Intrinsic Side-Channel Analysis Resistance and Efficient Masking

A case study of the use of SCA-related metrics and of design strategies leading to low-cost masking for CAESAR candidates

By Ko Stoffelen

Master Thesis in Computing Science
Supervised by dr. Lejla Batina
Second reader: prof.dr. Joan Daemen
Radboud University, Nijmegen
August 2015

Abstract

Metrics such as the transparency order and the confusion coefficient have been proposed to assess the intrinsic resistance to SCA attacks of a given S-box at the design stage. We extensively compare the metrics that have appeared in literature and apply them to the S-boxes used in the CAESAR competition for new authenticated encryption schemes. Although the most promising metrics are consistent in their predictions and behave as expected under different circumstances, the results are not reflected by CPA simulation results.

We then look at cipher design strategies that reduce the costs of masking. We compute the multiplicative complexity of the S-boxes by encoding the problem in logic and feeding it to SAT solvers, and we provide implementations with a provably minimum number of nonlinear operations, for which the cost of masking is quadratic in the number of gates. We also compare fundamental high-level operations used by the CAESAR candidates and show which ciphers are expected to have the lowest masking costs and why.

Acknowledgements

This thesis was submitted in partial fulfilment of the requirements for the degree of Master of Science in Computing Science. The research has been conducted within the Digital Security department of the Institute for Computing and Information Sciences at Radboud University in Nijmegen, the Netherlands.

First of all, I would like to take the opportunity to thank Lejla Batina, my supervisor, for helping me out whenever and wherever she could. Secondly, I am thankful to Kostas Papagiannopoulos for acting as second supervisor and for showing so much interest in this work. My thanks also go out to Joan Daemen for carefully reading my thesis, to Stjepan Picek for providing me with the latest version of his S-box Evaluation Tool, and to Theodosis Mourouzis for providing more details on his circuit encoding methodology.

Finally, I am very grateful to my family, and in particular to my parents and my brothers, for having no clue what all of this is about but who support me unconditionally nonetheless, and to my friends. Thank you all.

Contents

Chapter 1

Introduction

We are living in a world where everything has become digital and everything has become connected. The amount of data travelling around the globe every millisecond is simply staggering. It is in these days that we are starting to understand the tremendous nature of the consequences. Many billions of dollars and even lives depend on pieces of programming code held together by duct tape and created by humans, who inherently make mistakes. All of this would collapse immediately without notions of security and reliability.

Security is about regulating access to assets. Over the last few decades, researches have created a large portfolio of methods to do this regulation. Some of them turned out to be not secure at all, some have been proven secure under certain assumptions and with all remaining methods, we merely hope for the best. A few methods in particular have enjoyed extensive amounts of research and still stand, such that they are considered the best methods to provide security in certain contexts. However, with time and progress, the notion of what exactly is secure, changes. Not just because computers become faster and smarter, but also because even better attacks can sometimes be found.

One such major breakthrough occurred around 1998. Methods that were secure in theory[1], turned out to be breakable in practice, when implementation details and physical characteristics of the devices executing the methods are added to the equation. Think, for instance, of the power consumption of the device, or of the electromagnetic field that it radiates. Being able to use such additional information changed the notion of security. In fact, this discovery lead to completely new fields of research. How this information can be used will be explained in more detail in Section 2.5. A plethora of countermeasures have been suggested to again make things secure, but they tend to come with quite significant additional costs.

1.1 Problem Statement

Especially for small devices, significant additional costs in terms of e.g. area are often not tolerable. Similarly, implementing a masking countermeasure in software can slow down a computation by a factor of 2 or more [50], which is undesired. There are situations where these disadvantages lead to ignoring countermeasures and to deliberately choosing less secure alternatives. On the one hand it is therefore important to know how it is possible to assess the security of a method against such implementation- and device-dependent attacks. With this knowledge, it can be used as a design criterion for new methods that are intrinsically more secure, regardless their

[1]And they still are. Or at least, reasonably secure, up to our current state of knowledge.

implementation and on which device they are run. On the other hand it is relevant to find out how the additional costs that come with countermeasures can be reduced, so it becomes more likely that they are widely adopted.

1.2 Research Questions

This thesis assumes the context of the CAESAR candidates for authenticated encryption. CAESAR is an international multi-year competition aimed at finding new secure and efficient authenticated encryption primitives. This will be explained in more detail in Section 2.2.3. Within this context, the above directly translates to the following main research questions:

1. How can known metrics be used at the design stage to assess the intrinsic resistance of CAESAR candidates to implementation- and device-dependent attacks?
2. How can the costs of applying masking countermeasures to CAESAR candidates be reduced?

