2 noise vectors of dimension m whereas

each noise vector is the difference between a sample and the corresponding approximated deterministic part. More formally:

$$R_t = i_t(x_j, k) - h_t^*(x_j, k) \tag{14}$$

Then, the covariance matrix C is computed[15] using the N_2 noise vectors R_t. Each matrix element c_{ij} $(1 \leq i, j \leq m)$ is defined as

$$c_{ij} = cov(R_t(i), R_t(j)) \tag{15}$$

with i and j being two of the m chosen points in time. Note that the covariance is symmetric and hence computation of all c_{ij} for $i \leq j$ suffices.
The computation of the matrix C completes the profiling step. The deterministic part $h_t(x, k)$ of the side channel leakage is approximated by $h_t^*(x, k)$ and the random noise $\mathbf{R}_t$ is characterised by the m-dimensional probability density f_C.

$$f_C : \mathbb{R}^m \to \mathbb{R} \quad f_C(z) = \frac{1}{\sqrt{(2\pi)^m |C|}} \exp(-\frac{1}{2} z^T C^{-1} z), \quad z \in \mathbb{R}^m \tag{16}$$

where $|C|$ denotes the determinant of C and C^{-1} its inverse.

3.4.4 The key extraction step

This step basically comprises a maximum likelihood test hence it is less costly in computational efforts than the profiling step. The setting for the key extraction step is as follows: the adversary had (limited access) to device B and obtained N_3 side channel samples $s_t(x_j, k^\circ)$ $(j = 1, \ldots, N_3)$ with known plaintexts x_j. Now he wants to disclose the secret key k° that was used by device (B).

By assumption the noise in the side channel did not change, i.e. the noise vector $z_{tj} = s_t(x_j, k^\circ) - h_t^*(x_j, k)$ has a multivariate Gaussian distrib-

[15] Recall from 3.2.2 that a covariance matrix consists of the pairwise covariances of a random vector's elements.

ution with covariance matrix C. For each key hypothesis k the probability of observing z_{tj} if k is indeed the right key can be evaluated with (16). The adversary combines these probabilities for all N_3 samples, i.e. he evaluates

$$\prod_{j=1}^{N_3} f_C(z_{tj}) = \prod_{j=1}^{N_3} f_C(s_t(x_j, k^\circ) - h_t^*(\phi(x_j, k))) \tag{17}$$

for all subkeys $k \in \{0,1\}^s$, and decides for the key k that maximises the term.

As one is rather interested in a ranking of the key hypothesis than the actual probabilities, the formula can be simplified by disregarding constant terms in (16). If so, the adversary decides for the key k that *minimises*

$$\sum_{j=1}^{N_3} z_j^T C^{-1} z_j. \tag{18}$$

3.5 Compendium of differences

The following table shows the fundamental differences in the approaches of both attacks. For correctness we denote, that (non)deterministic shall be understood as (non) dependent on the relevant data, e.g. the key.

Sample portion	Template Attack	Stochastic Model
signal	deterministic, estimated → 256 average signals	deterministic, approximated → 9 sub-signals
noise	deterministic, characterised → 256 cov matrices	non-deterministic, characterised → one cov matrix

Table 2: Fundamental differences between Template Attacks and the Stochastic Model

Remarks of the original authors:

> The Template Attack extracts all possible information available in each sample and is hence the strongest form of side channel attack possible in an information theoretic sense given the few samples that are available. [10]

> Though our efficiency at key extraction is limited by template attacks profiling is much more efficient which is highly relevant if the designer of a cryptosystem is bounded by the number of measurements in the profiling step. [11]

Our insights on the efficiency of both attacks are given in Section 5 and thereafter.

4 Acquisition

This section deals with the acquisition of side channel samples. First, we will give a general outline of the side channel measurement work flow. Then, in the succeeding sub-sections, we will go over the different steps we had to process and provide detailed information.

4.1 Side Channel measurement work flow

In general, Side Channel Cryptanalysis requires a large number of precise measurements where each acquired sample needs to be stored for later analysis and all measurements should ideally be done in the same fashion and in a non-changing environment. Obviously, these requirements in terms of precision, constance, and speed can be faced by a high degree of automation. Usually, a Personal Computer is a central point of a setup and used to operate and coordinate all other devices as well as to store obtained measurement data. A digital oscilloscope (scope) is needed to perform the actual measurements and the necessary A/D conversion. Depending on the type of attack, one or several probes are required to link the scope to the device that is examined. For completeness we mention that further material might be necessary in order to put the devices into operating state and link them to the PC (power supplies, Smartcard reader, boards for I/O communication). Figure 60 shows the relations between the devices and the sequence of operations (within one measurement cycle).

Step 1 Initially, the scope needs to be setup with several parameters, like for example duration and resolution, and calibrated, such that the measurement range is used to full capacity. Normally, both can be done either manually or by software tools that communicate with an interface of the scope.

Step 2 An instruction to carry out the operation once is sent to the cryptographic device. Optionally, additional commands are sent to the device in order to change parameters as for example the plaintext.

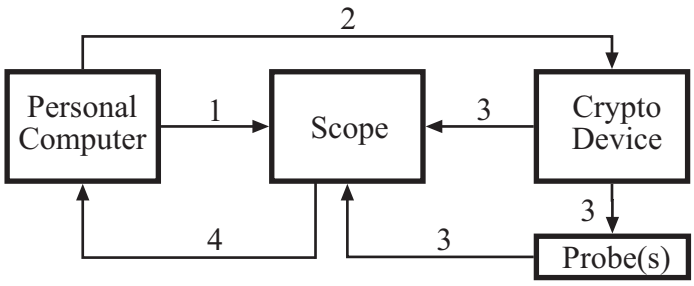


Figure 16: Side Channel measurement setup

Step 3 During the execution of the operation, the side channel information is acquired by the scope. It is highly advisable to synchronise the scope's and the device's operation, e.g. by means of a trigger signal, in order to limit storage efforts.

Step 4 Once the execution of the operation is finished by the device and the sample is recallable from the scope's memory, it is transferred to the PC for storage.

For further measurements, Steps 2 to 4 can be repeated in a cyclic way.

4.2 Micro Controller, AES Implementation

We used a so called *Funcard* [17] for our side channel measurements. In opposition to "normal" Smartcards, the microcontroller (μc) and the memory module are not monolithicly integrated, but wired, and embedded into the card body. Our Funcard contains an 8-bit RISC ATMega 163 μc [18] in Harvard architecture concept (separation of data and program memory). The internal memory of the μc is limited to 16KB FlashROM program memory, 512 Bytes E^2PROM (permanent) memory, and 1KB SRAM (non-permanent) data memory.

To bypass the daunting task of low-level I/O-programming we availed us of the Simple Operating System for Smartcard Education (SOSSE) [24], a modular open source operating system. It abstracts from the hardware layer

and provides the ISO standardized T=0 protocol (ISO 7816 [25]) for byte wise half duplex transmission of Application Protocol Data Units (APDUs).

The AES encryption algorithm was implemented straight forward according to [6] in Assembly language using the Atmel AVR Studio 4 IDE [26] with only one modification. We combined the SubBytes and ShiftRows transformation so that the result of SubBytes would be directly inserted into the State array at the right position. Furthermore, we added some lines of code that generate a trigger signal on the Smarctcard's I/O pin just before the initial Round Key addition begins, to synchronise the scope.
After we verified that the code works correctly, we integrated the AES encryption into SOSSE so that it could be invoked by an APDU command. Furthermore we added an APDU command to load a 128-bit key into the E^2PROM. Then, SOSSE was compiled with the avr-gcc [27] open source cross compiler and the hex-files programmed onto the card (FlashROM and E^2PROM) with the MasterCrd and MasterBurner software [28].

4.3 Acquisition setup, Parameters for measurements

In this section, we provide details about our acquisition setup and the parameters we used.

Digital Oscilloscope Agilent Infinium 54832D Mixed Signal Oscilloscope; key data: Bandwidth 1 GHz, Channels 4+16, max. sample rate 4GSa/s, Acquisition memory 2Mpts/channel [19]

Probe Agilent 1165A Miniature passive probe; key data: Division ratio 10:1, Input resistance 10MΩ [20]

EM Probe Langer EMV Technik near field probe RFU 5-2; key data: acquires surface and circular magnetic fields (see Figure 58 in appendix A), Resolution ∼ 5mm [21]

Preamplifier Langer EMV Technik preamplifier PA 303 connected to the EM probe; key data: Amplifying 30dB, Noise figure 4,5dB [22]

Card Reader CHIPDRIVE micro card reader, dismantled to ease access (see Figure 59 in appendix A) and to connect external, low-noise power supply (see below); key data: ISO 7816-3 conform, clock frequency 3.57 MHz [23]

DC Power Supply Statron direct current power supply

To dismantle the Smartcard reader eased access to its internal wiring. We soldered a 47Ω resistor into the ground of the card's power feed and used an Agilent Probe (channel A) to measure the potential drop over the resistor. The usual Smartcard power supply, which is done by connecting the reader to a PC's serial port, was disconnected and replaced with the Statron DC power supply. According to [29], the card's supply voltage is 5V ± 10% and the maximal current consumption is 10mA. Hence, the voltage drop over the resistor would not excess 10mA · 47Ω = 470mV. Accordingly, we set the Statron device to supply 5.5V.
We connected an additional wire to the I/O pin of the card reader and directed its other end to the outside so that the second Agilent probe (channel B) can detect the trigger signal.
The card holding socket was attached headfirst, so that the contact area of the Smartcard pointed down. Preliminary tests showed that the contact area partly shields EM emanation. Furthermore, we unsealed the card reader's backside in the chip area to bring the EM probe as close to the chip as possible. The EM probe was connected to the scope (channel C) through the preamplifier. Figure 60 in appendix A shows our overall setup.

The scope was set to obtain samples of 20000 points at a rate of 200MS/s from channels A and C after detecting the trigger signal on channel B. Each point was sampled in 8-bit resolution. The 20000 points cover 100μs which matches the time that the card needs to compute the initial Round Key addition and the first normal round of an AES encryption. With the cards clock frequency being 3.68MHz, one clock cycle takes ∼0.27μs and for each of the ∼360 covered clock cycles ∼55 points are sampled.

4.4 Fixed key, Variable key

Altogether, we carried out three sets of measurements for our experiments. As our main focus is Two-Step Side Channel Cryptanalysis, we obtained at first a pair of sets of measurements. One large set for the profiling step and one smaller set for the classification step.

Fixed key We began with carrying out a pair of sets of measurements using a *fixed* key. Our approach aims at recovering the 128bit AES key k in portions of 8 bits, thus we represent the full key as a concatenation of 16 subkeys k_j $(j = 0, \ldots, 15)$. The plaintext x is represented in the same manner, i.e. $x = (x_1, \ldots, x_{15})$. The first set of measurements is supposed to serve the profiling step. We used a fixed key k and plaintexts x, randomly chosen from a uniform distribution, to obtain (following the recommendations in [10]) ≈1000 samples per operation $(x_j \oplus k_j)$, a total of 231448 samples. For the second set, we loaded a different key k^* onto the Smartcard and again used random plaintexts x to obtain 3000 samples. This set is supposed to serve the classification step.
Figure 17 shows the distribution of the samples within the first profiling set with respect to x_0, i.e. the first plaintext byte. Since this set was obtained using a fixed key, the distribution could be permuted by $\oplus k_0$ to then illustrate the distribution of $x_0 \oplus k_0$.

Variable key Because of several observations that we made while working on the samples of the first pair of sets of measurements (see Section 5, step 4, observation 1), we decided to carry out a third measurement set. This time the plaintext x and the key k were chosen randomly before each invocation of the encryption operation. As before, we obtained a set of ≈256000 samples for the profiling step. Figure 18 depicts the distribution of the samples within the second profiling set with respect to $x_0 \oplus k_0$.

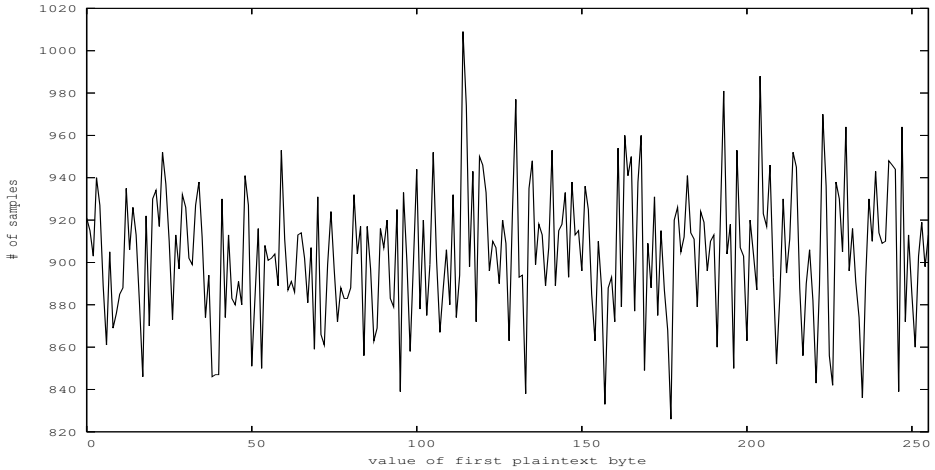


Figure 17: Distribution of first profiling set with respect to x_0

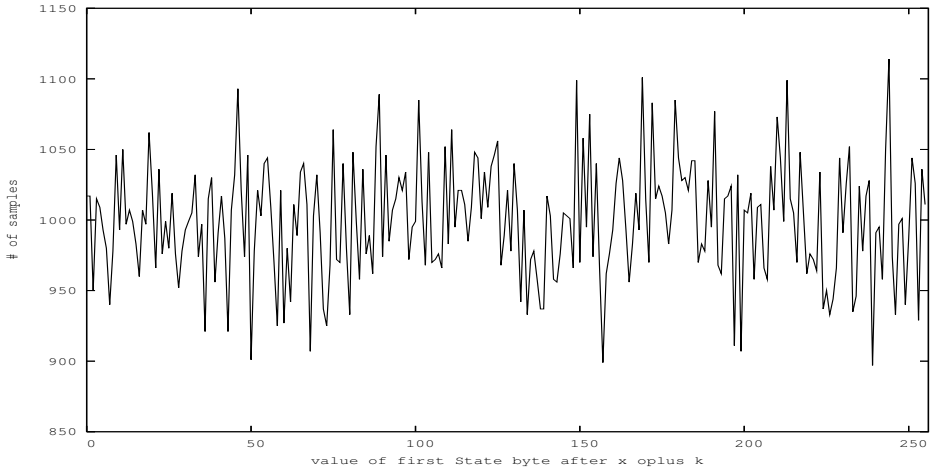


Figure 18: Distribution of second profiling set with respect to $x_0 \oplus k_0$

5 Experimental Results - fixed key

In this section we outline our implementations of both attacks, provide concrete results, and report in detail on the experience we gained working on the samples of the first pair of sets of measurements.
All programs were implemented in C-language and all computations on side channel samples were carried out in 64-bit floating point precision[16].
Section 5.1 deals with the Template Attack and Section 5.2 with the Stochastic Model. In Section 5.3, we compare both attacks.

5.1 Template Attack

The authors of [10] claim, that the Template Attack is the "*strongest form of side channel attack possible in an information theoretic sense*". The results we present later on will, depending on various circumstances, support and disprove this statement. Furthermore, the authors argue that within their assumptions (see later on) Template Attacks are superior to SPA- and DPA-style attacks, as in the former case all available information in each side channel is used. We agree on the superiority of Template Attacks, see Section 3.5.

5.1.1 Remarks and Improvements (1)

Remark 1 (concerning the profiling step): We point out that if the samples were obtained in a way as described in Section 4 (fixed key), device A does not need to be programmable and even knowledge of the employed key is unessential. The amount of samples an adversary possesses after the profiling step is far more than enough to disclose the employed key in a DPA attack. In fact, we were able to extract the full 128-bit key k both using one thousand samples from the power channel and using one thousand samples from the EM channel. Figure 19 exemplarily shows the resulting peaks in the correlation curves for the correct subkey k_0.

[16] data type *double* on a 32-bit Intel Processor with a ported GCC

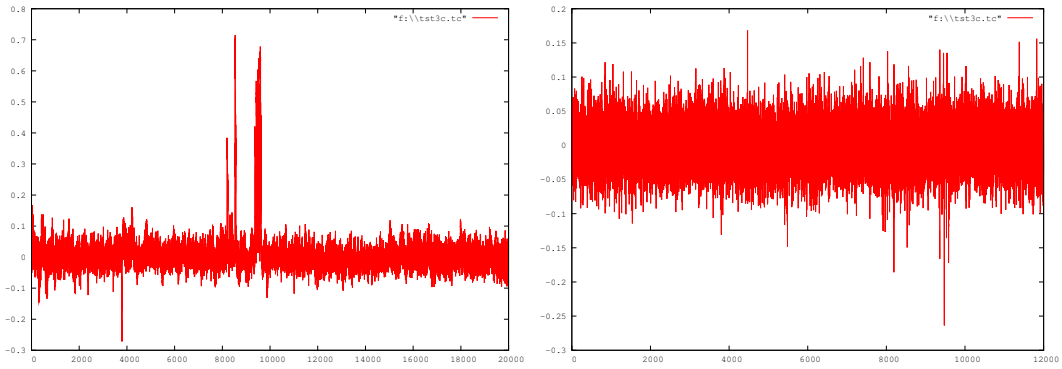


Figure 19: Correlation curves for the correct subkey $k_0 = 0x3C$ on the power channel (left) and the EM channel (right)

This observation weakens the assumptions about an attacker's minimal powers in order to successfully mount a Side Channel Attack against the AES. The remaining minimum requirements are:

- availability of a device A that is identical to device B so that an adversary can attach the necessary probes and carry out the required amount of measurements
- **either** knowledge of the plaintexts if their distribution may be assumed to be approximately uniform
- **or** ability to chose plaintexts so that they are approximately uniformly distributed.

IMPROVEMENT 1 (concerning the choice of *interesting* points in time): Carrying out preliminary tests we quickly discovered that the sum of pairwise differences of the average signals, i.e. $\sum_{i,j=0}^{K} m_i - m_j$ for $j \geq i$, is not an optimal basis for choosing the *interesting* points in time. This is due to the fact, that positive and negative differences add up to 0. While this effect is desirable to filter eventually present noise, it hides as well valuable peaks that derive from significant signal differences with alternating algebraic sign. Therefore we implemented two more measures that served as the basis for this choice.

The first one computes the sum of absolute pairwise differences of the average signals $\sum_{i,j=0}^{K} |m_i - m_j|$ for $j \geq i$ so that the hiding effect does not emerge anymore for the cost of a noise floor $\not\approx 0$.
The second one computes the sum of squared pairwise differences of the average signals $\sum_{i,j=0}^{K} (m_i - m_j)^2$ for $j \geq i$ so that large differences become magnified while very small differences become reduced. Again, a noise floor $\not\approx 0$ is the price.
Figures 20 and 21 (see pp. 49) depict the three measures for the cases that the average signals m_i were computed from 231448 power channel samples.

IMPROVEMENT 2 (concerning the classification step): The original Template Attack only provides a sample classification strategy based on a single available sample. While this seems to be a realistic scenario in the context of stream ciphers[17], the situation might be less tight in the context of block ciphers. To pay tribute to the eventuality that several samples are available in the classification step, we developed a *differential* strategy that processes several samples.
For every available sample, we compute the probabilities that the sample represents this or the other operation O_i, e.g. $x_0 \oplus k_0$, in the "traditional" way. Then, we purge the offsets between these probability distributions that are caused by the different plaintexts so that the probabilities are now assigned to key hypothesis and "in line". Finally, we add up the probability distributions and select the key hypothesis that yields maximum probability.
More formally: Let S_n and x_n $(n = 1, \ldots, m)$ denote the available samples resp. the corresponding known plaintexts and O_i $(i = 0, \ldots, 255)$ denote the operations $x_n \oplus k^\circ$ where k° is the unknown key. First, we compute:

$$\begin{array}{cccc} prob(S_1 \to O_0), & prob(S_1 \to O_1), & \ldots, & prob(S_1 \to O_{255}) \\ prob(S_2 \to O_0), & prob(S_2 \to O_1), & \ldots, & prob(S_2 \to O_{255}) \\ \vdots & \vdots & \vdots & \vdots \\ prob(S_m \to O_0), & prob(S_m \to O_1), & \ldots, & prob(S_m \to O_{255}) \end{array}$$

[17] [35] presents an amplified attack for the case of several available samples

Then, each line n of the array is permuted by $\oplus x_n$ so that it represents key hypothesis k_i instead of operation hypothesis $x_n \oplus k^\circ$. Note that each line n of the array is permuted by its corresponding x_n (which probably differ from line to line) so that the columns in the array below do not match the columns in the array above. One column represents the correct key hypothesis $k = k^\circ$, but its position in the array is unknown so far.

$$\begin{array}{ccccccc} prob(S_1 \to k_0), & \ldots, & prob(S_1 \to k^\circ), & \ldots, & prob(S_1 \to k_{255}) \\ prob(S_2 \to k_0), & \ldots, & prob(S_2 \to k^\circ), & \ldots, & prob(S_2 \to k_{255}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ prob(S_m \to k_0), & \ldots, & prob(S_m \to k^\circ), & \ldots, & prob(S_m \to k_{255}) \end{array}$$

Finally, all probabilities pointing to a unique key hypothesis are added up

$$\sum_{n=1}^{m} prob(S_n \to k_0), \quad \ldots, \quad \sum_{n=1}^{m} prob(S_n \to k^\circ), \quad \ldots, \quad \sum_{n=1}^{m} prob(S_n \to k_{255})$$

and the hypothesis yielding maximal probability is selected.

$$max\left(\sum_{n=1}^{m} prob(S_n \to k_i)\right) \to k \stackrel{?}{=} k^\circ$$

This strategy has the advantage that the presence of *difficult* samples can be compensated. Even if the correct key is not the best candidate for any single sample, chances are good that their combined probability distributions guide to the right decision.
We give a simplified example: Let the correct subkey $k^\circ = 01_2$ and

$x \oplus k^\circ \to$	00_2	01_2	10_2	11_2
S_1	0,1	0,08	0,02	0,03
S_2	0,1	0,02	0,01	0,08

be the individual probability distributions derived from two samples S_1 and S_2 with corresponding plaintexts $x = 00_2$ and $x = 10_2$. S_1 selects $x \oplus k^\circ = 00_2$ which leads to the guess that $x \oplus k^\circ \oplus x = k^\circ = 00_2$, which is wrong. In the same manner, S_2 selects $x \oplus k^\circ = 00_2$ which leads to the guess that $x \oplus k^\circ \oplus x = k^\circ = 10_2$, which is again wrong. Purging the offsets and adding

the probability distributions yields

$k^\circ \rightarrow$	00_2	01_2	10_2	11_2
S_1	0,1	0,08	0,02	0,03
S_2	0,01	0,08	0,1	0,02
$\sum$	0,11	0,16	0,12	0,05

which leads to the correct guess $k^\circ = 01_2$.

5.1.2 Implementation

The Template Attack was implemented closely to the description in 3.3. After some preliminary tests, the implementation was modified according to improvements 1 and 2. In the following we describe our implementation step-by-step and provide data examples to illustrate the procedure. Additionally, we introduce several abbreviations which will be used throughout the sequel.

Profiling step The Template Attack aims at generating a template, i.e. an estimation of the signal and a characterisation of the noise, for each key dependent *operation*. As mentioned before, we define a unique *operation* by the value of one selected byte in the AES *state* array s after the initial Round Key addition, e.g. $s_{0,0} = x_0 \oplus k_0 \in \{0,1\}^8$ and hence we generate 256 templates. The implementation of this byte-wise attack can attack any byte in s but for the sake of clarity we restrict our attention to $s_{0,0}$.

The first step aims at generating indexes of the N_1 available samples. For each value of x_0 we create an **index** file that points to all samples that correspond to it. Note that indexing in this manner leads to the same partitioning as indexing for $x_0 \oplus k_0$ or even S-box$(x_0 \oplus k_0)$ because k_0 is fixed. Hence the index names could be permuted to represent the other arrangements. For example: index$_{255}$ corresponds to $x_0 = 255$ but as well to $operation_{255 \oplus k_0}$ and to S-box$(255 \oplus k_0)$. Furthermore, an additional file **distribution** is created that contains the length of each index file, i.e. the amount of curves that correspond to each plaintext x_0. Figure 17 in Section 4.4 illustrates the content of such a distribution file. The implemented function is named

`assign_directory_contents(char *byte)`. It creates all the files mentioned above for the selected **byte** $\in (0, \ldots, 15)$. We focused on byte = 0 and will from now on omit this parameter.

Step 2 computes the average signal for each operation (part 1 of each template). We implemented a function `average_curves(*value,*no_of_files)` that computes the average of all no_of_files samples which correspond to x = value and used it in a loop as follows:

```
for 0 ≤ i < 255
begin
   value ← i
   no_of_files ← distribution of byte[value]
   average_curves(byte, value, no_of_files)
end
```

The resulting average signals will be referred to as $\mathbf{average}_{value}$.

Step 3 computes the basis for the choice of interesting points in time. As described in [10] we compute the sum of pairwise differences (**sod**) of the average signals. Additionally, with respect to improvement 1, we compute the sum of absolute pairwise differences (**soad**) and the sum of squared pairwise differences (**sosd**). All three measures are computed by the function `compute_differences()` at the same time as follows:

```
for 0 ≤ k < data_length (each point in time)
begin
   for 0 ≤ i < 256
   begin
      for i+1 ≤ j < 256
      begin
         sod[k] = sod[k] + average_i[k] - average_j[k]
         soad[k] = soad[k] + |average_i[k] - average_j[k]|
         sosd[k] = sosd[k] + (average_i[k] - average_j[k])^2
      end
   end
end
```

Figure 20 shows the resulting curves sod (red, lower curve) and soad (green, upper curve) for $N_1 = 231448$ samples from the power channel. One can notice that soad magnifies peaks ($\sim$ 4000) and even more important makes peaks visible (8000-10000) that do not appear in sod. On the other hand the "strongest peak to noise floor" ratio decreases from $\sim$30:1 for sod to $\sim$ 4:1 for soad. Figure 21 shows the same soad curve (green, lower curve)

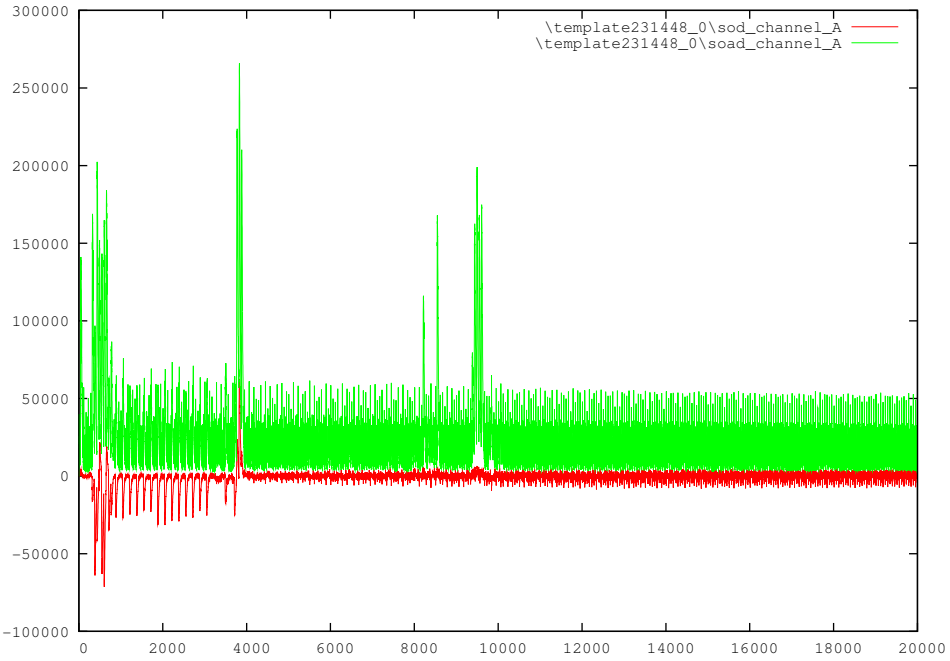


Figure 20: **sod** and **soad** curves for $N_1 = 231448$ (power channel) as functions of time

together with the resulting sosd curve (red, upper curve). Note that the scale for the vertical axis changed. One can see that sosd magnifies peaks even stronger than soad and that the "strongest peak to noise floor" ratio increases from $\sim$ 4:1 for soad to $\sim$ 20:1 for sosd.

Step 4 comprises the actual choice of interesting points in time. [10] gives no more advise on how to choose the points than "*identify and select only points at which large differences show up*". We developed a function `find_`

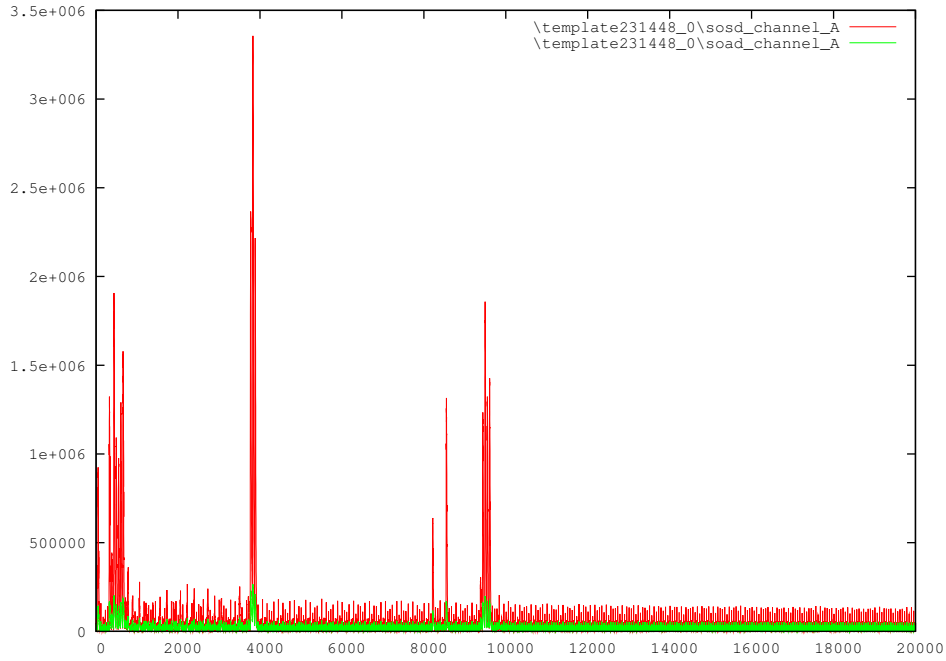


Figure 21: **soad** and **sosd** curves for $N_1 = 231448$ (power channel) as functions of time

`points_of_interest(curve,p)` that recursively identifies the **p** strongest peaks in curve $\in$ {sod, soad, sosd} and stores their positions. The list of these points will be referred to as **poi[]** (Points Of Interest). Figure 22 depicts the resulting selection for p = 20 and curve = sosd (derived from $N_1 = 231448$ power samples). Apparently, the strong peak in the area of 4000 in Figure 21 actually consists of three peaks that now become visible because of the limited range of the x axis. As expected, the function chose the 20 highest points on the curve. From the ratio

$$\frac{\text{sampling frequency of the scope}}{\text{clock frequency of the card}} = \frac{200MS/s}{3.68MHz}$$

we know that each clock cycle of the card is represented by ~55 points on the curve. (If the clock frequency is unknown, it can be easily estimated by applying an FFT to a sample, see Figure 23 where the highest peak indicates a clock frequency of ~3.6 MHz.) One can deduce that the three peaks in Figure 22 represent three clock cycles but that only two of them are covered

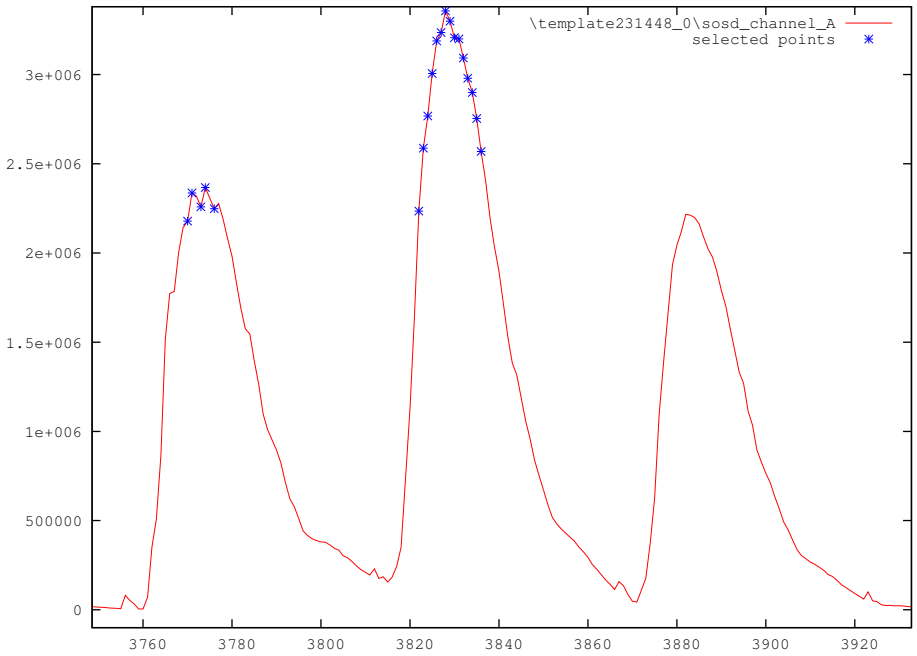


Figure 22: **sosd** curve for $N_1 = 231448$ (power channel) and the selected 20 points

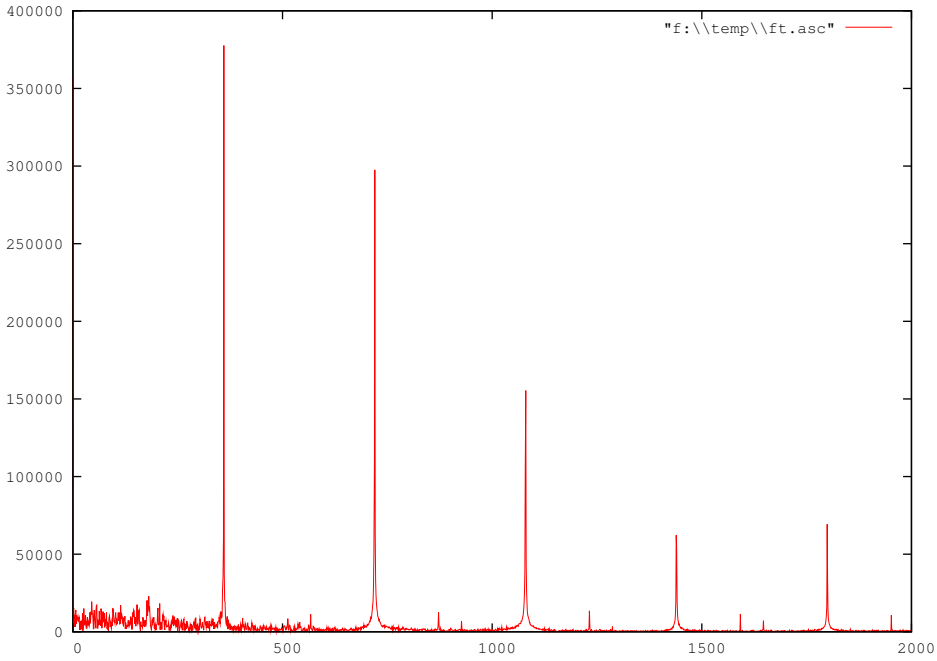


Figure 23: Discrete Fourier Transform of a sample from the power channel

by the selected points. We assumed this to be suboptimal and modified the function such that it would not consider the nearest $\delta = 54$ points to the left and right of every selected point. In other words, the function would now select at most 1 point per clock cycle.

Observation 1

From preliminary tests and consideration of both the AES' structure and the samples' nature we concluded that although all three differential curves show peaks in the area 0-1000 we should prevent the selection of those instants. Recall that each average curve represents a unique value of x_0 and that our implementation of the AES begins with the initial Round Key addition which computes $x \oplus k$ where k is fixed. Given that the differential curves are computed as described above, e.g. $\sum_{i=0,j>i}^{255} average_i - average_j$, their part that covers the initial Round Key addition basically represents

$$\begin{array}{lll}
\overbrace{(0 \oplus k_0) - (1 \oplus k_0)}^{average_0 - average_1} & + \overbrace{(0 \oplus k_0) - (2 \oplus k_0)}^{average_0 - average_2} & + \ldots \quad + \overbrace{(0 \oplus k_0) - (255 \oplus k_0)}^{average_0 - average_{255}} \\
& + \overbrace{(1 \oplus k_0) - (2 \oplus k_0)}^{average_1 - average_2} & + \ldots \quad + \overbrace{(1 \oplus k_0) - (255 \oplus k_0)}^{average_1 - average_{255}} \\
& \ddots & \ddots \quad \vdots \\
& & + \overbrace{(254 \oplus k_0) - (255 \oplus k_0)}^{average_{254} - average_{255}}
\end{array}$$

so that the peaks in this area are only caused by the differences in x_0. In other words: due to the fact that for our sample set 1, sorting the samples by x_0 yields the same partitioning as sorting by $x_0 \oplus k_0$, we get noticeable peaks in the differential curves at those points in time when the algorithm processes x_0 or $x_0 \oplus k_0$. Obviously, the peaks for x_0, that is during the initial Round Key Addition, do not indicate *operation*-dependent differences in which we are interested. This hypothesis was empirically proven by experiment, see Section 5.4 and result table 6 at the end of this section. We modified the function accordingly such that it would ignore the first 3300 points[18].

[18] This boundary was estimated using the sosd curve computed for the fifteenth byte in the AES State array and verified by the clock cycle count of the simulated μc.

Furthermore, we added a feature that would prevent the selection of points in the noise floor. More precisely, the function fixes a noise limit at 10% of the highest peak's value and considers any point below this limit as not selectable. Figure 24 depicts the revised selection for curve = sosd (N_1 = 231448) and p = 9, which is the maximum number of points that fulfill all requirements.

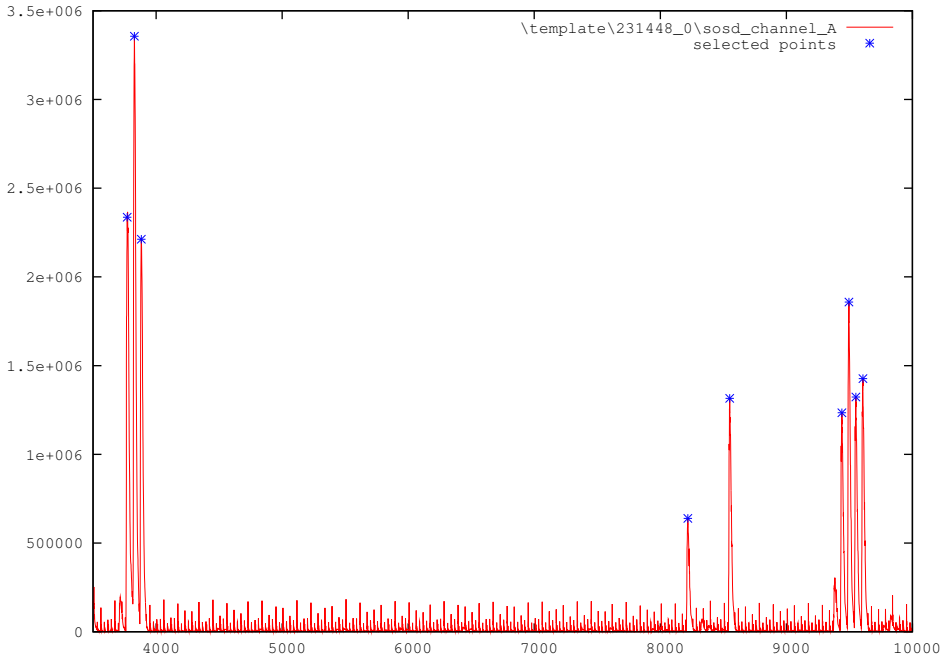


Figure 24: **sosd** curve for N_1 = 231448 (power channel) and the selected 9 points

The steps 5 and 6 jointly perform the characterisation of the noise. The former is a preparatory step that supplies required data to the latter which generates the noise characterisation.
Step 5 extracts the noise within all samples that correspond to one operation. We implemented a function `compute_noise_vectors(*value,*no_of_files)` that extracts the noise vectors from the `no_of_files` samples pointed to by index_{value}. For each sample, it computes the difference of the $\textbf{average}_{value}$ and the sample at the p selected points. An example is depicted in Figure 25. It shows a sample from the power channel (red, upper curve)

corresponding to $x_0 = 106$, the appropriate average$_{106}$ (green, middle curve), and the entire extracted noise (blue, lower curve). Note that we actually compute and store only the noise values at the selected points, indicated by a blue cross.

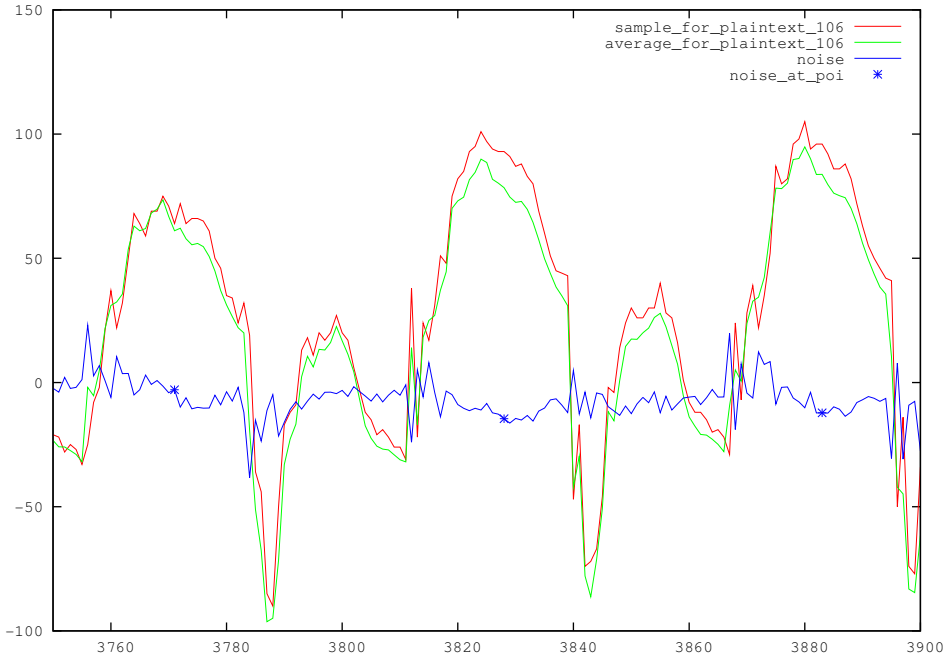


Figure 25: A sample, the corresponding average, and the extracted noise

Once the loop within the function terminates, it has generated a $(p \times no\_of\_files)$ - array that holds the noise values, see Figure 26.

sample / poi[]	0	1	...	8
1	-2.9067702553	-14.5615982242	...	0.1398446171
2	...	...	...	...
⋮	⋮	⋮	⋱	⋮
901	...	...	...	...

Figure 26: Layout of the "noise array"

Step 6 generates a $(p \times p)$ - covariance matrix (part 2 of each template) that characterises the noise corresponding to one operation with the help of the noise array. Recall that, given a vector $\mathbf{R} = (R_0, \ldots, R_8)$ of random

variables, the matrix layout is

$$cov(\mathbf{R}) = \begin{pmatrix} \sigma^2_{R_0} & cov(R_0, R_1) & \dots & cov(R_0, R_8) \\ cov(R_1, R_0) & \sigma^2_{R_1} & \dots & cov(R_1, R_8) \\ \vdots & \vdots & \ddots & \vdots \\ cov(R_8, R_0) & cov(R_8, R_1) & \dots & \sigma^2_{R_8}) \end{pmatrix},$$

that the matrix is symmetric, and that one matrix element, e.g. $cov(R_0, R_1)$, is computed as $cov(R_0, R_1) = E((R_0 - \overline{R_0})(R_1 - \overline{R_1}))$. As the true values of $\mathbf{R}$ are unknown (we only know no_of_files realisations of the random vector) the matrix elements have to be computed using the sample covariance formula, see Section 3.2. Under the assumption, that the noise vector has a multivariate Gaussian distribution (cf. [10]) with mean vector $\overline{\mathbf{R}} = (0, \dots, 0)$ the formula can be simplified so that

$$cov(R_0, R_1) = \frac{1}{n-1} \sum_{i=1}^{n} (R_{0_i} - \overline{R_0})(R_{1_i} - \overline{R_1}) = \frac{1}{n-1} \sum_{i=1}^{n} R_{0_i} R_{1_i}.$$

Function `compute_cov_matrix(*value,*no_of_files)` computes the matrix element by element as follows:

```
for 0 ≤ i < p
begin
  for i ≤ j < p
  begin
  temp ← 0
    for 0 ≤ k < no_of_files
    begin
      temp ← temp + noise_array[k][i] * noise_array[k][j]
    end
    temp ← temp / (no_of_files - 1)
    cov_matrix[i][j] ← temp (assign value to matrix element i,j)
    cov_matrix[j][i] ← temp (and to element j,i)
  end
end
```

The two outermost loop variables i, j consecutively select all elements in the upper triangle of the covariance matrix resp. select each pair of columns

of the noise array *once*. The innermost loop computes the covariance for each selection (the loop variable runs through all rows of the noise array) which is then assigned to both appropriate positions in the symmetric covariance matrix. Once all three loops have terminated, the matrix is stored as covariance_matrix$_{value}$.
Steps 5 and 6 are repeatedly invoked in a loop so that a covariance matrix is computed for each operation. The profiling step is completed with the termination of this loop.

Classification step The goal of the classification step is to classify a single side channel sample S from device B. This means to correctly deduce the operation that was executed by B while the sample was measured from the sample's properties. The approach is as follows: for each template$_{value}$, the noise in S is extracted as in step 5 using the estimated signal average$_{value}$. Then, the probability of observing such noise if indeed it derives from operation$_{value}$ is computed with (7) resp. (8). We used a sample from our measurement set 2 to serve as S.

Step 7 randomly selects one of the 3000 samples in measurement set 2 to serve as S.

Step 8 extracts the noise within sample S for the considered hypothesis. Function `compute_noise_vector_for_classification(*hypothesis,*S)` does this in exactly the same way as it was done in step 5. It computes the difference of average$_{hypothesis}$ and S at the p selected points in time.

Step 9 then computes the probability of observing such noise under the given hypothesis using the appropriate covariance matrix (CM) and stores the probability for later comparison. We developed a set of functions to compute each probability as provided in the pseudo code of step 10, innermost loop.

Step 10 identifies the best hypothesis. Function `sort_descending(prob-`

`abilities, identifiers)` sorts the probabilities in descending order while simultaneously sorting the hypothesis' identifiers in the same order so that the identifier of the best hypothesis ($x_0 \oplus k_0$) is the first one in the list. The key hypothesis is then be computed by XOR addition of the plaintext x_0 that corresponds to the sample.

Steps 7, 8, 9, and 10 are (repeatedly) invoked by a superior function `classify()` to compute the 256 probabilities and choose the best candidate.

```
rand ← gen_rand()
S ← load_curve(rand)
x_0 ← load_curve_plaintext(rand)
for 0 ≤ hypothesis < 256
begin
   compute_noise_vector_for_classification(*hypothesis, *S)
   CM ← load_covariance_matrix(hypothesis)
   Det ← compute_determinant(CM)
   CM ← compute_inverse(CM)
   probability[hypothesis] ← compute_probability(*CM, *Det, *S)
end
sort_descending(probabilities, identifiers)
selection ← identifier[0] ⊕ x_0
```

To attain a certain degree of precision for the success rate, we decided to classify 1000 randomly selected samples and count in the variable *correct*, how often the best candidate is indeed the correct key. The success rate is then given by $\frac{correct}{1000}$. Note that this is quite a strong measure as we do not consider any other case where the correct key is for example on the 2^{nd} or 3^{rd} place in the hypothesis ranking. Doing so would certainly increase the success rate but soften its significance.

```
correct ← 0
for 0 ≤ k < 1000
begin
  rand ← gen_rand()
  S ← load_curve(rand)
  x_0 ← load_curve_plaintext(rand)
  for 0 ≤ hypothesis < 256
  begin
    compute_noise_vector_for_classification(*hypothesis, *S)
    CM ← load_covariance_matrix(hypothesis)
    Det ← compute_determinant(CM)
    CM ← compute_inverse(CM)
    probability[hypothesis] ← compute_probability(*CM, *Det, *S)
  end
  sort_descending(probabilities, identifiers)
  selection ← identifier[0] ⊕ x_0
  if (selection = key) correct ← correct + 1
end
output ← correct / 1000
```

With respect to improvement 2, we futher modified the function `classify()` so that an analysis based on several N_3 samples S would be possible. By the way we implemented this functionality we would be able to observe both the change of success probability caused by an increased number of samples and the development of the separation between the different candidates. Note that the probability distributions now need to be permuted in order to purge the effect of the random plaintexts x_0 and retrieve a probability distribution for key hypothesis, see improvement 2. We extended the array probability[hypothesis] so that it would keep track of the characteristics mentioned above: probability[key_hypothesis][no_of_sample]. For the first sample S the probabilities are computed as before and stored after permutation. For each succeeding sample, we compute its probability distribution, permute it, add it to the one of the preceding sample, and store it in the appropriate line of the array. After the N_3 samples have been classified, the aggregated probabilities in the last row of the array are sorted to identify the best key hypothesis in the usual way.
The final version of `classify(*byte)` is:

```
correct ← 0
for 0 ≤ k < 1000
begin
  for 0 ≤ i < no_of_files_for_classify
  begin
    rand ← gen_rand()
    S ← load_curve(rand)
    x_0 ← load_curve_plaintext(rand)
    for 0 ≤ hypothesis < 256 (hypothesis = x_0 ⊕ k_0)
    begin
      compute_noise_vector_for_classification(*hypothesis, *S)
      CM ← load_covariance_matrix(hypothesis)
      Det ← compute_determinant(CM)
      CM ← compute_inverse(CM)
      if (i == 0)
       probability[i][hypothesis ⊕x_0] ← compute_probability(*CM, *Det, *S)
      else
       probability[i][hypothesis ⊕x_0] ← probability[i-1][hypothesis⊕x_0]
       + compute_probability(*CM, *Det, *S)
    end
  end
  sort_descending(probability, identifiers)
  selection ← identifier[0]
    if (selection = key) correct ← correct + 1
end
output ← correct / 1000
```

Once this loop terminates, the classification step is complete and all results are stored for later comparison.