However, the research methods and conclusions are more generally applicable. There is no fundamental difference between the CAESAR candidates for authenticated encryption and other cryptographic algorithms with regard to their security against implementation- and device-dependent attacks or with regard to the costs of countermeasures. The CAESAR candidates are taken as a case study, but they are sufficiently diverse to represent a larger set of cryptographic methods.

1.3 Structure

Some preliminaries on cryptography, the CAESAR candidates and side-channel analysis are provided in Chapter 2. We then attempt to answer the first research question by looking at the metrics that exist in literature. They are covered in Chapter 3. New results show how the metrics perform in practical scenario's and a comparison between metrics is made. Chapter 4 and Chapter 5 are more related to the second research question. In Chapter 4, the number of costly nonlinear operations is reduced for several CAESAR candidates. Chapter 5 provides an overview of the expected masking costs by looking at the types of operations, together resulting in a general strategy to design ciphers with low masking costs. Chapter 6 summarises the main conclusions and Chapter 7 gives some ideas for future work.

Chapter 2

Preliminaries

2.1 Security Goals

In information security, there are some goals that are often pursued. *Confidentiality* means that access to data is limited by rules. For example, it is a matter of confidentiality when only certain individuals are allowed to read a certain message. *Data integrity* refers to that data cannot be altered by someone who is unauthorised or without him or her being detected. *Data origin authenticity*, or hereafter simply authenticity, means that someone can guarantee where data originates from and that it has not been altered after it came from this origin. It can be seen that data origin authenticity implies data integrity. *Entity authenticity* means that someone can be sure who he or she is communicating with. Finally, *availability* and *non-repudiation* deal with the availability of data when it is required and with not being able to deny one's actions, respectively.

2.2 Symmetric Cryptography

2.2.1 Encryption

Cryptography is the study of designing mathematical techniques to provide security goals, even in a context with adversaries. One use case is when one wants to send information to someone else, without others being able to read or to alter the information. Symmetric cryptography, also known as secret-key cryptography, is a particular kind of cryptography where the sender and receiver possess a shared *key*. Using this key, the sender can *encrypt* his information, or the *plaintext*, to something that should be incomprehensible to anyone else, the *ciphertext*. This process is called symmetric encryption and it deals with the security goal of confidentiality. Only with the correct key is the receiver able to *decrypt* the information to the original plaintext.

The primitives that are used to build symmetric encryption functions can be further categorised as *block ciphers*, *permutations* and *stream ciphers*. Block ciphers operate on fixed-length blocks. When used for encryption, they provide an encryption function $E(x, k) : \mathbb{F}_2^n \times \mathbb{F}_2^{|k|} \to \mathbb{F}_2^n$, where x is a plaintext of length n and k is a key of length $|k|$. The inverse of this mapping is commonly (but nut necessarily) used to build the decryption function. When the plaintext that one wants to encrypt is larger than n, the plaintext is split into multiple blocks. Stream ciphers generate a keystream of arbitrary length that usually only depends on the key[1], which is then combined with

[1]This does not hold for e.g. self-synchronizing stream encryption, where the keystream also depends on the last n ciphertext bits, for some n.

the plaintext to compute the ciphertext, often using the bitwise `XOR` ('exclusive or') operation or using bitwise addition. All these primitives are run in a certain mode of operation, that dictates how to use the primitive on arbitrary messages. Some examples of modes are Electronic Codebook (ECB), Cipher Block Chaining (CBC), Counter (CTR), and the sponge construction. A block cipher can also be used to provide stream encryption when, for instance, counter mode is used.

2.2.2 Authentication

In cryptographic protocols, secure communication also requires authenticity, to guarantee that the message that was received was indeed sent by the right sender and not by someone else, and to guarantee that the message has not been tampered with. The traditional way to provide authenticity is to add a *message authentication code* (MAC or tag) to the message. The MAC depends on a key and on the plaintext. Primitives such as block ciphers run in particular modes can be used to construct MAC functions.

2.2.3 Authenticated Encryption

There are a lot of details to get right when combining encryption and MACs. Several flaws in popular protocols such as SSL/TLS were due to this issue. Several approaches are possible, but the three best-known are encrypt-then-MAC, encrypt-and-MAC (e.g. as in SSH), and MAC-then-encrypt (e.g. as in SSL and TLS).

Because of these difficulties with combining confidentiality, integrity, and authenticity, researchers have looked at block cipher modes of operation that provide all of them at once in an efficient way. These are called authenticated encryption (AE) schemes, or authenticated encryption with associated data (AEAD). Some notable examples are GCM and CCM.

While AES-GCM is commonly used in TLS 1.2, there are not many alternatives available that have been extensively studied and that could be used as a drop-in replacement in case anyone will break AES-GCM in the future. For this reason, among others, the CAESAR competition was announced in 2013. Similarly to the previously held public competitions for block ciphers (AES), stream ciphers (eSTREAM) and hash functions (SHA-3), a multi-year multi-round competition would be the solution to find more secure authenticated encryption schemes, preferably more efficient than AES-GCM.