In fact we invoked the function classify() several times per classification cycle for a varying number of points p. Knowing that `find_points_of_interest()` finds the interesting points in a descending order of impact, we were interested in quantifying this impact. The guiding question was: Is a higher number of points always desirable or is it possible, that an additional point, with minor impact, worsens the success rate?
The way we implemented all steps above, in particular the handling of the covariance matrices, makes it possible to go into this matter with very little

effort. We use a global variable P to control how many of the p points that were found are actually considered during the probability computation. By decreasing the value of P we are able to omit the last $p - P$ points that were found.

5.1.3 Results for various parameter settings

Carrying out the experimental analysis we noticed that there are three parameters that have major impact on the success rate of a Template Attack against AES. They are:

1. the number of curves available during the profiling step $\mathbf{N_1}$
2. the number of interesting points that can be found in the profiling step **p** respectively that are used in the classification step **P**
3. the number of curves available in the classification step $\mathbf{N_3}$.

We test all combinations $N_1 \times$ P $\times N_3$ for $N_1 \in \{10k, 20k, 25k, 30k, 40k, 50k, 231448\}$, P $\in \{9, 6, 3\}$, and $N_3 \in \{1, 2, 5, 10\}$.
The procedure of the experiments is always the same: we choose a value for N_1 and carry out the profiling step to generate templates based on the p points that are found. Then we carry out the classification step using N_3 = 10,5,2,1 samples from the device under attack at which, each time, we start using all P = p points and then reduce their number by 3 until this number becomes smaller than 1. Below we only present the results reflecting those parameter values, that we find most significant. Appendix B provides all result tables for the Template Attack.

Due to our inability of presenting the results in a four-dimensional table, which would be the optimal choice, we have to decide for another strategy. We present several tables where each table represents a fixed value N_1 and variations of P and N_3. N_1 varies from table to table.

Table 3 presents the results based on the best possible choice for N_1. The selection algorithm found 9 points whose positions in time are given in the

$N_1 = 231448$	$p = 9$	channel = power
poi[]	3771, 3828, 3883, 8218, 8551, 9440, 9496, 9551, 9607	

	SR of real classification				SR of trial classification			
P \ N_3	1	2	5	10	1	2	5	10
9	20,5	56,0	97,8	99,9	19,7	60,3	98,6	100,0
6	14,2	43,1	94,3	100,0	16,7	47,2	97,6	99,9
3	8,5	29,2	82,1	99,2	9,7	25,0	81,1	99,6

Table 3: Success rates (SR) for $N_1 = 231448$ and channel = power

second line of the table. This distribution of selected points will be referred to as *optimal distribution* in the further analysis of the Template Attack. From the two blocks containing the success rates for real and trial classification, one can observe that samples from device B are classified as good as the samples from device A that were used during the profiling step. This indicates an optimal profiling. Apparently, both parameters (P, N_3) have direct impact on the success rate. The correlation of a parameters value and the success rate is best described as logarithmic[19], which is also true for simultaneous changes of both parameters' values. The parameter N_3 has a stronger influence on the success rate than P as one can observe that for each value of P a success rate $\sim 100\%$ can be attained while a fixed (small) N_3 sets an upper limit for the succes rate.

Table 4 presents the results for $N_1 = 50000$. The first striking observation is, that although N_1 has been reduced by a factor > 4 the success rates' order of magnitude is the same. The selection algorithm found 7 points from the optimal distribution, one that is slightly displaced but still in the correct processor cycle (9552 instead of 9551), and one that is not related to the optimal distribution at all and in fact is a bad selection. This is caused by an increased noise floor in sosd which in turn is caused by more noise in the individual averages. Figure 27 illustrates, how the peak at 8218 "sinks" in the noise floor and a slightly higher "noise peak" at 19235 is selected instead.

[19] We rather describe a tendency of an imagined curve's run than precisely comparing it to a given logarithm.

$N_1 = 50000$	$p = 9$	channel = power

poi[]	3828, 3771, 3883, 9496, 9607, 9552, 8551, 9440, 19235

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	15,9	45,8	92,5	99,6		23,5	61,9	99,4	100,0
6	13,5	44,4	93,3	99,7		15,4	52,9	97,6	100,0
3	9,5	27,5	77,2	98,2		8,0	28,6	83,0	98,5

Table 4: Success rates for $N_1 = 50000$ and channel = power

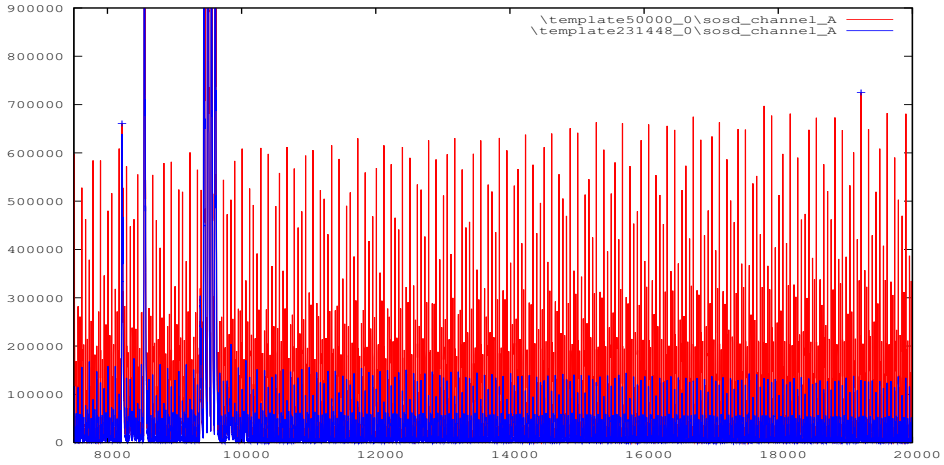


Figure 27: sosd curves for $N_1 = 50000$ (red) and $N_1 = 231448$ (blue), power channel

But apparently, this mis-selection does not have a big effect on the success rates, in particular for increased values of N_3. The badly selected point is only used for computing success rates in the case P = 9. By comparing this line of the table to the appropriate line of the table above one can see that the success rates of the trial classification did not change whereas those of the real classification decreased by up to 23% especially for $N_3 = 1$ and 2. This happens due to the reason that the well selected 8 points, constantly guiding to the right selection, outweigh the single mis-selection, which guides to random key candidates so that the probabilities do not add up for $N_3 > 1$.

Table 5 which provides the results for $N_1 = 25000$ shows that a further bisection of the number of curves used in the profiling step has a strong impact on the results. The selection algorithm found only one point from

$N_1 = 25000$	$p = 9$	channel = power

poi[]	3828, 3772, 3884, 9509, 9620, 8564, 17790, 19235, 15289

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	3,4	9,5	32,3	61,5		13,4	42,8	93,7	100,0
6	6,6	21,0	65,9	96,3		11,4	35,9	89,0	100,0
3	8,5	24,4	78,1	97,6		9,0	30,7	85,4	99,0

Table 5: Success rates (SR) for $N_1 = 25000$ and channel = power

the optimal distribution, four points are more or less displaced but still in the correct processor cycle, and four points are not related to the optimal distribution at all. Figure 28 illustrates the further increased noise floor in sosd that is responsible for the poor selection. By looking at the success rates

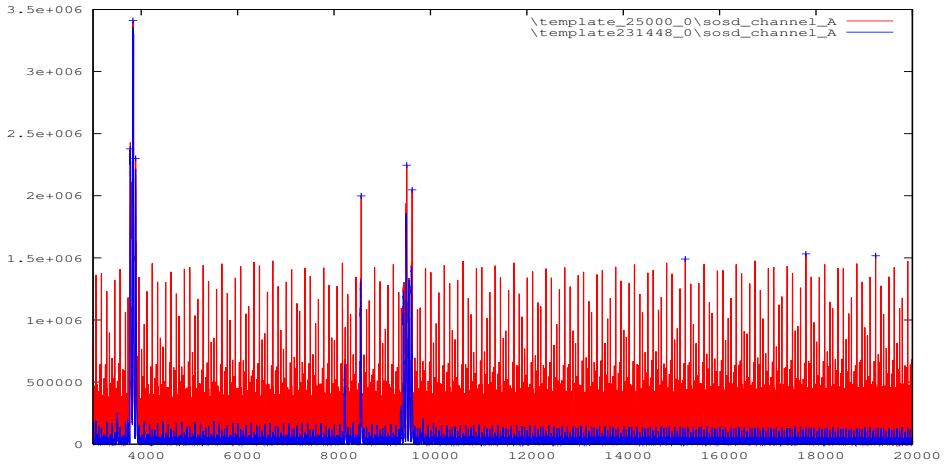


Figure 28: sosd curves for $N_1 = 25000$ (red) and $N_1 = 231448$ (blue), power channel

one can observe that the correlation of the parameter P and the success rate has changed for the case of real classification, in particular the algebraic sign is reversed so that an increased number of points yields a worse success

rate. For the case of trial classification, the correlation's direction remains the same. Our approach to explain this observation, where we focus on the badly selected points, is as follows:
the training curves "randomly" differ at those points, which is indicated by the peaks in sosd, so that even these points identify and contribute to a correct classification of a training curve. The samples from device B probably differ as well at these points, but the "randomness" in the difference is another, so that the points do not identify a sample. One might name the badly selected points *false positives*. They are indeed a criterion to distinguish the training samples, but they are not a *characteristic* criterion, that can be applied to distinguish samples $\notin$ the training set.
For a reduced number of points P = 3 one can observe that the success rates are still similar to the corresponding rates in the tables above.

The general tendencies of all observations described above are illustrated in Figure 29.

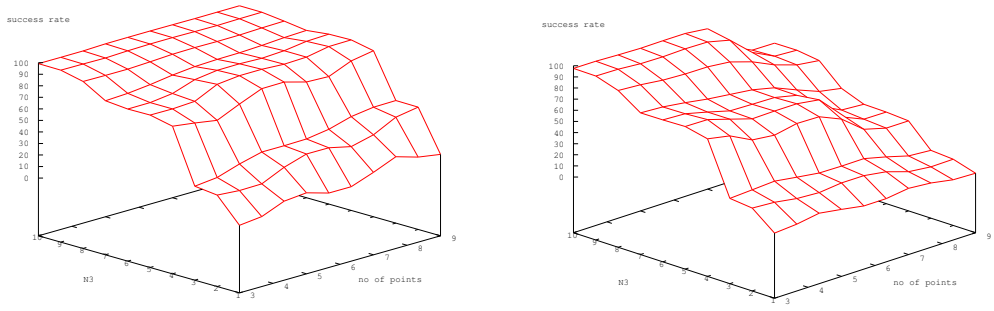


Figure 29: Success rates as functions of N_3 and P for $N_1 = 231448$ and 25000

Table 6 presents the results for $N_1 = 231448$ and a *non-bounded* point selection algorithm (see Observation 1). The point selection algorithm selected four instants which cover the initial Round Key addition. From comparison of the success rates to those in Table 3 one can clearly observe the negative impact of the false positives in case of real classification. Comparing the success rates of real and trial classification leads to the same conclusion: the

$N_1 = 231448$	$p = 9$	channel = power
poi[]	3828, 771, 3883, 437, 9496, 659, 9607, 326, 9551	

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	0,7	1,5	0,7	0,2		38,9	79,0	99,6	100,0
6	1,9	2,6	2,4	0,3		26,1	63,2	97,9	99,9
3	8,8	25,6	81,7	99,2		6,8	25,6	80,7	99,2

Table 6: Success rates (SR) for $N_1 = 231448$, channel = power, and a non-bounded point selection

selected points covering the initial Round Key addition contribute well in the trial classification, i.e., one can observe the usual logarithmic dependency of the success rates, while the algebraic sign of the dependency is reverted in the case of real classification. Less points, which means as well less false positives, lead to better success rates. Furthermore, the assumption is justified by comparing the success rates of trial classification in Tables 6 and 3.

5.2 Stochastic Model

As stated in Section 3.1 the AES algorithm operates on an array of 16 bytes, the State. After the initial RoundKey Addition, each byte of the state represents $x \oplus k$ for the corresponding bytes of the key and the plaintext. In the first normal round, the SubBytes transformation and the ShiftRow transformation both operate on these bytes separately. Therefore, it is sufficient for any attack that targets on one of these three transformations, to consider $h_t(state)$ with $state \in \{0,1\}^8$ as the real leakage function (cf. invariance property 'Equal Images under Different Subkeys' on page 31. This step reduces the effort to finding 2^8 mappings to $\mathbb{R}$ in the case of AES and should be applicable for many block ciphers.

5.2.1 Preliminary notes

Before an attack can be carried out, an adversary has to come to a couple of decisions in which he has a higher degree of freedom, than for the Tem-

plate Attack. More precisely, one has to select – considering all consequences stated in Section 3.4.2 (trade off problem) – a vector subspace $\mathcal{F}_u$ and the u functions $g_{jt} : \{0,1\}^p \times \{0,1\}^s \to \mathbb{R}$. As mentioned before, [11] does not give concrete advice or assistance for the procedure, but several different choices, based on AES, are presented and their results compared. We start our investigation with the most promising choice that is presented so that the setting for our attack is as follows:
We decide to attack the 128-bit key in steps of one byte thus for any given instant t the samples are modeled as $I_t(x,k) = h_t(x,k) + R_t$ with x and $k \in \{0,1\}^8$ being the appropriate sub-plaintext and -key. We exploit the EIS property of the deterministic sample portion, which stands for a virtually lossless reduction of the vector space in which $h_t^*(x,k)$ has to be approximated. With $\phi(x,k) = x \oplus k$ the effort is reduced from 2^{16} to 2^8 and samples are regarded as $I_t(x,k) = h_t^*(\phi(x,k)) + R_t$. The strategy to attack the AES State s after the initial RoundKey addition equals the strategy we applied in Section 5.1. The following choice, which is a further reduction of the vector space, is the most difficult one and this is where we heavily rely on [11]. We choose the nine-dimensional bit-wise coefficient model that is referred to as vector subspace $\mathcal{F}_9$. This implicates a further reduction from 2^8 to 9 dimensions. For the usual reason that (non-linear) diffusion of key candidates is highly desired the functions $g_j (j = 1, \ldots, 8)$ aim at the S-box output, i.e. $g_j(\phi(x,k)) \in \{0,1\}$ is the j-th bit of S-box$(\phi(x,k))$. $g_0(\cdot)$ always returns 1. Note that we use the following assignment to address bits:

MSB = 8	7	6	5	4	3	2	1 = LSB

Altogether, the deterministic sample portion is approximated by

$$h_t^*(x,k) = \sum_{j=0}^{8} b_{jt} \cdot g_j(x \oplus k) = b_{0t} + \sum_{j=1}^{8} b_{jt} \cdot g_j(x \oplus k). \tag{19}$$

The coefficients b_{jt} with $j \neq 0$ estimate the bit-wise data dependent sample portions and b_{0t} is an estimator for a non data dependent part.

5.2.2 Implementation

We implemented the Stochastic Model closely to the description in 3.4. In the following we describe our implementation step-by-step and provide data examples to illustrate the procedure. Furthermore, we introduce some abbreviations that will be used for clarity and when the results are presented. As before in Section 3, we would like to mention that parts of the description of our implementation of the Stochastic Model resemble the corresponding paragraphs in Section 5.1. Since both attacks are Two-Step SCAs that use device characterisations, there obviously are similarities. Nevertheless, for completeness and the sake of readability we will give an almost entire description of our implementation and only point to Section 5.1 in case of continuous consistency.

Profiling Step The Stochastic Model aims at generating an approximation of the deterministic sample portion in a chosen vector subspace and a characterisation of the noise. We approximated the deterministic sample portion in $\mathcal{F}_9$ (see 5.2.1) so that, for the deterministic part, *this* byte-wise attack aims at estimating the data dependent sample portion of each bit in S-box$(x \oplus k)$ and a non data dependent contribution. These will be referred to as *sub-signals.*
Our implementation of the Stochastic Model attacks one byte of the AES state array s resp. one sub-key at a time. It can attack any byte in s but for the sake of clarity and comparability to the Template Attack we focus on $s_{0,0}$ respectively the first keybyte.
The Stochastic Model uses one half of the available samples for the approximation of the deterministic sample portion and the other half for the characterisation of the noise. These amounts of samples will be referred to as N_1 and N_2.

The first step aims at determining the $N_1 \times 9$ design matrix A that contains the composition of sub-signals for all N_1 samples. Since k is fixed, A is implicitly given by the distribution of the plaintexts x_i. Figure 30 shows the

layout of A for this concrete scenario. Recall that $g_j(x \oplus k) \in 0, 1$ is the j-th bit of S-box$(x \oplus k)$.

$$A = \begin{pmatrix} 1 & g_1(x_1 \oplus k) & \cdots & g_8(x_1 \oplus k) \\ 1 & g_1(x_2 \oplus k) & \cdots & g_8(x_2 \oplus k) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & g_1(x_{n_1} \oplus k) & \cdots & g_8(x_{n_1} \oplus k) \end{pmatrix}$$

Figure 30: Design Matrix A for the bit-wise coefficient model ($\mathcal{F}_9$) exploiting EIS property

To determine A, we implemented a function `create_matrix_a()` that scans through all N_1 plaintexts and determines the 9 necessary bits for each plaintext x_i as follows. The S-box is implemented as a table lookup.

```
for 0 ≤ i < N1
begin
   A[i][0] ← 1
   for 0 ≤ bitpos < 8
   begin
      a[i][bitpos + 1] ← ( S-box(xi ⊕ k) >> bitpos ) & 1
   end
end
```

The second step computes the $9 \times N_1$ system matrix $S = (A^T A)^{-1} A^T$ which is only possible, if the 9×9 matrix $A^T A$ is regular, thus invertible. During our experimental analysis this was always the case, as long as we chose $N_1 \geq 2000$.

Step 3 aims at determining the b-vectors which will be the essential core of the estimator h_t^*. Recall that a b-vector contains the contribution of each of the 9 sub-signals to the approximated deterministic sample portion for an instant t. Whether a sub-signal actually contributes to the approximator $h_t^*(x, k)$, depends on the corresponding bit in S-box$(x \oplus k)$ being set. Our samples contain $m = 20000$ points, thus $t = 0, 1, \ldots, m - 1$. The b-vectors

are computed separately for each instant t by evaluating $\mathbf{b}_t = S \cdot \mathbf{i}_t$ with $\mathbf{i}_t$ being a column vector that holds the measured value for instant t of all N_1 samples. That is

$$\begin{pmatrix} b_{0,t} \\ b_{1,t} \\ b_{2,t} \\ \vdots \\ b_{8,t} \end{pmatrix} = \begin{pmatrix} S_{0,0} & S_{0,1} & \dots & S_{0,N_1-1} \\ S_{1,0} & S_{1,1} & \dots & S_{1,N_1-1} \\ S_{2,0} & S_{2,1} & \dots & S_{2,N_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ S_{8,0} & S_{8,1} & \dots & S_{8,N_1-1} \end{pmatrix} \begin{pmatrix} \text{sample}_0(t) \\ \text{sample}_1(t) \\ \text{sample}_2(t) \\ \vdots \\ \text{sample}_{N_1-1}(t) \end{pmatrix}.$$

For instant t, $b_{0,t}$ contains the non data dependent sub-signal's contribution and the $b_{j,t}$ with $j \neq 0$ contain the contribution of sub-signal j.

A straight forward implementation of this step is eminently costly, particularly for a large number of samples N_1, because for each computation of a b-vector *all* samples have to be read from hard disk [20]. Therefore we implemented the function `compute_b_vectors()` to embark the following strategy: load as many instants t from all N_1 samples as free memory is available, compute the b-vectors for these instants, then again, read as many of the remaining instants as possible etc.

Step 4 deals with the generation of the approximated deterministic sample portion.
For the remainder of this section, we "rotate" the representation of the b-vectors, so that a b-vector now has dimension $m = 20000$ and there are 9 of them. The benefit of this change of notation is that it eases to address the course of a sub-signal in the overall time frame. Instead of $(b_{0,0}, b_{0,1}, b_{0,2}, \dots, b_{0,m-1})$ we can simply use b_0. To address the contribution of sub-signal 0 at a specific instant, we write $b_0(t)$ as if it was a function.

[20] In the concrete scenario where each sample covers 20000 instants and e.g. $N_1 = 1000$ samples, the effort adds up to 20 million sample read operations. If sufficient RAM is available, all samples can be loaded into it prior to the computations in order to reduce this effort. For our analysis with N_1 increasing up to 115724 samples with a filesize of 20KB, neither the first approach (2.3 billion read operations) nor the second (2.3 GB free RAM) were feasible.

Figure 31 exemplarily shows the estimated power consumption of the non data dependent sub-signal (b_0) and of the sub-signal that contributes, if the least significant bit of S-box($x \oplus k$) is set (b_1) for the overall time frame. Both were computed from $N_1 = 115724$ power samples.

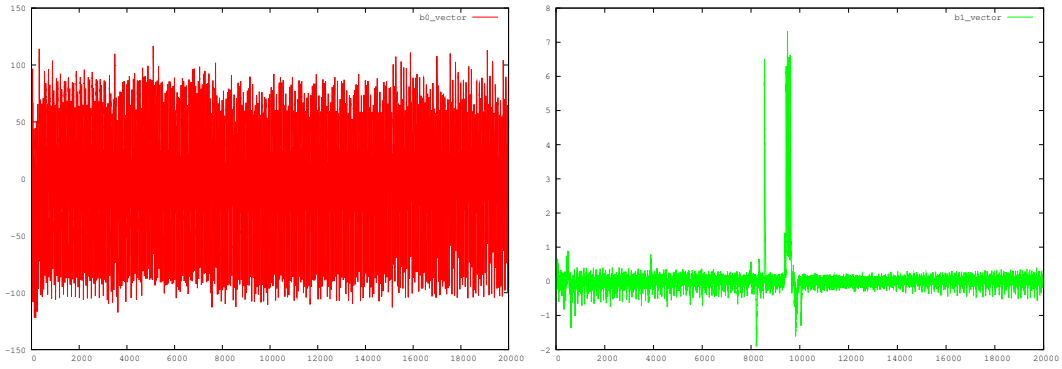


Figure 31: Non data dependent sub-signal b_0 and sub-signal b_1 that contributes for LSB(S-box$(x \oplus k)) = 1$ as a function of t

In opposition to the previous presentation $h_t^*(x, k)$, e.g. in (19) and with respect to the "rotated" representation, we implemented one discrete function

$$h^*(t, x, k) = b_0(t) + \sum_{j=1}^{8} b_j(t) \cdot g_j(x \oplus k). \tag{20}$$

which seemed to be more convenient. Function `double h(t, x, k)` assembles the approximation in exactly this way:

---

temp $\leftarrow b_0(t)$
for $0 \leq$ i < 8
begin
  temp $\leftarrow$ temp + $b_{i+1}(t) \cdot$ ((S-box$(x \oplus k)$ >> i) `&` 1)
end
output $\leftarrow$ temp

Step 5 computes the basis for the choice of interesting points in time. Instead of computing the Euclidean vector norm $||b_{0,t}, \ldots, b_{8,t}|| = \sqrt{\sum_{i=0}^{8} b_{i,t}^2}$

for all instants t as proposed in [11], we computed another measure for the sake of comparability. The sum of squared pairwise differences (**sosd**) of $h^*(t, x, k)$ for all possible values of x which yields the same result as the Euclidean vector norm, apart from a constant factor, is computed by function `compute_differences()`:

```
for 0 ≤ t < data_length (each point in time)
begin
  for 0 ≤ i < 256
  begin
    h1 ← h(t, i, k)
    for i+1 ≤ j < 256
    begin
      h2 ← h(t, j, k)
      sosd[t] = sosd[t] + (h1 − h2)^2
    end
  end
end
```

Figure 32 shows the resulting curve sosd and an example of the Euclidean vector norm as a function of time. Both curves were derived from $N_1 = 10000$ power samples. Note that the scale for the vertical axis is omitted to underline their analogousness in terms of conclusion.

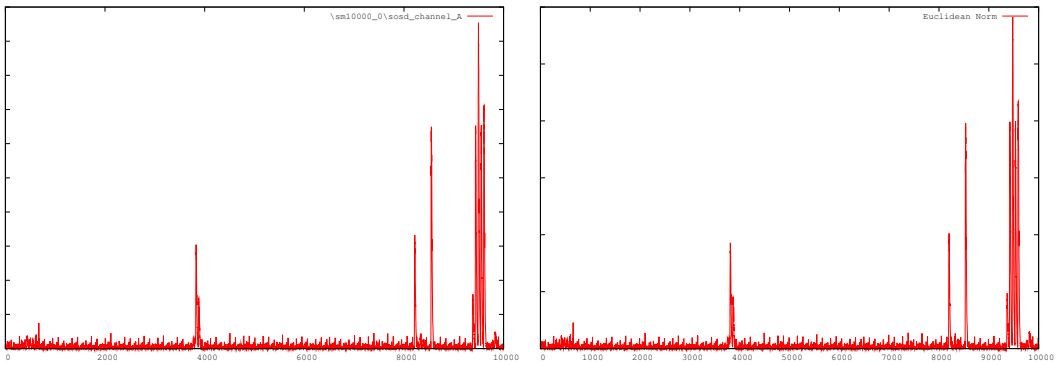


Figure 32: sosd and Euclidean vector norm derived from $N_1 = 10000$ power samples as a function of t

Step 6 comprises the actual choice of interesting points in time. [11] gives no more advise on how to choose the points than "*choose instants*" and a comparison of several selection strategies. An optimal strategy, if existent, seems to be unknown.
Since the basis for the selection of points (sosd) is the same as during our experiments with the Template Attack, we decided to re-use the selection algorithm as well for enhanced comparability. The selection algorithm is implemented in function `find_points_of_interest(curve, p)`. For details on the strategy we refer to Section 5.1.2 Step 4 whereat for the Stochastic Model it is not necessary to prevent the selection of points in the range 0 - 3300.

Steps 7 and 8 jointly perform the characterisation of the noise in the side channel R_t. The former is a preparatory step that supplies required data to the latter which generates the noise characterisation.
Step 7 extracts the noise within the N_2 samples that were not used so far. We implemented a function `compute_noise_vectors(N2)` that consecutively computes the difference of a sample and the appropriate approximator at the p selected instants for all N_2 samples. In the following pseudocode, **poi[]** is the set of selected instants.

```
for 0 ≤ i < N2
begin
  sample ← load_curve(i)
  x ← load_curve_plaintext(i)
  for 0 ≤ j < p
  begin
    noise_array[i][j] ← sample[poi[j]] - h(poi[j], x, k)
  end
end
```

For an illustrative example of a noise vector we refer to Figure 25 Section 5.1.2.

Step 8 generates the p × p covariance matrix that characterises the noise in the side channel. The theoretic approach of the covariance matrix generation is exactly the same as for the Template Attack, described in 5.1.2. The crucial difference is that during a Template Attack one generates a covariance matrix for each data dependency whereas in the Stochastic Model the noise is not assumed to depend on x, k and hence only one covariance matrix is generated. The implemented function `compute_covariance_matrix(N2)` equals the function `compute_covariance_matrix(*value, *no_of_files)` presented in Section 5.1.2 except for simply storing the matrix as covariance_matrix since there is only one.
The profiling step is completed with the generation of the matrix.

Classification Step The classification step aims at classifying N_3 side channel samples $S(x_j, k^\circ)$ from device B. This means to correctly deduce the secret key k° that was used for encryption by B while the samples were measured from the samples' properties. The approach is as follows: for each key hypothesis $k \in \{0, \ldots, 255\}$ the noise in the first sample $S(x_1, k^\circ)$ is extracted at the selected p instants using the appropriate approximator $h^*(t, x_1, k)$ as in step 7. The probability of observing such noise if indeed it derives from x_1, k is computed according to Equation (16). Then, the procedure is repeated for the remaining $N_3 - 1$ samples, thereby multiplying the probabilities for one key hypothesis as indicated in Equation (17). We used samples from our measurement set 2 to serve as S.

Step 9 randomly selects one of the 3000 samples from measurement set 2 as $S(x_j, k^\circ)$.

Step 10 extracts the noise within $S(x_j, k^\circ)$ for a given hypothesis k using the appropriate approximator $h^*(t, x_1, k)$. Function `compute_noise_vector_for_classification(*hypothesis,*S)` does this in exactly the same way as it was done in step 7. That is, it computes the difference of $S(x_j, k^\circ)$ and $h^*(t, x_1, k)$ at the selected instants. The resulting noise vector will be referred to as $\mathbf{z}$.

Step 11 computes the probability of observing such noise using the covariance matrix CM and stores the result for later comparison. We composed a set of functions to compute the probability as follows:

```
CM ← load_covariance_matrix()
CM ← compute_inverse(CM)
probability ← compute_probability(CM)
```

Since we are rather interested in a ranking of the key hypothesis than in the actual probabilities, compute_probability(CM) omits constant terms in Equation (16) which results in $\mathbf{z}^T CM^{-1}\mathbf{z}$ (see Equation (18)). This saves implementation effort and speeds up the procedure as in particular the determinant of the covariance matrix need not be determined.

Step 12 identifies the best hypothesis. The implemented function `sort_descending(probabilities,identifiers)` sorts the probabilities in descending order while simultaneously sorting the hypothesis' identifiers (values of k) in the same order so that the identifier of the best hypothesis is the first in the list.

The last four steps are invoked by a superior function `classify()` to compute the probabilities for the 256 key hypothesis using N_3 samples and choose the best candidate.

```
CM ← load_covariance_matrix()
CM ← compute_inverse(CM)
for 0 ≤ i < N_3 begin
   rand ← gen_rand()
   S ← load_curve(rand)
   x ← load_curve_plaintext(rand)
   for 0 ≤ hypothesis < 256
   begin
      compute_noise_vector_for_classification(*hypothesis, *S)
      if (i==0)
       probability[i][hypothesis] ← compute_probability(CM)
      else
       probability[i][hypothesis] ← probability[i-1][hypothesis]
       + compute_probability(CM)
   end
end
sort_descending(probability[N_3], identifiers)
selection ← identifier[0]
```

Note that probability[][][] is now a two-dimensional array. The one-dimensional list probability[N_3] based on which the selection is performed contains exactly the aggregated probabilities described in Equation (18). The motivation to implement a two-dimensional array is the same as for the Template Attack.

The rest of the procedure during the classification step is identical to our implementation of the Template Attack. Rather than recapitulating, we refer to Section 5.1.2 for details and only provide the essentials here.
To attain a certain precision, function `classify()` was further modified to carry out the classification step 1000 times while counting in the variable *correct*, how often the selected candidate was indeed k°. The success rate is then given by $\frac{correct}{1000}$. Furthermore, the final version of `classify()` was invoked several times per classification cycle. Before each invocation we varied the number of points P which would omit consideration of some of the least significant points selected in step 6 during the probability computations in step 11. The motivation for this decision was to observe and possibly quantify the impact of selected points on the success rate.

5.2.3 Results for various parameter settings

Carrying out our experimental analysis we noticed, that four parameters have major impact on the Stochastic Model's efficiency in terms of success rates. The first parameter is the vector subspace $\mathcal{F}_u$ in which the deterministic leakage portion is approximated. The comparison of several choices with respect to success rates in [11] indicated the bitwise coefficient model ($\mathcal{F}_9$) to be most efficient, hence the results presented in this section are all based on that choice. The other three parameters are the same as presented in Section 5.1.3, i.e.,

1. the number of curves available during the profiling step $\mathbf{N_1} + \mathbf{N_1}$
2. the number of interesting points that can be found in the profiling step **p** respectively that are used in the classification step **P**
3. the number of curves available during the classification step $\mathbf{N_3}$.

We tested all combinations $N_1+N_2\times$ P $\times N_3$ for $N_1+N_2 \in \{2k, 10k, 20k, 25k, 30k, 40k, 50k, 231448\}$, P $\in \{9, 6, 3\}$, and $N_3 \in \{1, 2, 5, 10\}$.
The procedure of the experiments was always the same as for the Template Attack. The results presented below reflect only those parameter values, that we find most significant. Appendix C provides all result tables for the Stochastic Model.
The presentation format of the results is the same as in section 5.1.3, i.e. we present several tables where each table represents a fixed value $N_1 + N_2$ and variations of P and N_3.

Table 7 presents the results for the best possible choice of $N_1 + N_2$. The selection algorithm found 9 points whose positions are provided in the second line of the table. We will refer to this distribution of points as the *optimal distribution* in the further analysis. From the two blocks containing the success rates for real and trial classification one can observe that classification of samples from device B is - in tendency - less successful than classification of samples from device A. This fact actually indicates a non optimal profiling

$N_1 + N_2 = 231448$	$p = 9$	channel = power

poi[]	9496, 9607, 9551, 8551, 9440, 8218, 3828, 9385, 3883

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	4,3	12,6	52,1	90,7		3,3	12,2	63,1	96,3
6	1,9	8,1	43,0	91,2		3,1	9,4	55,2	94,4
3	1,9	4,9	23,6	68,5		1,0	6,1	34,1	80,0

Table 7: Success rates (SR) for N_1 and $N_2 = 115724$ and channel = power

step since we assume balanced success rates for an optimal (lossless) profiling. The results provided further below will motivate, why we still refer to this value of $N_1 + N_2$ as best possible and regard the non-optimal profiling as a result of the approximation.
Both of the blocks clearly show, that the parameters P and N_3 have direct impact on the success rate. For each of the parameters, the correlation of its value and the success rate is best described as logarithmic, which is also true for simultaneous changes of both parameters' values. There are no more clearly visible tendencies in the correlations, in particular we cannot state on which parameter's influence is stronger.

Table 8 shows the results based on $N_1 + N_2 = 25000$. Although we reduce the number of training curves in the profiling step by a factor $\sim$10 one can, in the case of real classification, still observe success rates in the same order of magnitude. The success rates of trial classification do not seem to be affected at all, there is no visible tendency of change. The selection algorithm found seven points from the optimal distribution and two points that are slightly displaced but still in the correct processor cycle (9552 instead of 9551 and 9441 instead of 9440). The bad selection must be an implication of the deterministic part's worse approximation. At least we can notice, as Figure 33 shows, that it is not caused by a substantially increased noise floor. Since all computed success rates are affected by at least one badly selected point, a statement quantifying the effect of these points is not feasible.

$N_1 + N_2 = 25000$	$p = 9$	channel = power
poi[]	9496, 9607, 9552, 8551, 9441, 8218, 3828, 9385, 3883	

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	4,4	8,0	39,8	85,3		3,1	13,9	64,1	97,6
6	2,3	7,1	33,8	80,6		2,4	10,5	57,8	97,1
3	0,9	4,2	19,2	61,4		1,2	4,6	34,4	81,6

Table 8: Success rates (SR) for N_1 and $N_2 = 12500$ and channel = power

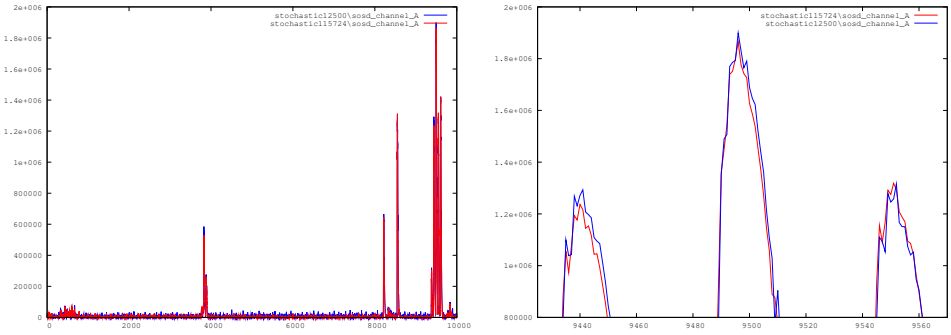


Figure 33: sosd for $N_1 + N_2 = 231448$ and 25000 as a function of time

Table 9 shows the results for $N_1 + N_2 = 10000$. The further reduction of $N_1 + N_2$ by a factor 2.5 clearly has an impact on the success rates in the case of real classification. As before, the success rates of trial classification seem not to be affected, there is no visual tendency of decline. The selection

$N_1 + N_2 = 10000$	$p = 9$	channel = power
poi[]	9496, 9605, 8551, 9441, 8218, 3828, 15845, 12677, 13344	

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	1,0	1,9	16,6	46,9		4,0	10,8	58,3	96,1
6	1,1	2,4	15,5	49,5		3,2	11,4	53,8	93,6
3	0,4	1,7	9,7	34,0		2,0	5,2	38,8	88,9

Table 9: Success rates (SR) for $N_1 + N_2 = 10000$ and channel = power

algorithm found only 4 points from the optimal distribution, two are slightly displaced but still in the correct processor cycle and three points are not

related to the optimal distribution at all. The bad selection is partly a result of "unlucky" circumstances, as can be seen by looking at Figure 34 and the distribution of poi[] in Tables 7 and 9. The point at 9496 is well selected,

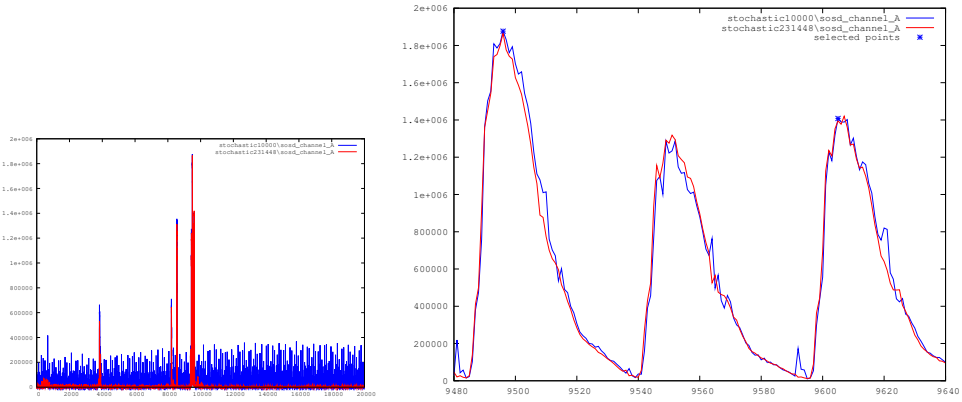


Figure 34: sosd for $N_1 + N_2 = 231448$ and 10000 as a function of time

the peaks of both sosd curves show a similar shape at that instant. Since the shape of the peak for the third processor cycle is slightly different than optimal, the point for this cycle is selected a little bit to the left (9605) from the optimal position (9607). This "unluckily" eliminates the chance to select a point during the middle cycle at all, because the algorithm disregards an entire cycle before and after every selected point. The three badly selected points past 10000 are a result of a perceivably increased noise floor. Their negative influence on the success rate for real classification can be observed by comparing the table rows for P = 9 and P = 6.

5.3 Comparison

In this section, we compare the efficiency of the profiling and the classification step of our implementations of the Template Attack and the Stochastic Model as they were introduced in Sections 5.1 and 5.2. The determination of all numbers was performed as objective as possible, that is in particular:

- identical mathematical operations, e.g. the computation of a covariance matrix, are performed by the same code to exclude differences in the computational approach

- both attacks use sosd as basis for the selection of interesting points; as sosd is computed from the averaged resp. approximated deterministic signal portions, it can be regarded as a measure for an attacks ability to deal with noise
- both attacks used the same point selection algorithm with constant parameters (δ = 54, p=9), we do not adapt any settings to specific situations

We focus on comparing the efficiency of the attacks against the three parameters, which showed to have strong influence on both of them, i.e.,:

1. What is the impact of N_1 resp. $N_1 + N_2$?
2. What is the impact of N_3?
3. What is the impact of P?

The attacks had to compete with each other in all possible parameter combinations of N_1 resp. $N_1 + N_2 \in \{10k, 20k, 25k, 30k, 40k, 50k, 231k\}$, P $\in \{9, 6, 3\}$ and $N_3 \in \{10, 5, 2, 1\}$. For the sake of clarity, we perform the comparison by means of graphs here, the interested reader can find all individual numbers in the result tables in Appendixes B and C.

Profiling Efficiency

For a start, we compare the efficiency of the attacks during the profiling step. As before, we define the set of points which was found for N_1 resp. $N_1 + N_2$ = 231448 as the *optimal distribution* with respect to each attack. For decreasing numbers N_1 resp. $N_1 + N_2$ we determine how good the selected set of points is by comparing it to the optimal distribution. If a selected point is in the optimal set, it has weight 1. If it is not in the optimal set but still in the correct processor clock cycle, it has weight 0.5. All other selected points have weight 0.

More formally: let poi_opt[] be the optimal distribution and poi[] be the set of p selected points $\mathcal{P}_i$ $(i = 1, \ldots, p)$. We say:

- $\mathcal{P}_i$ has weight $w_i = 1$, if $\mathcal{P}_i \in$ poi_opt[]
- $\mathcal{P}_i$ has weight $w_i = 0.5$, if $\mathcal{P}_i \pm x \in$ poi_opt[], with $x \leq \delta$
- $\mathcal{P}_i$ has weight $w_i = 0$, if $\mathcal{P}_i \pm x \notin$ poi_opt[], with $x \leq \delta$

where δ is the limit of tolerance, see Section 5.1.2, Step 4. We use the sum of these weights, i.e., $\sum_{i=1}^{p} w_i$, as a measure for profiling efficiency for a given number of training curves N_1 resp. $N_1 + N_2$.
The performance at point selection is an adequate measure for the overall profiling efficiency, because it is based on sosd, thus on the averaged resp. approximated data depended sample portions. On the other hand, the performance at noise characterisation depends straight on the quality of sosd, thus on the averages resp. approximator, and the selected points and is therefore "covered" by the above measure.

Figure 35 shows a plot of the sum of these weights as a function of the number of samples used in the profiling step (N_1 resp. $N_1 + N_2$). Note that

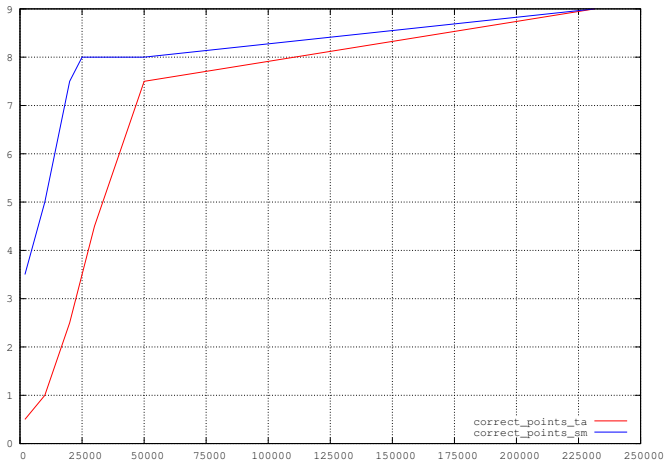


Figure 35: Well selected points in the profiling step

the Stochastic Model uses only 50% (N_1) of the available samples ($N_1 + N_2$) to approximate the deterministic leakage function on which the computation of sosd and hence the selection of points is based! The plot clearly indicates

the superiority of the Stochastic Model in terms of selecting the right characteristics and hence, in profiling efficiency. The difference probably originates from the way, in which each attack uses the samples to estimate resp. approximate the deterministic sample portion.

In an alternative approach for the Stochastic Model, one may use $N_1 + N_2$ samples to obtain the relevant points of the deterministic leakage function, before a re-run is done with the usual configuration.

Classification Efficiency

In the following, we compare the classification success rates of the attacks. We restrict our attention to real classification, thus classification of samples from device B, since it is the more interesting case and $N_3 \in \{1, 10\}$ for the sake of clarity. Again, all individual numbers are provided in Appendixes B and C.

First, we compare the success rates of both attacks for fixed parameters. The graphs show, how good each attack deals with a given situation. Thereafter, we compare the success rates for variations of N_1 resp. $N_1 + N_2$, $N_3 \in \{1, 10\}$, and, each time, the optimal choice of P. The graph shows the most, that each attack can make of a given number of curves during the profiling step when classifying one resp. ten samples from device B.

Figure 36 shows the success rates plotted as a function of N_1 resp $N_1 + N_2$ for fixed P = 9 and $N_3 \in \{1, 10\}$. As expected, one can observe, that both an increasing number of curves for the profiling step and an increasing number of curves for the classification step have a positive impact on both attacks' success rates. Additionally, the positive impacts intensify each other when effective simultaneously. As observed before, the increase of the success rates can be described as logarithmic. For both attacks it seems, there exists a saturation threshold with respect to N_1. Once the threshold is reached, additional curves in the profiling step only yield a small positive increase of the success rates.

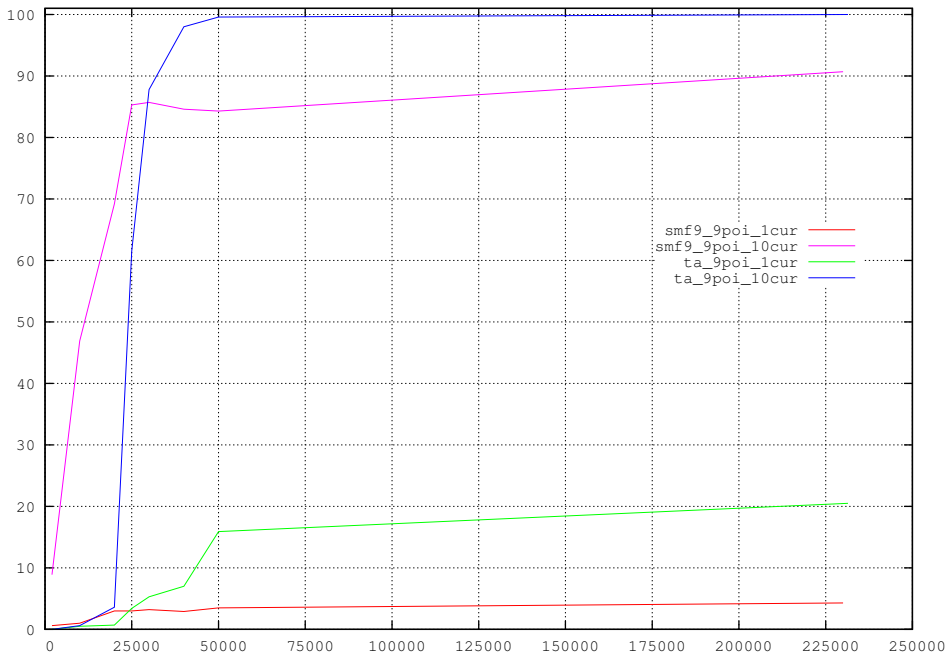


Figure 36: Plot of the Template Attack's and the Stochastic Model's success rates as a function of the number of curves during the profiling step. It is P=9 and $N_3 \in \{1, 10\}$.

Another observation is, that the Stochastic Model is more efficient than the Template Attack for small N_1, more precisely for $N_1 < 30000$ if $N_3 = 10$ and $N_1 = 25000$ if $N_3 = 1$. This supports, in accordance with our argumentation for the superiority of the Stochastic Model during the profiling step, the following assumption: for a given value of N_3, there exists a threshold value for N_1 so that for smaller N_1 the superiority of the Stochastic Model in the profiling step (caused by the approximation) outweighs its inferiority in the classification step. Or, with respect to the Template Attack: for N_1 larger than the threshold value, the effort of the more complex but as well more precise profiling step pays off and yields superior results in the classification step.