2.3 Boolean functions and S-boxes

Constructing block ciphers can be done in a lot of different ways. Section 2.1.2 of [59] summarises some features to distinguish block ciphers. One important aspect is that ciphers have to contain a nonlinear part in order to withstand cryptanalysis. Often, this is provided by a so-called S-box, which is a nonlinear function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$. Many properties are defined for S-boxes that relate to their theoretical security [11]. Only a handful is explained in the next section.

2.3.1 Properties

An S-box is called a *square* S-box when $n = m$. A *fixed point* of S is a value x such that $S(x) = x$.

When $m = 1$, the function is called a *Boolean function*. In general, the S-box can be thought of as being composed of Boolean *coordinate functions* $S_i : \mathbb{F}_2^n \to \mathbb{F}_2$ for all $i \in \{0, \dots, m-1\}$.

A Boolean function f is called *balanced* when $\Pr[f(\cdot) = 1] = \frac{1}{2}$. An S-box is called balanced when all coordinate functions are balanced. When an S-box takes every value of $\mathbb{F}_2^m$ the same number 2^{n-m} of times, it is implied that an S-box is balanced. Balanced S-boxes with $n = m$ are permutations on $\mathbb{F}_2^n$.

The *Walsh transform* W_S of an S-box S is the function

$$W_S(a, v) = \sum_{x \in \mathbb{F}_2^n} (-1)^{(v \cdot S(x)) \oplus (a \cdot x)},$$

where $\oplus$ denotes addition modulo 2 and $b \cdot c = \bigoplus_{i=0}^{n-1} b_i c_i$ is the dot product of two vectors b and c.

A Boolean function f is called *bent* when its Walsh transform has a constant absolute value [63]. Bent functions exist only for even n.

Let the Hamming weight HW of $x \in \mathbb{F}_2^n$ be defined as:

$$\mathrm{HW}(x) = \sum_{i=0}^{n-1} x_i,$$

where x_i is the i^{th} bit of x. Informally, the Hamming weight corresponds to the number of non-zero positions in the bit string representation of x. The Hamming distance HD between two vectors $a, b \in \mathbb{F}_2^n$ is $\mathrm{HD}(a, b) = \mathrm{HW}(a \oplus b)$, where $\oplus$ denotes the bitwise `XOR` operation. When a Boolean function f is bent, $\mathrm{HW}(f) = 2^{n-1} \pm 2^{\frac{n}{2}-1}$ [16].

Let $\mathbb{F}_2^{m*}$ be defined as $\mathbb{F}_2^{m*} = \mathbb{F}_2^m \setminus (\underbrace{0, \ldots, 0}_{m})$. Then the *linearity* of an S-box S is [11]:

$$\mathrm{Lin}(S) = \max_{a \in \mathbb{F}_2^n, v \in \mathbb{F}_2^{m*}} |W_S(a, v)|.$$

The *nonlinearity* of S is its distance to the nearest affine function on $\mathbb{F}_2^n$ [18], i.e.:

$$\mathrm{Nonlin}(S) = 2^{n-1} - \frac{1}{2}\mathrm{Lin}(S).$$

When a Boolean function f is bent, $\mathrm{Nonlin}(f) = 2^{n-1} - 2^{\frac{n}{2}-1}$, which is the maximum possible nonlinearity [57].

The *algebraic degree* of a Boolean function f, $\deg(f)$, is the maximum number of variables of the terms in the algebraic normal form (ANF) of f [11]. f is *affine* when $\deg(f) \leq 1$. f is *linear* when f is affine, non-constant and when $f(0) = 0$. When f is bent and $n > 2$, $\deg(f) \leq \frac{n}{2}$ [63]. The algebraic degree of an S-box S is:

$$\deg(S) = \max_{i \in \{0, \ldots, n-1\}} \deg(S_i).$$

The *differential delta uniformity* δ of an S-box is the largest value in the difference distribution table excluding the 2^n in the upper left position [8]. It holds for all S-boxes that $\delta(S) \geq 2$.

2.3.2 Desired Values

A lot of research has aimed at finding systematic methods for designing S-boxes with good cryptographic properties, but especially for larger S-boxes, this still remains a largely unsolved problem. For cryptographic applications, S-boxes with linear coordinate functions are undesired. Usually, it is desired to be invertible and to have a nonlinearity an algebraic degree as high as possible, while δ should be as low as possible, although this depends on the design strategy and does not hold in general. Recent results argue that this last criterion should in fact always be relaxed [40]. A high nonlinearity implies that an S-box is very suitable to be part of a cipher that is very resistant to linear cryptanalysis [48]. A high algebraic degree means that algebraic cryptanalysis [4] is harder and a low differential delta uniformity means that differential cryptanalysis [8] will be harder.