Figure 37 shows the success rates plotted as a function of N_1 resp $N_1 + N_2$ for fixed P = 6 and $N_3 \in \{1, 10\}$. For the Template Attack, one can observe that for $N_1 < 40000$ the choice P = 6, thus disregard of the three least sig-

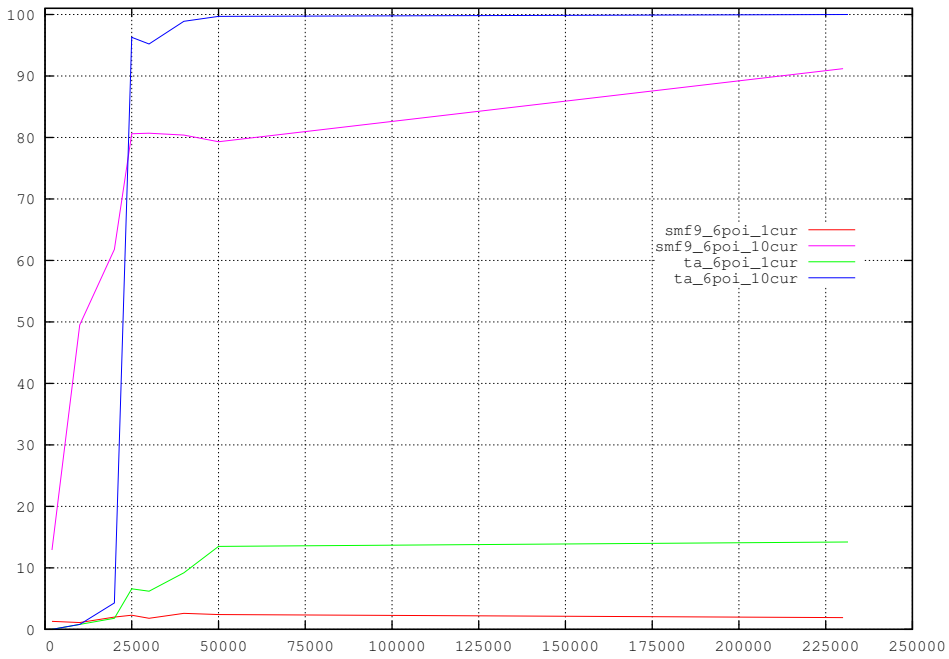


Figure 37: Plot of the Template Attack's and the Stochastic Model's success rates as a function of the number of curves during the profiling step. It is P=6 and $N_3 \in \{1, 10\}$.

nificant points, yields better success rates than P = 9. As this is true for all tested values of N_3 it indicates a general trend. For $N_1 = 40000$ there is no significant difference in the success rates, it can be seen as an inflexion point of the trend. For $N_1 > 40000$ the trend is inverted, i.e., P = 9 yields better results than P = 6, but becomes less obvious for increasing N_3 because the success rates get very close to the boundary of 100%.

The Stochastic Model shows a similar behavior, but the inflexion point is much lower. Especially for small N_3, the trends are easier to see in the result tables than in the graphs. For $N_1 > 10000$, the choice P = 9 yields better results than P = 6. For $N_1 < 10000$, P = 6 is the better choice.

These observations give further support to our assumption. For N_1 smaller than the inflexion point, the entropy[21] of the least significant selected points is so small, that to involve them in the (real) classification process worsens

[21] or significance, force of expression

the success rates. For N_1 larger than the inflexion point, their entropy is good enough to improve the success rates. In accordance with our argumentation for the superiority of the Stochastic Model in the profiling step, its inflexion point (~ 10000) is reached earlier than the Template Attack's (~ 40000).

Figure 38 shows the success rates plotted as a function of N_1 resp N_1+N_2 for fixed P $= 3$ and $N_3 \in \{1, 10\}$. For the Template Attack, the plot basically

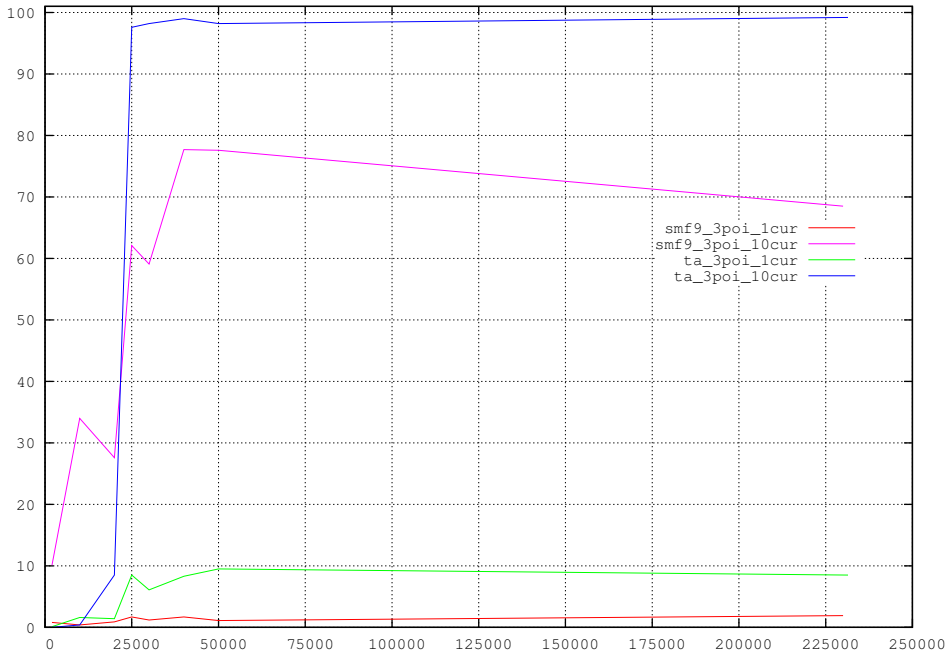


Figure 38: Plot of the Template Attack's and the Stochastic Model's success rates as a function of the number of curves during the profiling step. It is P$=3$ and $N_3 \in \{1, 10\}$.

shows a continuation of the observations made above. For $N_1 < 40000$ the choice P $= 3$ yields even better success rates than P $= 6$, while the opposite is noticeable for $N_1 > 40000$. This observation again supports the argumentation that for small N_1 the entropy in the least significant points is too low and excluding them from the (real) classification step yields better results. For the Stochastic Model, the choice P $= 3$ always[22] results in worse success

[22] within the boundaries of our experiments

rates than for P = 6 . We give a cautious explanatory approach: due to the approximation of the deterministic sample portion it might be true that the force of expression of each single point is bounded upwards. In this case, three points, even though their entropy meets this boundary, might simply be not enough to reliably distinguish key candidates.

Figure 39 shows the success rates plotted as functions of N_1 resp $N_1 + N_2$ for $N_3 \in \{1, 10\}$. For each attack and each instant, P was selected to maximise the success rates. One can observe, that each pair of plots intersects at

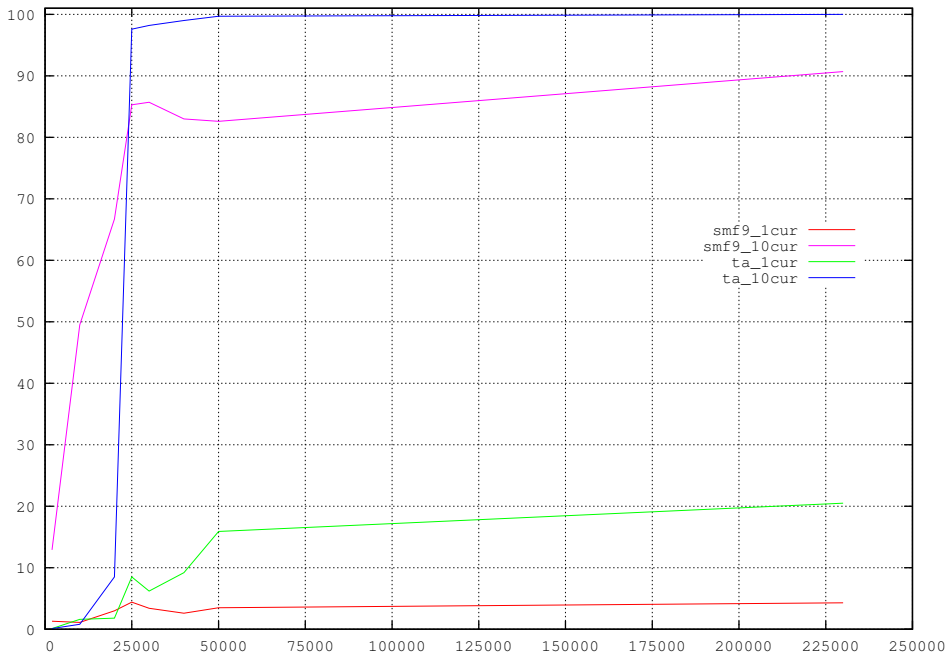


Figure 39: Plot of the Template Attack's and the Stochastic Model's success rates as a function of the number of curves in the profiling step. It is $N_3 \in \{1, 10\}$ and P chosen optimally.

least once. Hence, a general statement on which attack yields better success rates is not feasible as this depends on the number of curves that are available in the profiling step.
If a large number of samples is available (e.g. > 20000), the Template Attack yields higher success rates due to its higher precision. If only a small number

of samples is available (e.g. < 20000), the Stochastic Model is the better choice, because of its superior ability to filter noise.

We want to point out explicitly that these results refer to our

- implementation of the AES encryption algorithm
- choice of acquisition equipment
- measurement parameters
- implementations of the attacks

and should not be generalized. Due to the problematic of side channel information quality, the general observations only remain valid for slightly modified procedures which will, very likely, already yield different absolute numbers. Nevertheless, for a methodical comparison a case study has to be based on identical starting conditions. This case study aimed at giving a systematic and fair comparison and at the determination of critical parameters for further improvements.

5.4 Fixed key vs. variable key

As mentioned earlier in Section 4.4, we carried out a third measurement series. Each of the 256000 samples represents AES encryption of a randomly chosen plaintext with a randomly chosen key. Figure 40 which illustrates the sosd curve computed from all samples of this third set confirms our observation 1 from Section 5.1, Step 4. The sosd curve computed from samples that represent random plaintexts and random keys shows far smaller peaks during the initial Round Key addition than the sosd curve computed from samples that represent a fixed key.

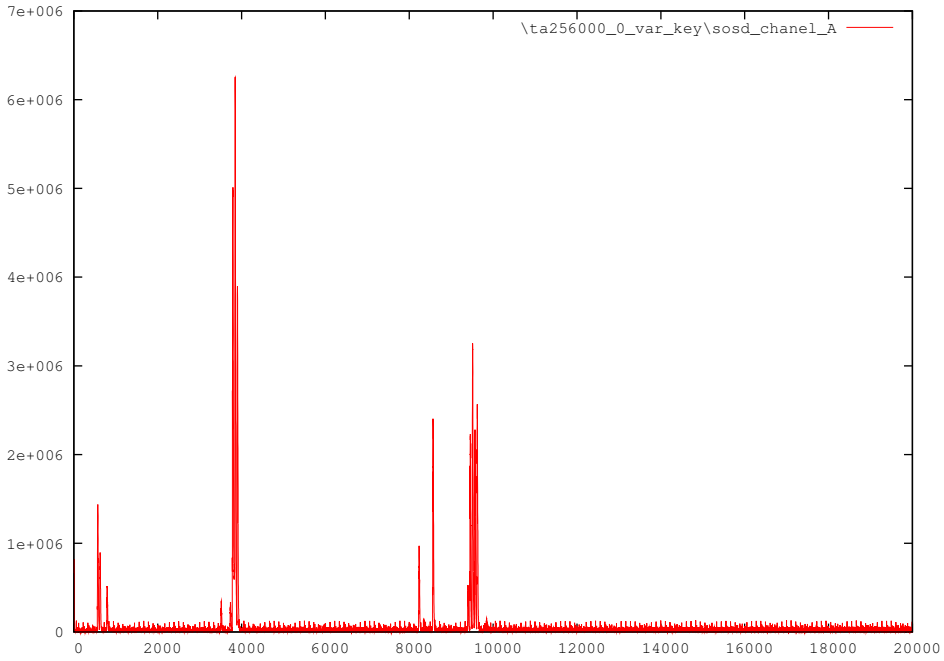


Figure 40: sosd curve computed from $N_1 = 256000$ samples representing random plaintexts and keys

6 Analysis of Results, Overall observations

We clearly showed, that the two statements

> The Template Attack extracts all possible information available in each sample and is hence the strongest form of side channel attack possible in an information theoretic sense given the few samples that are available. [10]

> Though our efficiency at key extraction is limited by template attacks profiling is much more efficient which is highly relevant if the designer of a cryptosystem is bounded by the number of measurements in the profiling step. [11]

are not true in a universal way, but that one has to consider the circumstances.
Due to the approximation of the deterministic sample portion, the Stochastic Model is more efficient in the profiling step, which is highly relevant if the number of training samples is limited. This leads to superior success rates in the classification step. On the other hand, the approximation sets an upper boundary for the Stochastic Model's entropy, which limits its efficiency in the classification step, if "enough" samples are available.

Due to its higher precision, the Template Attack is less efficient in the profiling step, if only a small number of training samples is available. This leads to inferior success rates in the classification step. On the other hand, the Template Attacks's greater entropy pays off and yields superior success rates, if "enough" samples are available.

6.1 Weaknesses and strengths

Template Attack The strength of the Template Attack is, that it in fact extracts far more information from the samples than the Stochastic Model. Given enough samples in the profiling step, it is clearly superior to the Stochastic model in the classification step, due to the precise estimation of the average signal and the 256 covariance matrices. On the other hand, that is

its weakness as well. Because it "learns" so detailed, it requires much more samples in the profiling step than the Stochastic Model, to reach the same level of precision (see Figure 35). Due to the fine partitioning of the samples, the Template Attack needs many samples to reduce the noise in the side channel.

Stochastic Model The Stochastic Model's strength is the ability to "learn" quickly from a small number of samples. One weakness lies in the reduced precision due to the approximation in a vector subspace. We recall from Section 3.4.2:
Apparently, the number of required samples in the profiling step increases with the number of dimensions u, if the same level of precision is aspired for the β_{jt}. One might see this as a trade off problem for a **fixed** number of samples in the profiling step: a small number of dimensions u reduces the searchable space, which might exclude good candidates $h' \in \mathcal{F}$ but gives better estimators for the best h^* still included in $\mathcal{F}_{u;t}$. A large number of dimensions u will more likely include a very good candidate h^* but its estimators will be less precise.
So far, no better vector subspace (which is significantly smaller than 2^8) than $\mathcal{F}_9$, the bitwise coefficient model, has been discovered. A second weakness is the usage of only a single covariance matrix.

6.2 Average vs. Approximator

In this section, we determine, how good the Stochastic Model can approximate the deterministic sample portion, which is ideally estimated by the Template Attack. The approach is as follows: we assume that the average signals computed by the Template Attack are the optimal estimation and use them as reference value. For each key dependency, we generate the Stochastic Model's approximated deterministic sample portion, compute the difference to the reference value, and average these differences over all key dependencies. The resulting average difference, which we present below for selected

values of N_1, show how good the Stochastic Model approximates in average.

Figure 41 shows plots of the average difference between the approximated (Stochastic Model) and the averaged (Template Attack) deterministic signal, that were computed from $N_1 = 10k, 50k, 231448$ samples. Considering the

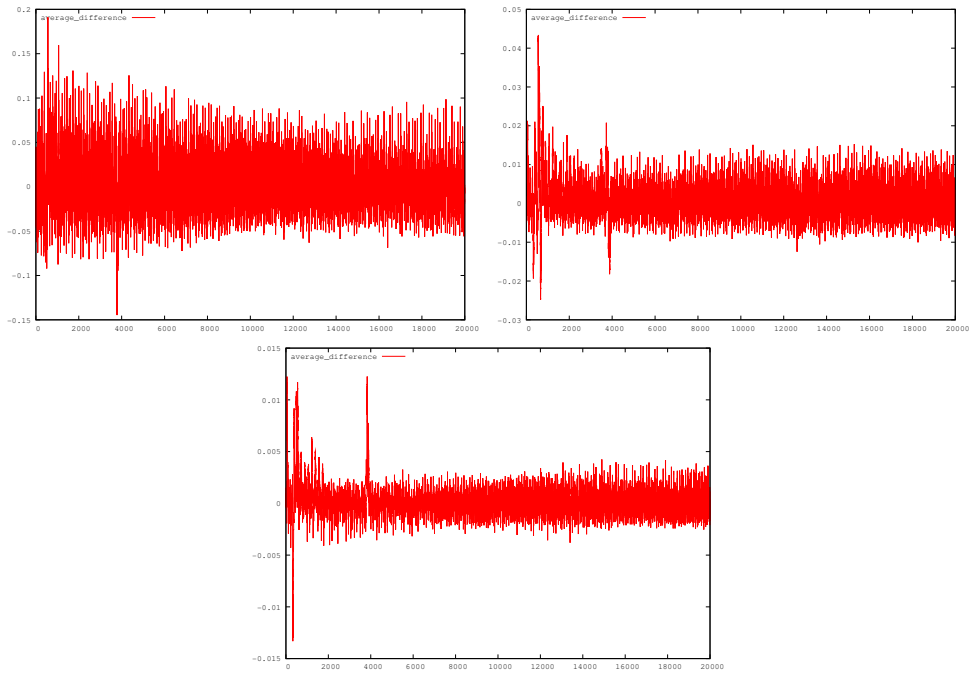


Figure 41: Average differences between the approximated deterministic sample portions of the Stochastic Model for N_1 = 10k, 50k, 115k *and* 231448 and the reference values

varying scale of the vertical axis, one can observe how the approximation decreasingly differs from the reference value for an increasing numbers of samples N_1. We assume, that this tendency is caused by the average signal, that becomes precise more slowly than the approximator, but finally is more precise for a large number N_1. The peaks especially visible in the plot on the lower right hand side ($N_1 = 231448$) are addressed in Section 6.4.2.

6.3 One vs. 256 Covariance Matrices

In this section, we exemplarily determine the influence of the number of covariance matrices, that are used, on the efficiency in the classification step. The approach is as follows: First, we generate the approximated deterministic sample portion of the Stochastic Model for $N_1 = 20000$. Then, we feed them into a Template Attack ($N_1 = 20000$), where we use them instead of the average signals. Table 10 opposes the derived success rates (real classification) of an attack with the original Stochastic Model ($N_1 + N_2 = 20000$) and of the modified attack based on approximated signals derived from $N_1 = 20000$ samples, which are re-used during the noise characterisation. Both attacks selected an identical set of points.

N_1 resp. $N_1 + N_2 = 20000$	$p = 9$	channel = power

	One covariance Matrix					256 covariance matrices			
P \ N_3	1	2	5	10		1	2	5	10
9	3,0	6,2	27,2	66,7		4,2	10,5	46,4	90,1
6	2,0	4,5	21,5	61,8		1,7	4,4	29,6	74,4
3	0,9	1,9	8,8	27,6		1,7	5,4	27,7	75,2

Table 10: Success rates of Stochastic Model attacks with one and 256 covariance matrices

Especially for $P = 3$, thus consideration of only the most significant points, one can observe a clear superiority of the attack that uses 256 covariance matrices.

6.4 Improvements (2)

In this section we present improvements to the Template Attack and the Stochastic Model which we developed after longterm examination of each attack's characteristics.

6.4.1 Template Attack with T-Test

The Template Attack's weakness is its poor ability to reduce the noise in the side channel samples if the adversary is bounded in the number of samples in

the profiling step. For small N_1, the remaining noise distorts the sosd curve, which we used as the basis for the selection of interesting points so far (the effect is even worse for the originally suggested sod curve).
Recall that sosd represents the sum of squared pairwise differences of the average signals $\sum_{i,j=0}^{K}(m_i - m_j)^2 \quad$ with $j \geq i$. Although sosd is clearly superior to sod (see Figures 20 and 21) its significance is limited if the underlying average signals disperse because of remaining noise. Figure 42 illustrates the problem.

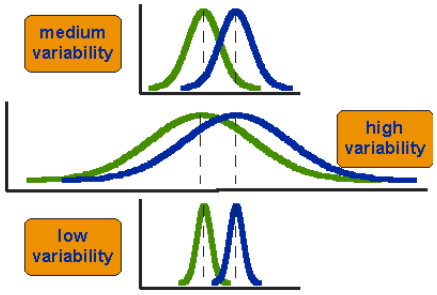


Figure 42: Distributions with equal mean and different dispersion [33]

Obviously, the (equal) mean value is not a sufficient criterion to distinguish the distributions in presence of varying dispersion (noise).

The T-Test is an advanced statistical tool to meet the challenge of distinguishing noisy signals. When computing the significant difference of two sets, it does not only consider the distance of their means but as well their variability in relation to the number of samples.

$$t = \frac{\overline{x} - \overline{y}}{\sqrt{\frac{\sigma_X^2}{n_x} + \frac{\sigma_Y^2}{n_y}}}$$

In other words: the higher the dispersion is in two sets, the less the distance between their means is weighted. Figure 43 depicts the T-Test's approach.

We added a step 2b to our implementation of the Template Attack that computes the variance for each operation from all corresponding samples and their previously computed average. Its core is function `compute_`

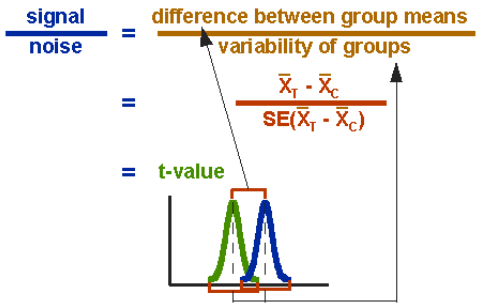


Figure 43: T-Test considers means' distance and variability [33]

`variances(*byte,*value,*no_of_files)` which is invoked for each operation once:

```
average_curve ← load_average_curve(value)
for 0 ≤ curve < no_of_files
begin
   for 0 ≤ instant < 20000
   begin
      variance[instant] ← variance[instant]
       + (curve[instant] - average_curve[instant])²
   end
end
for 0 ≤ instant <no_of_files
begin
   variance[instant] ← variance[instant] / (no_of_files)
end
```

Furthermore, we modified step 3 of our implementation to compute the sum of squared pairwise t-differences (sost) for all instants instead of sosd.

Figure 44 illustrates the striking difference between sosd and sost for $N_1 = 50000$ and 10000 samples. The scale of the vertical axis is not the same for all plots, but as one is not interested in comparing the absolute height of the peaks, this can be disregarded. What is important, and this is why we chose to present the plots in the way we do, is the relative distance between the peaks and the noise floor in each curve. While the reduction of N_1 by a factor 5 leads to a very distorted sosd signal, the significance of sost in terms

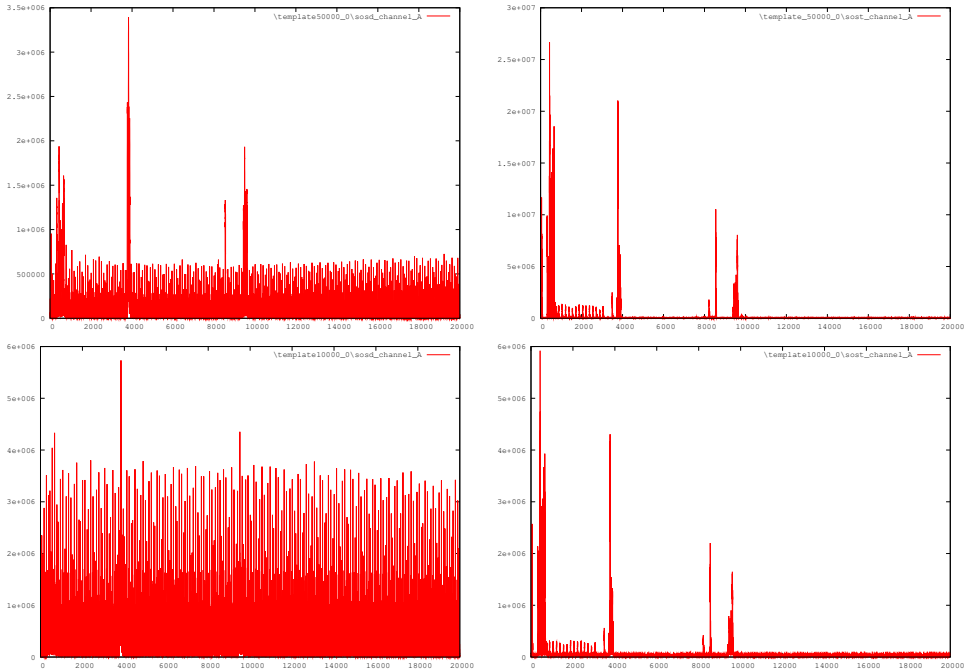


Figure 44: sosd (left) and sost (right) as functions of time, $N_1 = 50000$ (top) and 10000 (bottom)

of where to find interesting points does not change. Apart from the different scale, the peaks have a virtually identical shape. One can observe as well that sost generates a significantly better highest peak to noise floor distance than sosd by just looking at the upper two plots while paying attention to the different scales.

6.4.2 High-Order Stochastic Model with F_{17}

One weakness of the Stochastic Model with $\mathcal{F}_9$, the bitwise coefficient model, is the reduced precision due to the approximation of the deterministic sample portion. Even if enough samples are provided to estimate the bitwise contribution as good as possible, the overall efficiency is bounded by the approximation itself. In other words: it exists a threshold from which on additional samples in the profiling step do not increase the entropy of the approximator anymore, one might call this saturation. But because even a perfectly estimated approximator is only an approximator, precision is lim-

ited.
The obvious solution to this problem is to increase the number of dimensions of the vector subspace in order to generate a more precise approximator for the cost of needing more samples in the profiling step (trade off problem). But as the authors of [11] already analysed several high-dimensional vector subspaces and concluded that $\mathcal{F}_9$ seems to be most efficient, we decide to follow a different attempt.

Our approach arises from comparing the sosd curves of the Stochastic Model and the Template Attack, see Figure 45 (left). As one can see, the peaks on the right hand side of the plot are identical for both attacks, whereas the peaks at prior instants are not alike. Due to the fact that the underlying

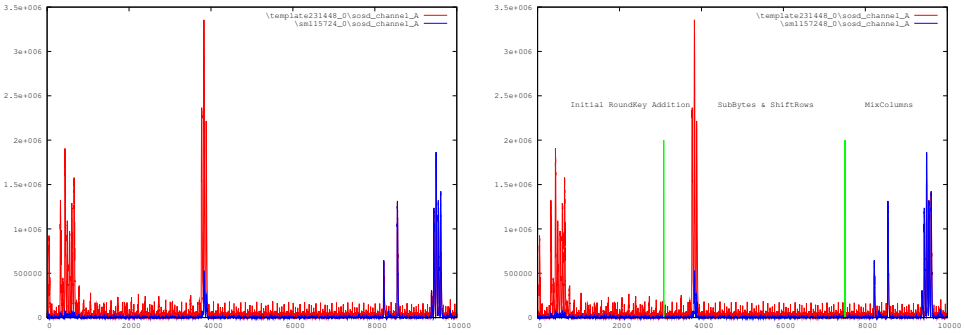


Figure 45: sosd curve of the Template Attack (red) and the Stochastic Model (blue) as functions of time

samples represent only one fixed key, the Template Attack's sosd curve shows peaks for x, $x \oplus k$, and $\text{Sbox}(x \oplus k)$. Knowing that our AES implementation unites the SubBytes and the ShiftRows transformation, we concluded an assignment of AES transformations to sosd peaks, which is depicted in Figure 45 (right). The peaks indicate, from left to right, the initial Round Key addition, the combined SubBytes and ShiftRows transformation, and the MixColumns transformation.
Since the Stochastic Model only approximates the deterministic sample portion at $\text{Sbox}(x \oplus k)$, it can not track bits "through" the Sbox and hence we conclude, that its most significant peaks (at the right hand side of the plot)

indicate the MixColumns transformation, which supports the assumed assignment.
Our approach aims at the fact that the Stochastic Model "overlooks" instants covering the Sbox lookup which yield the strongest peaks in the sosd curve of the Template Attack. We increase the number of dimensions of the vector subspace, but rather than increasing the level of detail at one instant of the AES encryption, we add consideration of a second instant. We (re-)define the selection functions g_j of the 17-dimensional vector subspace $\mathcal{F}_{17}$ as follows:

- as for $\mathcal{F}_9$, $g_0(\cdot)$ always returns 1
- as for $\mathcal{F}_9$, $g_j(j = 1, \ldots, 8)$ aim at the S-box output, i.e. $g_j(\phi(x, k)) \in \{0, 1\}$ is the j-th bit of S-box$(\phi(x, k))$
- $g_j(j = 9, \ldots, 16)$ aim at the S-box **input**, i.e. $g_j(\phi(x, k)) \in \{0, 1\}$ is the j-th bit of $\phi(x, k)$

In formal notation that is:

$$g_j(\phi(x,k)) = \left\{ \begin{array}{ll} 1 & \text{if } j = 0 \\ j\text{-th bit of S-box}(\phi(x,k)) & \text{if } 1 \leq j \leq 8 \\ j\text{-th bit of } \phi(x,k) & \text{if } 9 \leq j \leq 16 \end{array} \right\} \quad (21)$$

Although the general idea of the adaption of our implementation of the Stochastic Model is simple, steps 1 - 4 of the implementation are involved. We will restrict our attention to the essential modifications of steps 1 and 4 while steps 2 and 3 basically have to be upgraded to cover 17 instead of 9 dimensions.

Step 1 is modified to determine the $(N_1 \times 17)$ - matrix A whose layout is illustrated in Figure 46.

$$A = \begin{pmatrix} 1 & g_1(x_1 \oplus k) & \cdots & g_8(x_1 \oplus k) & g_9(x_1 \oplus k) & \dots & g_{16}(x_1 \oplus k) \\ 1 & g_1(x_2 \oplus k) & \cdots & g_8(x_2 \oplus k) & g_9(x_2 \oplus k) & \dots & g_{16}(x_2 \oplus k) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & g_1(x_{N_1} \oplus k) & \cdots & g_8(x_{N_1} \oplus k) & g_9(x_{N_1} \oplus k) & \dots & g_{16}(x_{N_1} \oplus k) \end{pmatrix}$$

Figure 46: Design Matrix A for $\mathcal{F}_{17}$ exploiting EIS property

Step 4, keeping our "rotated" representation of the b-vectors, is modified to approximate the deterministic sample portion by:

$$h^*(t,x,k) = b_0(t) + \overbrace{\sum_{j=1}^{8} b_j(t) \cdot g_j(x \oplus k)}^{\text{selection at Sbox output}} + \overbrace{\sum_{j=9}^{16} b_j(t) \cdot g_j(x \oplus k)}^{\text{selection at Sbox input}} . \tag{22}$$

Function `double h(t, x, k)` assembles the approximation in exactly this way:

temp $\leftarrow b_0(t)$
for $0 \leq$ i $<$ 8
begin
  temp $\leftarrow$ temp + $b_{i+1}(t) \cdot$ ((S-box$(x \oplus k)$ >> i) & 1)
end
for $0 \leq$ i $<$ 8
begin
  temp $\leftarrow$ temp + $b_{i+9}(t) \cdot$ ((($x \oplus k$) >> i) & 1)
end
output $\leftarrow$ temp

Figure 47 illustrates the considerable difference in the sosd curves computed by the Stochastic Model with $\mathcal{F}_9$ and with $\mathcal{F}_{17}$. Each time, the sosd curve of the Template Attack is provided as means of comparison. It is N_1 resp. $N_1 + N_2 = 231448$. The sosd curve of the High-Order Stochastic Model with $\mathcal{F}_{17}$ comprises additional clear peaks. The peaks on the left hand side of the plot indicate the initial Round Key addition and have a shape similar to the peaks in the sosd curve of the Template Attack. They arise due to

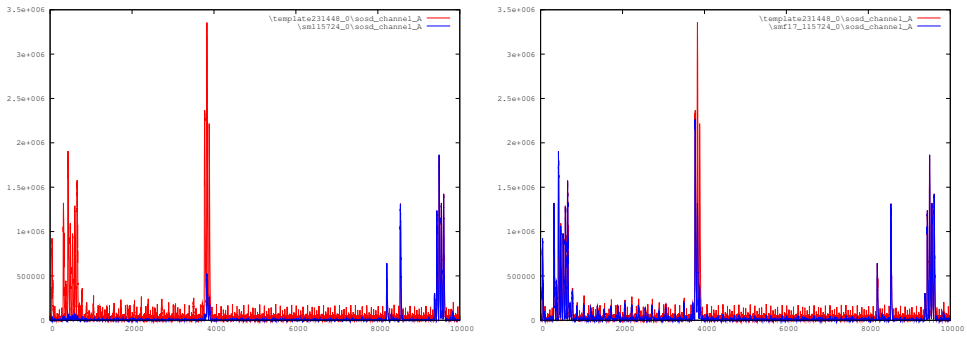


Figure 47: sosd curves the Stochastic Model (red) with $\mathcal{F}_9$ (left) and with $\mathcal{F}_{17}$ (right) and the Stochastic Model with $\mathcal{F}_{17}$ (blue) as functions of time

the same reasons as for the Template Attack (see observation 1). We added inclusion of the instant $x \oplus k$ but since the underlying samples represent a fixed k and the $\oplus$ operation is linear, the peaks indicate the different plaintexts x. As prior for the Template Attack, we modify the point selection algorithm to disregard all instants covering the initial Round Key addition, because they do not indicate key-dependent differences.
The peaks in the middle of the plot indicate parts of the combined SubBytes and ShiftRows transformations. The Template Attack can track an entire byte from $x \oplus k$ to ShiftRows(Sbox$(x \oplus k)$) and we are expecting the modified Stochastic Model to have the same ability, since it regards $x \oplus k$ and Sbox$(x \oplus k)$ while ShiftRows() is linear and should be covered automatically. But, as one can observe in the plots, the sosd curve of the Stochastic Model with $\mathcal{F}_{17}$ does not comprise all the peaks (in the middle of the plot) which can be seen in the sosd curve of the Template Attack. Since we implicitly added regard of $x \oplus k$, we assume that the high peak in the middle of the plot indicates the Sbox input. An explanatory approach for this difference:
The ShiftRows transformation can be disregarded because firstly it is linear and secondly, in the present case where we look at the first byte of the state array $s_{0,0}$, the transformation does not rearrange the bytes in the State.
The SubBytes transformation comprises the inversion of $s_{0,0}$ in GF(2^8) and an affine transformation that involves several bits of $s_{0,0}^{-1}$ per output bit. Since both sub-transformations compute each output bit from at least several in-

put bits, we assume that this is the difficulty and plan further research, see Section 9.1.

Another possible reason for the difference is the fact, that we implemented the Sbox as a table lookup and that the table's elements' addresses in the E^2PROM are not necessarily linear dependent on the lookup value. If so, the bit-wise attack can not correlate Sbox input bits to Sbox output bits, see Section 9.1.

7 Experimental Results - improved attacks

7.1 T-Test Template Attack

The enhanced signal to noise ratio in sost does nod lead to a higher number of selected points for a non-modified point selection algorithm. This is due to the fact that the algorithm still regards all points underneath 10% of the highest peak's value[23] as noise and hence disregards even peaks underneath that limit. A modified point selection algorithm which exploits the enhanced signal to noise ratio identifies up to 13 points.

The performance analysis which we subject the T-Test Template Attack to follows the usual procedure. We test all combinations $N_1 \times$ P $\times N_3$ for $N_1 \in \{3k, 5k, 10k, 20k, 30k, 40k, 50k, 231448\}$, P $\in \{13, 9, 6, 3\}$, and $N_3 \in \{1, 2, 5, 10\}$. Note in particular that due to the results of preliminary tests

- we increase the maximum value of P to 13
- we extend the interval of N_1 toward smaller values because the attack reaches the 100% boundary earlier than the original version.

For the sake of comparability, we provide results based on the non-modified point selection algorithm (9 points) and the adapted version (13 points). Furthermore, we only provide results for $N_1 = 231448$ resp. 10000, to stress the attacks reaction to a reduced N_1, as the overall observations are similar to those of the original attack.

The results are provided in the usual format. Table 11 presents the results for $N_1 = 231448$. The selection algorithms identify 9 resp. 13 points. As before, this set of points will be referred to as the optimal distribution. From the blocks containing the success rates one can observe the same behavior of the attack as for its original version, in particular the logarithmic increase of the success rate for increased values of P and N_3 and the fact that N_3 has bigger influence on the success rates than P.

[23] recall that sost generates a better highest peak to noise floor distance

$N_1 = 231448$	$p = 13$	channel = power
poi[]	3771, 8551, 9607, 3832, 3894, 9545, 9434, 9490, 3494, 3716, 8218, 9379, 9829	

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
13	22,9	62,2	98,9	100,0		24,1	68,4	99,5	100,0
9	18,4	57,2	98,9	100,0		18,9	58,4	99,3	100,0
6	15,0	48,4	95,9	100,0		16,1	48,9	97,2	100,0
3	4,8	17,4	67,9	96,4		5,5	20,7	73,3	98,0

Table 11: Success rates (SR) for $N_1 = 231448$ and channel = power

Table 12 presents the results for $N_1 = 10000$. The selection algorithms find 5 resp. 8 points from the optimal distribution and 4 resp. 5 points that are slightly displaced but still in the correct processor cycle. Note in particular that all selected points are related to the optimal distribution, there are no bad selections. From looking at the success rates one can observe that the

$N_1 = 10000$	$poi = 13$	channel = power
P	3772, 8551, 9607, 3832, 3888, 9546, 9435, 9490, 3494, 3716, 8218, 9379, 9826	

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
13	9,1	20,5	58,5	85,4		47,7	93,4	100,0	100,0
9	8,4	24,1	68,4	94,6		31,0	79,8	99,7	100,0
6	9,4	31,5	83,0	98,6		21,8	66,4	99,7	100,0
3	3,7	14,3	54,2	89,6		4,0	22,5	81,1	99,5

Table 12: Success rates (SR) for $N_1 = 10000$ and channel = power

correlation of the parameter P and the success rate has changed for the case of real classification. While the step from $P = 3$ to $P = 6$ yields an increased success rate, the algebraic sign is reversed for further steps to $P = 9$ and $P = 13$ so that an increased number of points yields a worse success rate. For the case of trial classification, the correlation's direction remains the same (cp. original attack).

7.1.1 Comparison Template Attack vs. T-Test Template Attack

In this section, we compare the efficiency of the original Template Attack and T-Test Template Attack with respect to the profiling step and the classification step, following the procedure introduced in Section 5.3.

Profiling Efficiency

Figure 48 shows the efficiency of both attacks in the profiling step. The applied measure is equal to the one in Section 5.3. The plot clearly indicates the

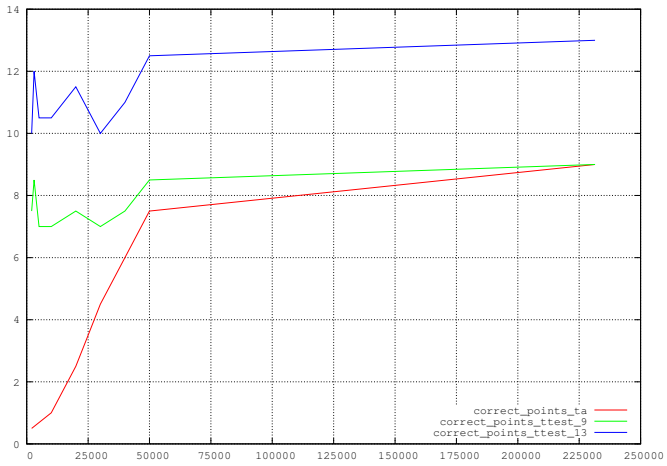


Figure 48: Well selected points in the profiling step

superiority of the improved version, the T-Test Template Attack, in terms of selecting the right instants and hence, in the profiling step. Considering Figure 44 again, the improved profiling efficiency obviously derives from the enhanced ability to suppress noise.

Classification Efficiency

In the following, we compare the classification success rates of the attacks. We restrict our attention to real classification, thus classification of samples from device B, since it is the more interesting case, $N_3 \in \{1, 10\}$ for the sake of clarity, and, each time, the optimal choice of P. The graph shows the most, that each attack can make of a given number of curves in the profiling step when classifying one resp. ten samples from device B.

Again, all individual numbers are provided in Appendixes B and D.

Figure 49 shows the success rates plotted as functions of N_1 for $N_3 \in \{1, 10\}$ and optimal choice of P. For the sake of comparability, we provide a plot of the T-Test Template Attack's success rates for $N_3 = 1$ where the range of points P is bounded by 9. For $N_3 = 10$, this is not necessary, as the success rates for a bounded and a non-bounded range of points are identical.

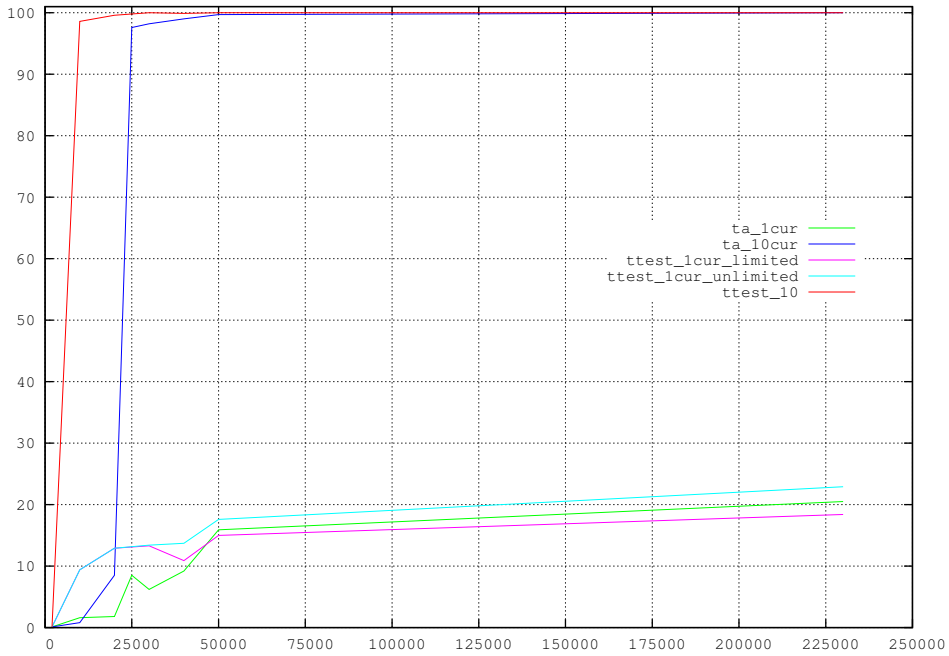


Figure 49: Plot of the Template Attack's and the T-Test Template Attack's success rates as a function of the number of curves in the profiling step. It is $N_3 \in \{1, 10\}$ and P chosen optimally.

7.2 High-Order Stochastic Model with F_{17}

The additional peaks in the sosd curve of the High-Order Stochastic Model with $\mathcal{F}_{17}$ lead to additionally selected points. The non-modified[24] point selection algorithm identifies p = 10 instead of 9 points. Figure 47 indicates, why

[24] the noise border remains at 10%; it omits instants 0 - 3300

a single additional point makes a significant difference. The tenth point is selected at an instant, which covers the combined SubBytes and ShiftRows transformation. That instant yields the strongest peak in the sosd curve which means that it is a very good criterion to distinguish key hypothesis. We do not consider a reduction of the noise border in the point selection algorithm, because the basis, on which the choice is made, remains the same (sosd).

The performance analysis of the High-Order Stochastic Model follows the usual procedure. We test all combinations $N_1 + N_2 \times \mathrm{P} \times N_3$ for $N_1 + N_2 \in \{2k, 10k, 20k, 30k, 40k, 50k, 231448\}$, $\mathrm{P} \in \{10, 9, 6, 3\}$, and $N_3 \in \{1, 2, 5, 10\}$. Note in particular that due to the results of preliminary tests

- we increase the maximum value of P to 10
- we extend the interval of N_1 toward smaller values because the attack reaches the 100% boundary earlier than the original version.

We only provide results for $N_1 + N_2 = 231448$ resp. 10000, to stress the attacks reaction to a reduced N_1 and for comparability with the T-Test Template Attack. The overall observations are similar to those of the original attack with $\mathcal{F}_9$. All result tables are provided in Appendix E.

The results are provided in the usual format. Table 13 presents the results for $N_1 + N_2 = 231448$.

$N_1 + N_2 = 231448$	$p = 10$	channel = power

poi[]	3774, 9496, 9607, 9551, 8551, 3829, 9440, 8218, 3884, 9385

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
10	9,8	26,6	84,0	99,5		10,2	33,8	88,9	99,4
9	9,1	29,7	83,2	99,8		8,8	30,3	88,0	99,9
6	4,7	22,3	76,6	99,2		8,4	23,6	79,6	99,3
3	3,1	13,0	55,1	92,1		4,8	15,3	61,7	96,4

Table 13: Success rates (SR) for N_1 and $N_2 = 115724$ and channel = power

The point selection algorithm identified p = 10 points, which will be referred to as the optimal distribution, as usual. From the blocks containing the success rates, one can observe the usual logarithmic dependency between P resp. N_3 and the success rates. Furthermore and in contrast to the original version, the attack shows a property similar to the Template Attacks. N_3 clearly has a stronger impact on the success rates than P.

Table 14 presents the results for $N_1 + N_2 = 10000$, hence less than 5% of the training samples. The point selection algorithm found five points

$N_1 + N_2 = 10000$	$poi = 10$	channel = power
poi[]	3774, 9496, 3829, 9605, 8551, 9441, 8218, 17679, 18346, 19791	

	SR of real classification				SR of trial classification			
P \ N_3	1	2	5	10	1	2	5	10
10	5,2	14,6	54,5	88,3	7,9	29,7	89,1	99,8
9	3,3	12,0	56,8	89,1	8,3	28,1	86,1	99,6
6	2,6	8,2	38,7	76,4	6,2	23,3	82,1	99,1
3	4,3	11,1	35,4	71,3	3,4	10,6	40,5	73,2

Table 14: Success rates (SR) for $N_1 + N_2 = 10000$ and channel = power

from the optimal distribution, two that are slightly displaced but still in the correct cycle, and three that are not related to the optimal distribution. As expected, the success rates for trial classification remain in the same order of magnitude. The success rates for real classification show the desired tendency. In opposition to the original attack, whose efficiency declines to $\sim$ 50% for such a reduction of training samples, the efficiency of the improved attack declines less, particularly for higher values of N_3 and P.

7.2.1 Comparison Stochastic Model vs. High-Order Stochastic Model

In this section, we compare the efficiency of the Stochastic Model with $\mathcal{F}_9$ and the High-Order Stochastic Model with $\mathcal{F}_{17}$ with respect to the profiling step and the classification step, following the usual procedure.

Profiling Efficiency

Figure 50 shows the efficiency of both attacks in the profiling step. The applied measure is equal to the one in Section 5.3. Since we did not modify

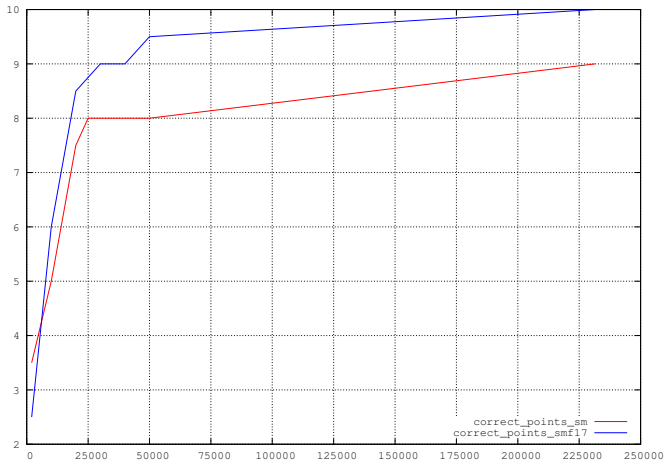


Figure 50: Well selected points in the profiling step

the basis, upon which the point selection algorithm operates (sosd), the two curves basically show a similar run. The different absolute values derive from the additional point that can be identified by The High-Order Attack due to the higher-dimensional vector subspace. As expected, the entropy of sosd and hence of each b-vector does not change. The analysis of the efficiency in the classification step will show, if additional b-vectors yield better results.

Classification Efficiency

In the following, we compare the classification success rates of both attacks. We restrict our attention to real classification, thus classification of samples from device B, since it is the more interesting case, $N_3 \in \{1, 10\}$ for the sake of clarity, and, each time, the optimal choice of P. The graph shows the most, that each attack can make of a given number of curves in the profiling step when classifying one resp. ten samples from device B.

Figure 51 shows the success rates plotted as functions of N_1 for $N_3 \in \{1, 10\}$ and optimal choice of P.

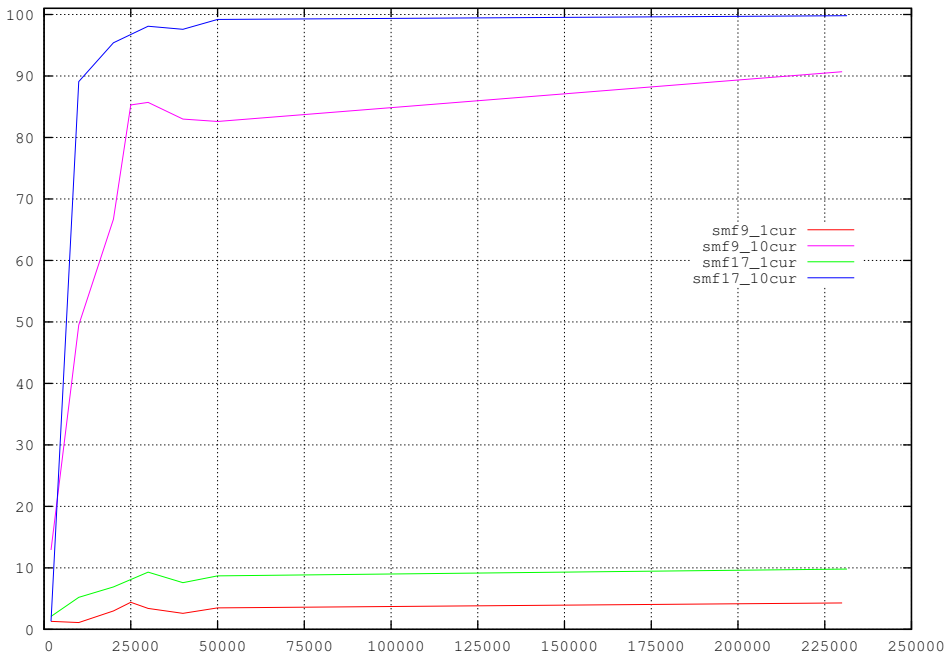


Figure 51: Plot of the Stochastic Model's and the High-Order Stochastic Model's success rates as a function of the number of curves in the profiling step. It is $N_3 \in \{1, 10\}$ and P chosen optimally.

The benefit of generating 8 more b-vectors with respect to the Sbox input is clearly visible. In opposition to the profiling efficiency, the efficiency in the classification step is significantly increased. For $N_1 + N_2 \leq 25000$, the improved attack reaches the same efficiency than the original attack, with at most the half amount of the training samples. This is in particular important, if one is bounded in the number of available samples in the profiling step. Furthermore, for $N_1 + N_2 > 25000$ and $N_3 = 10$, the High-Order Stochastic Model clearly exceeds the 90% success rate boundary and gets very close to 100% success.

7.3 Comparison of all four attacks

In this section we only provide two figures illustrating the efficiency of all four attacks in the profiling and classification step and a short summary of the observations. We provide them to give an overall survey of our work, the

individual plots will not be discussed in detail.

Figure 52 contrasts the profiling efficiency of all four attacks with respect to their individually optimised point selection algorithms and point selection basis. The T-Test Template Attack seems to be the best possible choice.

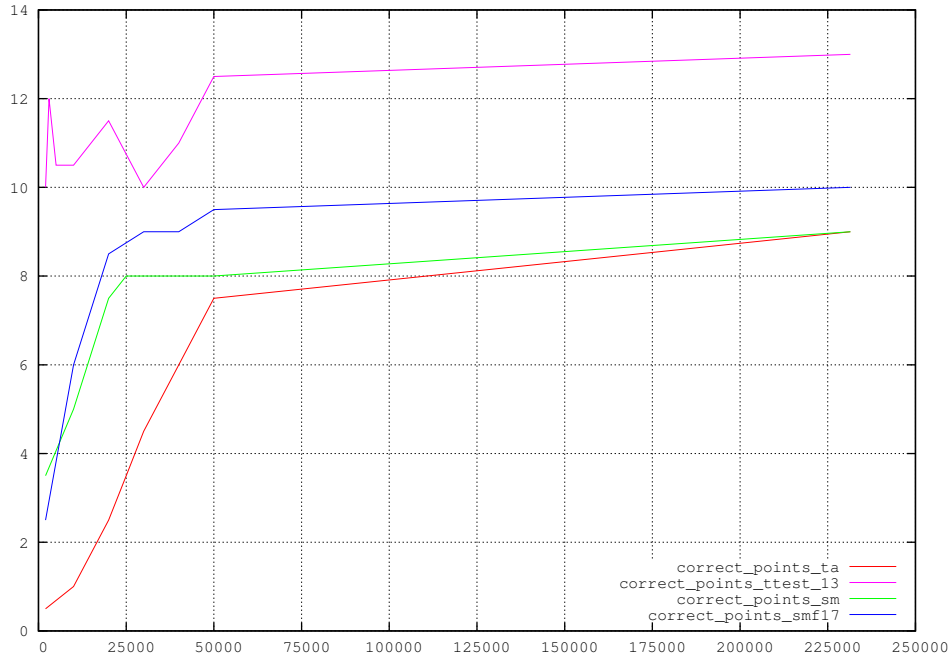


Figure 52: Plots of the individually optimised profiling performance of all four attacks as functions of N_1 resp. $N_1 + N_2$

Figure 53 contrasts the efficiency of all four attacks in the classification step, exploiting all of their individual optimisations. For almost all values of N_1 resp. $N_1 + N_2$, the T-Test Template Attack seems to be superior again.

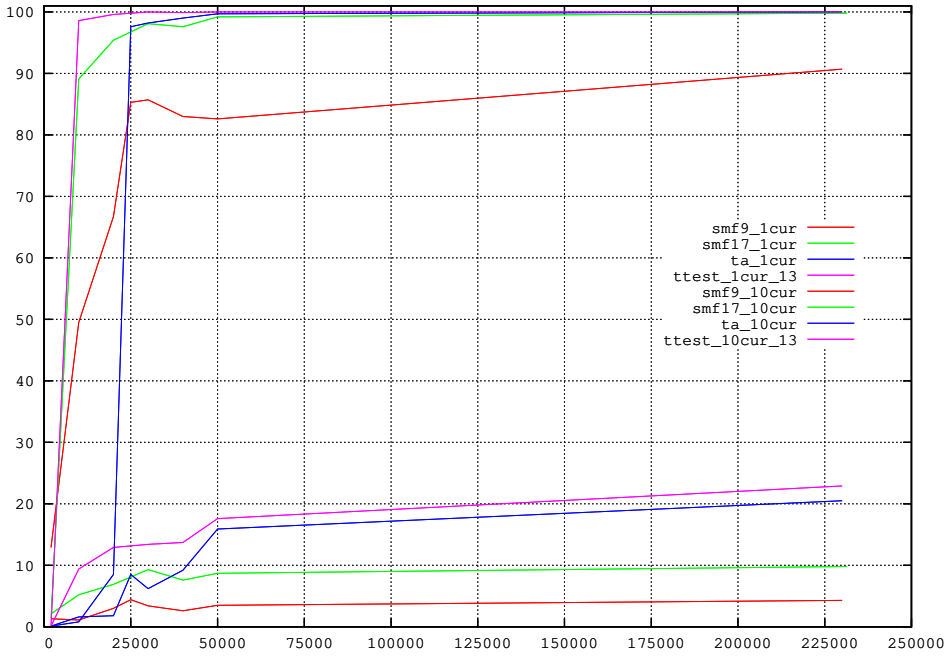


Figure 53: Plots of the individually optimised classification performance of all four attacks as functions of N_1 resp. $N_1 + N_2$, it is $N_3 \in \{1, 10\}$ and P chosen to maximise the success rate

8 Template Attacks, Stochastic Models, EM and Multi- Side Channel Attacks

All results reported so far are based on the side channel power consumption. In this section, we provide insights in our experience with the EM side channel and multichannel attacks.

In contrast to many publications, e.g. [16, 35, 36, 37, 34], we do not transform the EM samples into the frequency domain in order to isolate carrier frequencies and demodulate significant signals. We treat the EM samples and apply the attacks in exactly the same way as we did for the samples from the power channel. One might say, we perform a **Magnetic Flux Attack**, since the EM probe only acquires magnetic fields.

8.1 Electromagnetic Attacks

Apparently, our samples from the EM side channel are much more noisy than those of the power channel. Neither the Template Attack nor the Stochastic Model (both using sosd) are able to sufficiently suppress the noise and to extract the significant characteristics. Figure 54 shows the resulting sosd curves of the Stochastic Model $\mathcal{F}_9$ for $N_1 = 50000$ and 115724 samples.

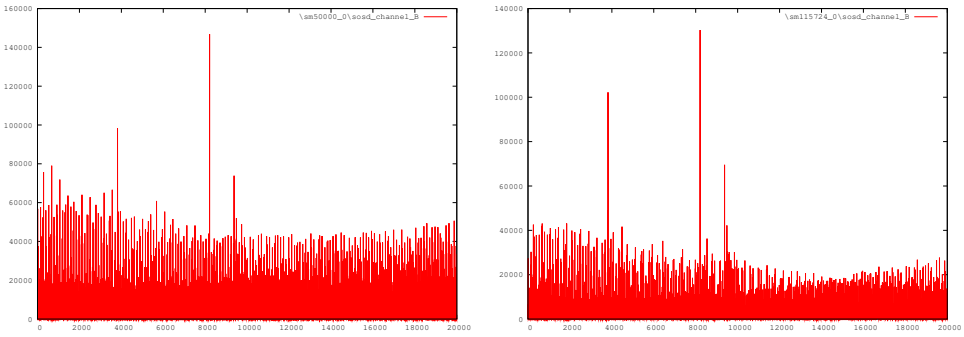


Figure 54: sosd curves of the Stochastic Model $\mathcal{F}_9$ derived from 50000 resp. 115724 samples of the EM channel

Although the plots show three significant peaks, the efficiency in the classification step does not exceed a success rate of 4.8%, which is achieved for

P = 6 and $N_3 = 10$. The reason for the low efficient profiling step is the high noise level in the samples, which probably derives from the bad resolution of the EM probe. With the probe covering an area of 5mm × 5mm, it is a daunting task to point at the part of the chip, that does the computation while keeping it away from the I/O busses and the power feed (cp. [16]). The non-efficient profiling (consider the low highest peak to noise floor ratio) explains the low success rates in the classification step.

We omit details on the original Template Attack's performance, because the derived sosd curves do not show any significant peaks and the success rates in the classification step do not exceed 1%.
Furthermore, we omit details on the High-Order Stochastic Model as well, as its performance does not considerably exceed the one of the original attack (it uses sosd as well) and focus on the most promising approach.

The T-Test Template Attack proved to be the most efficient attack operating on a small number of training samples which is equivalent to the ability of filtering noise in the profiling step. Hence, we focus on examining this attack's efficiency here.

Figure 55 shows the derived sost curves for $N_1 = 50000$ and 231448. One can see clear visible peaks and a reasonable highest peak to noise floor

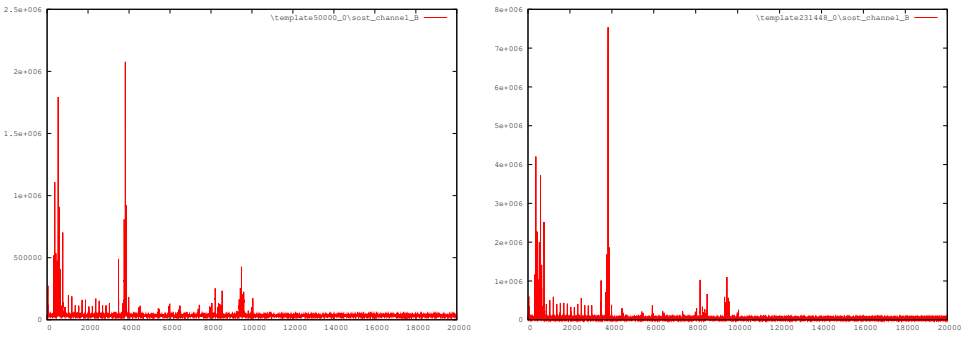


Figure 55: sost curves of the T-Test Template Attack derived from 50000 resp. 231448 samples of the EM channel

distance. Figure 56 compares the sost curves for the power channel and the EM channel derived from $N_1 = 50000$ samples. Note that the vertical axis is scaled logarithmic and we zoomed in on the Sbox lookup. One can observe

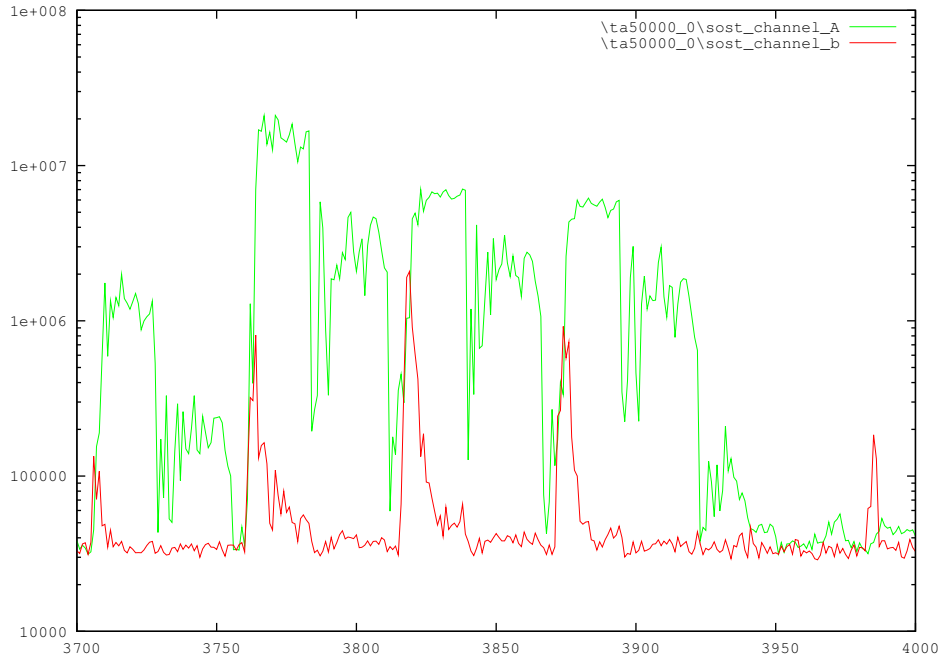


Figure 56: sost curves of the T-Test Template Attack derived from 50000 power and EM samples

very clearly that the electromagnetic radiation is related to the derivation of the power consumption.

8.1.1 Experimental Results

We apply the T-Test Template Attack to the samples of the EM side channel[25] in the same way as we did for the power channel. In particular, due to our experiences with the attack and preliminary tests, we use the reduced noise border in the point selection algorithm, that is 1% of the highest peak's value (cp. Section 7.1).

[25] measurement sets 1 and 2, fixed key

We test all combinations of $N_1 \times \mathrm{P} \times N_3$ for $N_1 \in \{50k, 231448\}$, P $\in \{34, 24, 14\}$, and $N_3 \in \{1, 2, 5, 10\}$. The results are presented in the usual format.

Table 15 presents the results for $N_1 = 231448$. The point selection al-

$N_1 = 231448$	$p = 34$	channel = EM

poi[]	3819, 3874, 3764, 9489, 8208, 3486, 3706, 8544,9375, 9545, 9600, 9434, 3986, 5931, 8321, 4484, 8043, 8432, 10043, 7374, 6429, 5429, 7985, 8376, 9988, 4539, 5484, 8487, 5987, 6487, 7429, 9822, 3319, 9266

	SR of real classification				SR of trial classification			
P \ N_3	1	2	5	10	1	2	5	10
34	3,2	8,1	25,9	51,2	8,1	23,4	67,3	96,1
24	3,4	8,4	23,3	46,4	6,0	16,2	54,2	89,3
14	1,9	5,7	17,2	37,4	3,0	9,9	34,0	68,9

Table 15: Success rates (SR) for $N_1 = 231448$ and channel = EM

gorithm identified 34 points, the optimal distribution. The two blocks of success rates indicate a non-optimal profiling, because the success rates for trial and real classification are not in the same order of magnitude. Whether a further increase of N_1 would yield a better profiling, is unknown. Since we do not have more than 231448 samples, we can not investigate this matter. A further indication for a non-optimal profiling is the fact, that in case of real classification the success rates barely increase from P = 24 to P = 34. Nevertheless, we want to investigate, what is possible under these circumstances although it is obvious, that the results can not compete against those from the power channel.

Table 16 presents the results for $N_1 = 50000$. The point selection algorithm found 19 points from the optimal distribution, 13 points are slightly displaced, and 2 are not related to the optimal distribution. Applying our defined measure, it yields a score of 26.5 correct points. The effect of the false positives is noticeable, since the success rates of trial classification increase while the ones of real classification decrease.

$N_1 = 50000$	$p = 34$	channel = EM

poi[]	3819, 3874, 3764, 3486, 9489, 9434, 8208, 8545, 9601, 9545, 3985, 10043, 9377, 8043, 3706, 8377, 5986, 8432, 7431, 6486, 8487, 4541, 5931, 7932, 8322, 9988, 4486, 5429, 7988, 7375, 5484, 6429, 9822, 9266

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
34	1,1	2,5	4,0	7,6		35,0	73,0	99,7	100,0
24	1,8	2,7	5,8	8,6		17,5	47,8	93,1	100,0
14	1,4	1,4	5,2	9,4		9,4	22,0	71,5	96,8

Table 16: Success rates (SR) for $N_1 = 50000$ and channel = EM

8.2 Multi-channel Attacks

For the reasons provided in Section 8.1, we focus on Multi-channel attacks with the T-Test Template Attack. The implementation effort is rather small, as we simply concatenate the side channel samples from the power channel (20000 points) and the EM channel (20000 points) to form a multi-channel sample (40000 points).

Figure 57 shows the sost curve derived from 231448 multi-channel samples and stresses the significantly lower signal to noise ratio in the EM samples (instants 20000 to 40000). Obviously, we will have to use a large number of points to select p and an even lower noise border in order to force the selection algorithm to identify points on the EM channel portion.

8.2.1 Experimental results

We apply the T-Test Template Attack using a reduced noise border at 0.1% of the highest peak's value for the point selection algorithm. Since we know that $p = 13$ for the power channel and $p = 34$ for the EM channel, we force the algorithm to identify up to $p = 47$ points. With respect to the (bad) success rates provided in 8.1, we restrict our attention to $N_1 = 231448$.

We test all combinations of $N_1 \times \mathrm{P} \times N_3$ for $N_1 = 231448$, $\mathrm{P} \in \{47, 40, 37,$

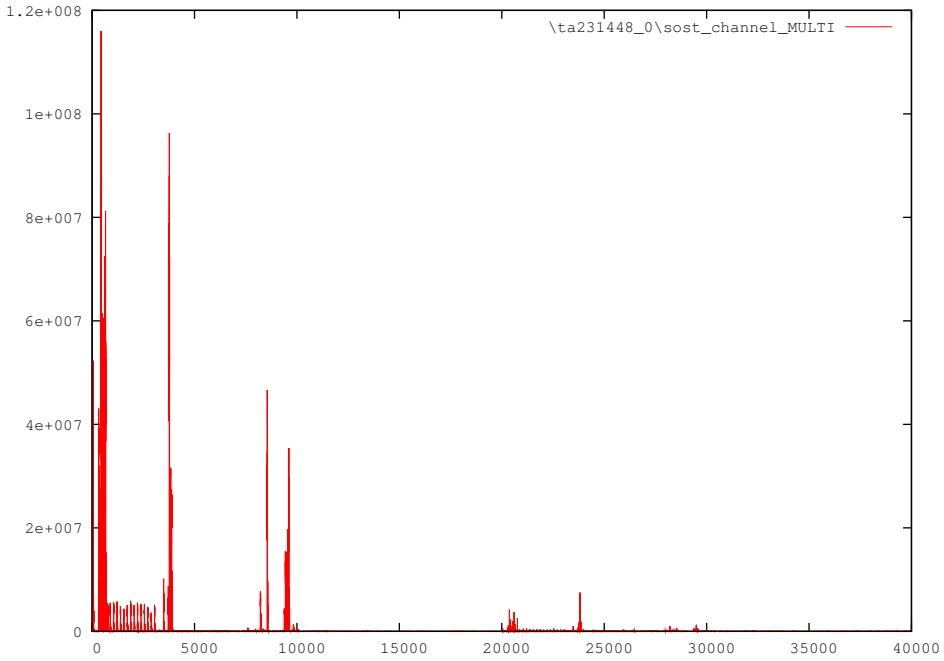


Figure 57: sost curve of the T-Test Template Attack derived from 231448 multi-channel samples

$30, 16\}$, and $N_3 \in \{1, 2, 5, 10\}$. The results are presented in the usual format.

As expected, the point selection algorithm only identifies instants from the power channel portion in the beginning. The first eleven selected points are the same as for a pure power attack. Only three points from the EM channel portion are selected, before the last point of the optimal distribution from a power attack (13 points) is selected. Hence, in order to be able to compare the results to those from the power attack, we provide results for $P = 16$ which reflects the selection for a noise border at 1%.
The two blocks of success rates indicate a non-optimal profiling. The difference between the success rates of trial and real classification is smaller than for a pure EM attack, but bigger than for a pure power attack. Obviously, the interacting effects of the power and EM profiling cancel each other out and reach an intermediate profiling efficiency. Nevertheless, the absolute values of the success rates indicate clearly, that the multi-channel attack is

$N_1 = 231448$	$p = 47$	channel = Multi

poi[]	3771, 8551, 9607, 3832, 3894, 9545, 9434, 9490, 3494, 3716, 8218, 23819, 9379, 23874, 23764, 9829, 29489, 28208, 23486, 7606, 23706, 28544, 29375, 29545, 29600, 29434, 10045, 23986, 25931, 8340 28321, 24484, 28043, 9773, 28432, 30043, 8410, 9885, 27374, 26429, 25429, 7951, 27985, 28376, 8006, 29988, 24539

	SR of real classification				SR of trial classification			
P \ N_3	1	2	5	10	1	2	5	10
47	26,6	68,5	96,2	99,9	48,5	88,7	100,0	100,0
40	26,9	69,2	98,4	99,8	41,3	84,3	99,9	100,0
37	24,7	67,6	98,1	100,0	40,6	84,6	100,0	100,0
30	26,6	69,5	98,3	100,0	36,6	81,2	99,9	100,0
16	25,1	66,0	99,5	100,0	29,9	69,9	99,8	100,0

Table 17: Success rates (SR) for $N_1 = 231448$ and channel = Multi

superior to the pure power attack. Particularly for $N_3 = 1$, one can observe a gain of up to 17% in the more interesting case of real classification. Even if we (theoretically) use the same noise border as for the power attack and compare the results for $P = 16$ to those of the power attack for $P = 13$, one can observe a gain of up to 10%.

9 Conclusion

In the course of this thesis, we profoundly analysed the Template Attack and the Stochastic Model. We identified relevant parameters for each attack, analysed their impact and explained their influences on the attacks' efficiencies. For each attack, we elaborated weaknesses and strengths.
Furthermore, we suggested improvements to both attacks with respect to at least one of their weaknesses and proved the increased efficiency.

We demonstrated that one of the improved attacks, even though applied to noisy EM side channel measurements in a non-sophisticated manner, yields remarkable results. Due to this increased efficiency, we were able to mount a multi-channel attack which can yield results that are noticeable superior to those of single-channel attacks.

We were able to show that the assumptions on an adversary's powers should be weakened in the context of two-step side channel attacks on block ciphers like the Advanced Encryption Standard. In particular we demonstrated, that the training device does not need to be programmable by the adversary, if the utilisation of a fixed key may be assumed.

9.1 Further Research

We have commenced promising research on further extensions of the vector subspace of the High-Order Stochastic Model. We add more dimensions whose selection functions aim at the centre of the Sbox lookup, that is the intermediate result after $(x \oplus k)^{-1}$ in GF(2^8) and before the affine function. Another approach, which we analyse at the moment, are vector bases that evaluate the logic AND sum of several Sbox input or output bits.

Since we developed powerful attacks which yield success rates of more than 25% given a single sample in the classification step, we plan to analyse their efficiency against protected implementations, in particular AES with boolean and arithmetic masking.

During the course of this work, we used a point selection algorithm that selects at most one point per processor cycle. First experiments have shown, that the choice of several points per cycle, hence characterisation of its shape, can yield improved results under certain conditions.

Topics that need further investigation are:

- the determination of characteristic differences, is **sost** optimal?
- point selection algorithms, one can imagine self learning solutions
- choices for the vector subspace with respect to Stochastic Models
- determination of a measure for side channel quality
- evaluation of the improved attacks in EM- and multichannel settings, when applied in more sophisticated attacks (demodulation)

References

[1] M. Matsui, Linear cryptanalysis method for DES cipher, In Advances in Cryptology - EUROCRYPT '93, LNCS 765, pp. 386-397, Springer Verlag, 2003

[2] E. Biham, A. Shamir, Differential Cryptanalysis of the full 16-Round DES, In Proceedings of Crypto '92, LNCS 740, Springer Varlag, 1991

[3] National Institute of Standards and Technology (NIST), Data Encryption Standard (DES), Federal Information Processing Standards Publication 46, 1977

[4] NIST, Announcing Request for candidate algorithm nominations for the Advanced Encryption Standard (AES), `http://csrc.nist.giv/CryptoToolkit/aes/pre-round1/aes-9709.htm`, 12. September 1997

[5] J. Daemen, V. Rijmen, The design of Rijndael: AES - the Advanced Encryption Standard, Springer Verlag, 2002

[6] National Institute of Standards and Technology (NIST), Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, 2001

[7] J. Daemon, V. Rijmen, AES submission document in Rijndael, Version 2, September 1999, `http://csrc.nist.goc/CryptoToolkit/aes/rijndael/Rijndael.pdf`

[8] Menezes, van Oorschot, Vanstone, The Handbook of applied Cryptography, CRC Press, 1996

[9] P.N. Fahn, P.K. Pearson, IPA: A new Class of Power Attacks, In Proceedings Workshop on Cryptographic Hardware and Embedded Systems, LNCS 1717, pp. 173-186, Springer Verlag, 1999

[10] S. Chari, J.R. Rao, P. Rohatgi, Template Attacks, In Proceedings Workshop on Cryptographic Hardware and Embedded Systems, LNCS 2523, pp. 13-28, Springer Verlag, 2002

[11] W. Schindler, K. Lemke, C. Paar, A Stochastic Model for Differential Side Channel Cryptanalysis, In Proceedings Workshop on Cryptographic Hardware and Embedded Systems, LNCS 3659, pp. 30-46, Springer Verlag, 2005

[12] Ecrypt - Network of Exellence in Cryptology, Side Channel Cryptanalysis Lounge, http://crypto.rub.de/en_sclounge.html

[13] P. Kocher, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, In Advances in Cryptology - Crypto 96, pp. 104 - 113, Springer Verlag, 1996

[14] P. Kocher, J. Jaffe, B. Jun, Differential Power Analysis, In Advances in Cryptography - Crypto 99, LNCS 1666, pp. 388-397, Springer Verlag, 1999

[15] J.-J. Quisquater, D. Samyde, ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards, In Proceedings Smart Card Programming and Security, 2001, Springer Verlag, LNCS 2140

[16] K. Gandolfi, C. Mourtel, F. Olivier, Electromagnetic Analysis: Concrete Results, In Proceedings Cryptographic Hardware and Embedded Systems, 2001, Springer Verlag, LNCS 2162, pp. 251-261

[17] Funcard with ATMega163 microcontroller, vendor: Open Platform

[18] Atmel Corporation, 2006, http://www.atmel.com/dyn/products/product_card.asp?familiy_id=607\&familiy_name=AVR+8\%2DBit+RISC+\&part_id=2027

[19] Agilent Technologies, 2005, http://www.home.agilent.com/USeng/nav/-11496.536882503/pd.html

[20] Agilent Technologies, 2005, http://www.home.agilent.com/USeng/nav/-536899907.0/cp.html

[21] Langer EMV Technik, http://www.langer-emv.de/en/produkte/prod_rf2.htm

[22] Langer EMV Technik, http://www.langer-emv.de/en/produkte/prod_pa.htm

[23] Towitoko Chipdrive micro card reader, obsolete product, now SMC Microsystems GmbH, http://www.chipdrive.de

[24] Simple Operating System for Smartcard Education, http://www.mbsks.franken.de/sosse/

[25] ISO/IEC 7816 [1-6], Identification cards – integrated circuit(s) cards with contacts, http://www.iso.org

[26] Atmel Corporation, AVR Studio 4, 2006, http://www.atmel.com/dyn/products/tools.asp?family_id=607

[27] AVR-GCC made available through the GNU project, http://www.avrfreaks.net/AVRGCC/

[28] Since the programmer and the software are (at least in Germany) often used to illegally access Pay-TV channels, we were not able to determine their actual origin. However, to buy resp. download them on the internet is not a problem.

[29] Rankl, Effing, Handbuch der Chipkarten, Hanser Verlag, 1995

[30] C. Ash, The Probability Tutoring Book, IEEE Press, 1993

[31] Zwillinger (edt.), Standard Mathematical Tables and Formulae, CRC Press, 1996

[32] Bronstein et al., Taschenbuch der Mathematik, Harri Deutsch Verlag, 2000

[33] Trochim, William M., The Research Methods Knowledge Base, 2nd Edition, http://trochim.human.cornell.edu/kb/index.htm, January 16 2005

[34] D. Agrwal, J. Rao, P. Rohatgi, Multi-channel Attacks, In Proceedings Cryptographic Hardware and Embedded Systems, LNCS 2779, pp. 2-16, Springer Verlag, 2003

[35] C. Rechberger, Side Channel Analysis of Stream Ciphers, Master Thesis, Technical University Graz, 2004

[36] D. Agrawal, J. Rao, P. Rohatgi, The EM Side-Channel(s), In Proceedings Cryptographic Hardware and Embedded Systems, LNCS 2523, pp. 29-45, Springer Verlag, 2002

[37] National Security Agency (NSA), Tempest Series, `http://cryptome.org/#NSA--TS`

[38] M.-L. Akkar, R. Bevan, P. Dischamp, D. Moyart, Power Analysis, What Is Now Possible..., In Advances in Cryptography - Asiacrypt 2000, LNCS 1976, pp. 489-502, Springer Verlag, 2000

A Measurement setup illustrations

Figure 58: Langer EMV Technik RFU 5-2 near field probe

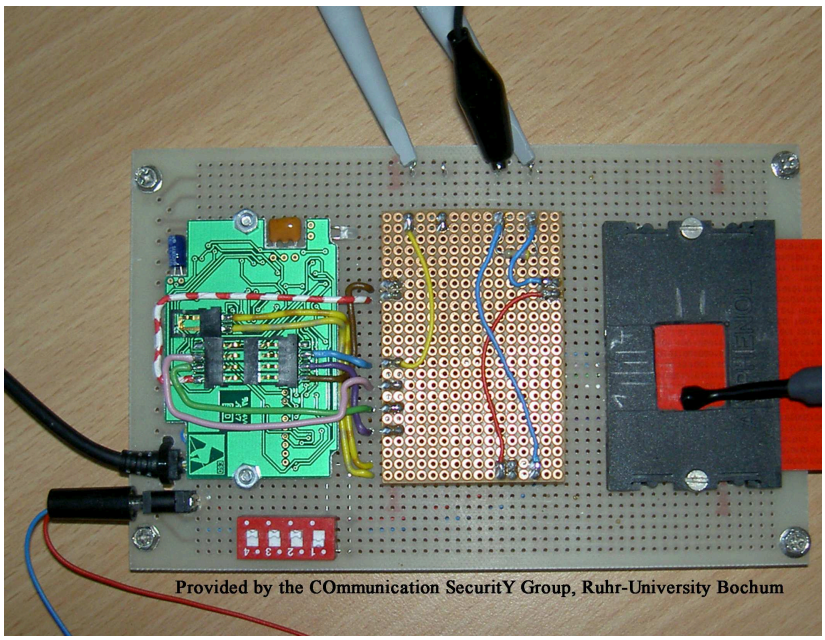


Figure 59: Dismantled Smartcard reader

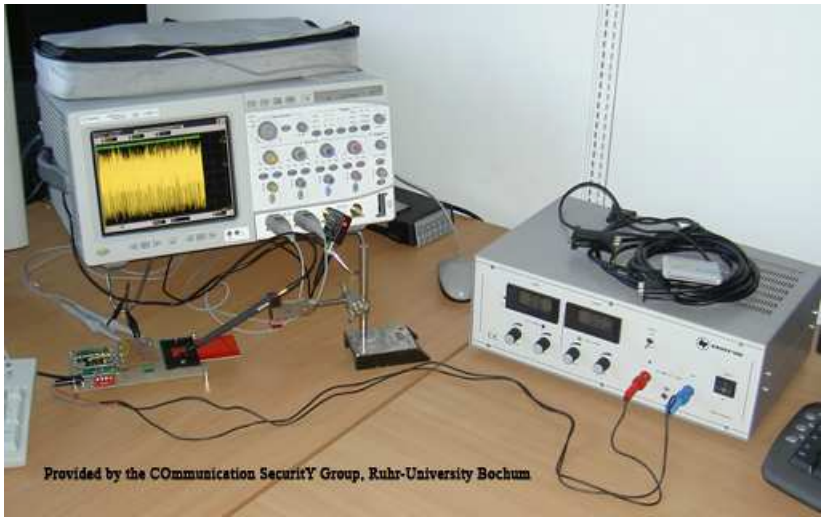


Figure 60: Side Channel measurement setup at COSY lab

B Result tables - Template Attack

$N_1 = 231448$	$p = 9$	channel = power

poi[]	3771, 3828, 3883, 8218, 8551, 9440, 9496, 9551, 9607

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	20,5	56,0	97,8	99,9		19,7	60,3	98,6	100,0
6	14,2	43,1	94,3	100,0		16,7	47,2	97,6	99,9
3	8,5	29,2	82,1	99,2		9,7	25,0	81,1	99,6

$N_1 = 50000$	$pp = 9$	channel = power

poi[]	3828, 3771, 3883, 9496, 9607, 9552, 8551, 9440, 19235

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	15,9	45,8	92,5	99,6		23,5	61,9	99,4	100,0
6	13,5	44,4	93,3	99,7		15,4	52,9	97,6	100,0
3	9,5	27,5	77,2	98,2		8,0	28,6	83,0	98,5

$N_1 = 40000$	$p = 9$	channel = power

poi[]	3828, 3771, 3883, 9496, 9620, 8564, 9552, 9441, 19235

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	7,0	25,9	75,3	98,0		15,8	53,1	97,3	100,0
6	9,2	29,9	80,8	98,9		13,5	38,2	92,0	100,0
3	8,3	26,6	76,	99,0		8,2	28,7	83,7	99,2

$N_1 = 30000$	$p = 9$	channel = power

poi[]	3828, 3771, 3884, 9509, 9620, 8564, 9441, 17790, 19235

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	5,3	15,5	55,0	87,8		13,0	46,1	93,8	100,0
6	6,2	18,7	68,4	95,2		10,4	32,2	87,2	99,7
3	6,1	24,0	77,1	98,2		9,9	29,1	85,0	98,9

$N_1 = 25000$	$p = 9$	channel = power

poi[]	3828, 3772, 3884, 9509, 9620, 8564, 17790, 19235, 15289

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	3,4	9,5	32,3	61,5		13,4	42,8	93,7	100,0
6	6,6	21,0	65,9	96,3		11,4	35,9	89,0	100,0
3	8,5	24,4	78,1	97,6		9,0	30,7	85,4	99,0

$N_1 = 20000$	$p = 9$	channel = power

poi[]	3840, 9509, 3774, 9620, 8564, 17790, 19235, 14900, 13066

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	0,7	1,5	2,4	3,6		6,7	22,3	74,1	97,8
6	1,8	1,5	3,1	4,3		5,3	15,5	54,8	89,9
3	1,4	2,8	6,0	8,5		2,7	7,3	27,4	57,7

$N_1 = 10000$	$p = 9$	channel = power

poi[]	3840, 9509, 4896, 13066, 12677, 10176, 7397, 10565, 10954

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	0.5	0.5	0.1	0.6		0.3	1.0	0.6	1.6
6	0.8	0.7	0.7	0.8		0.5	0.5	1.0	0.9
3	1.6	0.2	0.7	0.4		0.4	0.4	1.1	0.8

C Result tables - Stochastic Model

$N_1 + N_2 = 231448$	$p = 9$	channel = power

poi[]	9496, 9607, 9551, 8551, 9440, 8218, 3828, 9385, 3883

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	4,3	12,6	52,1	90,7		3,3	12,2	63,1	96,3
6	1,9	8,1	43,0	91,2		3,1	9,4	55,2	94,4
3	1,9	4,9	23,6	68,5		1,0	6,1	34,1	80,0

$N_1 + N_2 = 50000$	$p = 9$	channel = power

poi[]	9496, 9607, 8551 ,9552, 9441, 8218, 3828, 9385, 3883

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	3,5	8,7	37.6	82.6		3.3	10.6	61.7	98.2
6	2,1	5,9	37.1	79.3		3.2	9.6	55.6	95.9
3	1.1	6.0	30.8	77.6		2.0	6.7	43.1	88.2

$N_1 + N_2 = 40000$	$p = 9$	channel = power

poi[]	9496, 9607, 8551, 9552, 9441, 8218, 3828, 9385, 3883

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	2,1	9,4	36,9	83,0		3,2	11,8	61,7	97,7
6	2,6	6,3	33,9	80,4		3,1	9,4	59,5	97,0
3	1,4	5,0	28,4	77,7		3,2	5,7	40,4	91,0

$N_1 + N_2 = 30000$	$p = 9$	channel = power

poi[]	9496, 9607, 9552, 8551, 9441, 8218, 3828, 9385, 3883

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	3,4	7,0	42,2	85,7		3,7	13,4	66.5	97.5
6	1,8	8,2	36,6	83,0		2,7	9,5	59,1	96,5
3	1,2	3,5	17,9	57,9		1,7	3,6	35,1	81,0

$N_1 + N_2 = 25000$	$p = 9$	channel = power

poi[]	9496, 9607, 9552, 8551, 9441, 8218, 3828, 9385, 3883

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	4,4	8,0	39,8	85,3		3,1	13,9	64,1	97,6
6	2,3	7,1	33,8	80,6		2,4	10,5	57,8	97,1
3	0,9	4,2	19,2	61,4		1,2	4,6	34,4	81,6

$N_1 + N_2 = 20000$	$p = 9$	channel = power

poi[]	9496, 9608, 9552, 9441, 8551, 8218, 3828, 9385, 3883

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	3,0	6,2	27,2	66,7		3,9	13,9	66,3	98,1
6	2,0	4,5	21,5	61,8		2,0	9,0	58,0	96,0
3	0,9	1,9	8,8	27,6		1,8	3,6	31,2	83,5

$N_1 + N_2 = 10000$	$p = 9$	channel = power

poi[]	9496, 9605, 8551, 9441, 8218, 3828, 15845, 12677, 13344

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	1,0	1,9	16,9	46,9		4,0	10,8	58,3	96,1
6	1,1	2,4	15,5	49,5		3,2	11,4	53,8	93,9
3	0,4	1,7	9,7	34,0		2,0	5,2	38,8	88,9

$N_1 + N_2 = 2000$	$poi = 9$	channel = power

poi[]	9497, 9439, 8564, 9605, 8230, 3828, 5396, 5118, 4729

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
9	0,6	1,2	3,2	8,9		3,0	6,9	26,7	61,0
6	1,3	2,3	4,9	12,9		1,9	6,8	32,3	76,1
3	0,8	1,9	4,1	10,1		0,7	2,2	7,1	20,5

D Result tables - T-Test Template Attack

$N_1 = 231448$	$p = 13$)	channel = power

poi[]	3771, 8551, 9607, 3832, 3894, 9545, 9434, 9490, 3494, 3716, 8218, 9379, 9829

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
13	22,9	62,2	98,9	100,0		24,1	68,4	99,5	100,0
9	18,4	57,2	98,9	100,0		18,9	58,4	99,3	100,0
6	15,0	48,4	95,9	100,0		16,1	48,9	97,2	100,0
3	4,8	17,4	67,9	96,4		5,5	20,7	73,3	98,0

$N_1 = 50000$	$p = 13$)	channel = power

poi[]	3771, 8551, 9607, 3838, 3894, 9545, 9434, 9490, 3494, 3716, 8218, 9379, 9829

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
13	17,6	53,3	96,7	100,0		32,9	75,7	99,9	100,0
9	15,0	48,1	96,7	100,0		23,8	67,3	99,6	100,0
6	12,2	44,0	93,8	100,0		16,0	52,9	98,4	100,0
3	5,3	16,0	66,5	95,5		5,7	21,0	77,9	98,3

$N_1 = 40000$	$p = 13$	channel = power

poi[]	3767, 8551, 9607, 3838, 3894, 9546, 9435, 9490, 3494, 8218, 3710, 9379, 9835

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
13	13,7	32,1	85,9	99,6		29,1	71,4	99,9	100,0
9	10,9	35,2	90,8	99,9		24,5	63,1	99,3	100,0
6	10,4	37,3	89,8	99,9		18,4	51,7	97,5	100,0
3	4,9	14,1	64,0	95,0		4,7	23,2	75,8	98,6

$N_1 = 30000$	$p = 13$	channel = power

poi[]	3767, 8551, 9607, 3832, 3888, 9546, 9435, 9490, 3494, 8218, 3710, 9379, 9835

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
13	13,4	31,8	85,8	99,5		31,1	76,7	99,6	100,0
9	13,3	37,8	88,7	100,0		21,6	66,0	98,8	100,0
6	9,6	36,3	89,6	99,8		19,6	55,7	98,3	99,9
3	4,3	12,6	60,0	93,5		6,6	21,3	75,3	98,3

$N_1 = 20000$	$p = 13$	channel = power

poi[]	3771, 8551, 9607, 3832, 3888, 9546, 9435, 9490, 3494, 3716, 8218, 9379, 9826

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
13	12,6	33,0	83,2	99,2		36,7	84,5	99,9	100,0
9	11,1	33,2	87,1	99,4		26,6	70,9	99,6	100,0
6	12,9	34,9	89,4	99,6		18,3	57,4	98,4	100,0
3	3,3	14,6	59,7	93,8		6,4	21,0	78,6	98,9

$N_1 = 10000$	$p = 13$	channel = power

poi[]	3772, 8551, 9607, 3832, 3888, 9546, 9435, 9490, 3494, 3716, 8218, 9379, 9826

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
13	9,1	20,5	58,5	85,4		47,7	93,4	100,0	100,0
9	8,4	24,1	68,4	94,6		31,0	79,8	99,7	100,0
6	9,4	31,5	83,0	98,6		21,8	66,4	99,7	100,0
3	3,7	14,3	54,2	89,6		4,0	22,5	81,1	99,5

$N_1 = 5000$	$p = 13$	channel = power
poi[]	3771, 8551 ,9607, 3839, 3894, 9546, 9435, 9490, 3499, 3716, 8218, 9379, 9826	

	SR of real classification		SR of trial classification
	`not computable`		`not computable`

$N_1 = 3000$	$p = 13$	channel = power
poi[]	3771, 8551, 9607, 3839, 3894, 9545, 9434, 9490, 3494, 3716, 8218, 9379, 9826	

	SR of real classification		SR of trial classification
	`not computable`		`not computable`

E Result tables - High-Order Stochastic Model

$N_1 + N_2 = 231448$	$p = 10$	channel = power

poi[]	3774, 9496, 9607, 9551, 8551, 3829, 9440, 8218, 3884, 9385

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
10	9,8	26,6	84,0	99,5		10,2	33,8	88,9	99,4
9	9,1	29,7	83,2	99,8		8,8	30,3	88,0	99,9
6	4,7	22,3	76,6	99,2		8,4	23,6	79,6	99,3
3	3,1	13,0	55,1	92,1		4,8	15,3	61,7	96,4

$N_1 + N_2 = 50000$	$p = 10$	channel = power

poi[]	3774, 9496, 9607, 8551, 9552, 3829, 9440, 8218, 3884, 9385

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
10	7,9	27,6	82,9	99,2		9,9	32,0	89,2	99,9
9	8,7	28,3	80,5	99,1		9,6	30,9	87,6	99,6
6	5,0	19,3	75,1	97,3		5,1	23,6	81,2	99,1
3	3,5	11,7	55,9	90,8		4,8	14,8	63,3	95,7

$N_1 + N_2 = 40000$	$p = 10$	channel = power

poi[]	3774, 9496, 9607, 3829, 8551, 9552, 9441, 8218, 3884, 9385

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
10	7,6	24,3	77,2	97,5		10,0	32,3	91,6	99,6
9	6,6	23,2	72,6	97,6		9,5	36,1	88,1	99,8
6	6,7	18,6	69,7	96,9		6,7	25,2	83,3	98,8
3	3,4	12,1	49,9	89,8		4,0	16,4	66,3	96,0

$N_1 + N_2 = 30000$	$p = 10$	channel = power

poi[]	3774, 9496, 9607, 3829, 9552, 8551, 9441, 8218, 3884, 9385

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
10	9,3	25,1	73,7	97,7		10,2	33,4	90,2	99,8
9	8,4	22,5	74,0	98,1		10,1	33,1	90,2	99,9
6	5,1	22,4	68,3	95,6		6,3	23,0	82,4	99,9
3	4,2	10,2	52,0	88,1		4,8	16,5	65,6	96,4

$N_1 + N_2 = 20000$	$p = 10$	channel = power

poi[]	3774, 9496, 9608, 3829, 9441, 9552, 8551, 8218, 3884, 9385

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
10	6,6	20,0	65,0	95,4					
9	6,9	18,4	65,1	94,8		10,2	33,4	89,2	99,5
6	3,3	10,1	41,3	75,7		5,6	21,8	80,3	98,1
3	2,5	8,2	33,5	73,6		4,1	17,0	64,8	96,7

$N_1 + N_2 = 10000$	$p = 10$	channel = power

poi[]	3774, 9496, 3829, 9605, 8551, 9441, 8218, 17679, 18346, 19791

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
10	5,2	14,6	54,5	88,3		7,9	29,7	89,1	99,8
9	3,3	12,0	56,8	89,1		8,3	28,1	86,1	99,6
6	2,6	8,2	38,7	76,4		6,2	23,3	82,1	99,1
3	4,3	11,1	35,4	71,3		3,4	10,6	40,5	73,2

$N_1 + N_2 = 2000$	$p = 10$	channel = power

poi[]	9453, 3784, 3840, 9564, 8564, 6841, 3562, 5396, 5118, 4729

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
10	1,0	0,2	0,2	0,6		1,3	2,1	7,4	9,5
9	2,1	0,8	0,5	0,8		1,9	2,2	5,1	9,3
6	0,3	0,7	0,4	0,8		1,0	1,9	3,9	7,6
3	0,8	0,7	0,9	1,3		0,7	1,3	1,8	3,5

F Result tables - T-Test Template Attack, EM

$N_1 = 231448$	$p = 34$	channel = EM

poi[]	3819, 3874, 3764, 9489, 8208, 3486, 3706, 8544,9375, 9545, 9600, 9434, 3986, 5931, 8321, 4484, 8043, 8432, 10043, 7374, 6429, 5429, 7985, 8376, 9988, 4539, 5484, 8487, 5987, 6487, 7429, 9822, 3319, 9266

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
34	3,2	8,1	25,9	51,2		8,1	23,4	67,3	96,1
24	3,4	8,4	23,3	46,4		6,0	16,2	54,2	89,3
14	1,9	5,7	17,2	37,4		3,0	9,9	34,0	68,9

$N_1 = 50000$	$p = 34$	channel = EM

poi[]	3819, 3874, 3764, 3486, 9489, 9434, 8208, 8545, 9601, 9545, 3985, 10043, 9377, 8043, 3706, 8377, 5986, 8432, 7431, 6486, 8487, 4541, 5931, 7932, 8322, 9988, 4486, 5429, 7988, 7375, 5484, 6429, 9822, 9266

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
34	1,1	2,5	4,0	7,6		35,0	73,0	99,7	100,0
24	1,8	2,7	5,8	8,6		17,5	47,8	93,1	100,0
14	1,4	1,4	5,2	9,4		9,4	22,0	71,5	96,8

G Result tables - T-Test Template Attack, Multi-channel

$N_1 = 231448$	$p = 47$	channel = Multi

poi[]	3771, 8551, 9607, 3832, 3894, 9545, 9434, 9490, 3494, 3716, 8218, 23819, 9379, 23874, 23764, 9829, 29489, 28208, 23486, 7606, 23706, 28544, 29375, 29545, 29600, 29434, 10045, 23986, 25931, 8340 28321, 24484, 28043, 9773, 28432, 30043, 8410, 9885, 27374, 26429, 25429, 7951, 27985, 28376, 8006, 29988, 24539

	SR of real classification					SR of trial classification			
P \ N_3	1	2	5	10		1	2	5	10
47	26,6	68,5	96,2	99,9		48,5	88,7	100,0	100,0
40	26,9	69,2	98,4	99,8		41,3	84,3	99,9	100,0
37	24,7	67,6	98,1	100,0		40,6	84,6	100,0	100,0
30	26,6	69,5	98,3	100,0		36,6	81,2	99,9	100,0
16	25,1	66,0	99,5	100,0		29,9	69,9	99,8	100,0