For balanced S-boxes, the algebraic degree can be at most $n-1$. The S-boxes with the highest nonlinearity are bent, however, and that implies an algebraic degree of $\leq \frac{n}{2}$ [63]. So-called *almost bent* functions are related to bent functions but still have good nonlinearity properties [19].

For 4x4 S-boxes (i.e. $n = 4, m = 4$), it is known which S-boxes are 'optimal', where being optimal is defined as being bijective, having $\mathrm{Lin}(S) = 8$[2], and $\delta(S) = 4$. Whether this 'optimal' result is also really desired depends on the design strategy. There exist no 4x4 S-boxes with $\delta(S) = 2$ [46]. All 16! possible bijective S-boxes have exhaustively been classified. It turns out that up to affine equivalence, there exist only 16 different optimal S-boxes [46]. Bijectivity, Lin and δ are invariant under invertible affine transformations, so when S is an optimal S-box, $S'(x) = B(S(A(x)+a)) + b$ is an optimal S-box as well, for $A, B \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$, A and B invertible, $a, b \in \mathbb{F}_2^4$, and where $A(x)$ denotes multiplication of column vector x with square matrix A. 8 of these 16 have $\forall_{i \in \{0,\ldots,n-1\}} \deg(S_i) = 3$.

All of this is not known for larger S-boxes, as exhaustive searching is then no longer feasible. Attempts have been made to use evolutionary computation to find 'better' 8x8 S-boxes [61].

2.4 S-boxes of CAESAR Candidates and RECTANGLE

The CAESAR competition for new authenticated encryption schemes started out with 57 submissions for the first round. Nine of them have been broken or withdrawn before the end of the first round. In July 2015 it was announced which 30 candidates have made it to the second round of the competition.

An overview and categorisation of 50 candidates, which includes cryptographic features, security claims and parameter sizes, can be found in [1]. Most candidates are either based on a block cipher as central building block, a stream cipher, or a permutation. Apart from stream cipher based approaches and some dedicated methods, most candidates contain an S-box as the only nonlinear element of the algorithm. This section takes a closer look at the S-boxes that are involved in the competition. For all S-boxes it holds that they are bijective. The properties are computed using the SET toolbox [60].

[2]Or, equivalently, having $\mathrm{Nonlin}(S) = 4$.

2.4.1 AES

$$S = \begin{bmatrix}
99, & 124, & 119, & 123, & 242, & 107, & 111, & 197, & 48, & 1, & 103, & 43, & 254, & 215, & 171, & 118 \\
202, & 130, & 201, & 125, & 250, & 89, & 71, & 240, & 173, & 212, & 162, & 175, & 156, & 164, & 114, & 192 \\
183, & 253, & 147, & 38, & 54, & 63, & 247, & 204, & 52, & 165, & 229, & 241, & 113, & 216, & 49, & 21 \\
4, & 199, & 35, & 195, & 24, & 150, & 5, & 154, & 7, & 18, & 128, & 226, & 235, & 39, & 178, & 117 \\
9, & 131, & 44, & 26, & 27, & 110, & 90, & 160, & 82, & 59, & 214, & 179, & 41, & 227, & 47, & 132 \\
83, & 209, & 0, & 237, & 32, & 252, & 177, & 91, & 106, & 203, & 190, & 57, & 74, & 76, & 88, & 207 \\
208, & 239, & 170, & 251, & 67, & 77, & 51, & 133, & 69, & 249, & 2, & 127, & 80, & 60, & 159, & 168 \\
81, & 163, & 64, & 143, & 146, & 157, & 56, & 245, & 188, & 182, & 218, & 33, & 16, & 255, & 243, & 210 \\
205, & 12, & 19, & 236, & 95, & 151, & 68, & 23, & 196, & 167, & 126, & 61, & 100, & 93, & 25, & 115 \\
96, & 129, & 79, & 220, & 34, & 42, & 144, & 136, & 70, & 238, & 184, & 20, & 222, & 94, & 11, & 219 \\
224, & 50, & 58, & 10, & 73, & 6, & 36, & 92, & 194, & 211, & 172, & 98, & 145, & 149, & 228, & 121 \\
231, & 200, & 55, & 109, & 141, & 213, & 78, & 169, & 108, & 86, & 244, & 234, & 101, & 122, & 174, & 8 \\
186, & 120, & 37, & 46, & 28, & 166, & 180, & 198, & 232, & 221, & 116, & 31, & 75, & 189, & 139, & 138 \\
112, & 62, & 181, & 102, & 72, & 3, & 246, & 14, & 97, & 53, & 87, & 185, & 134, & 193, & 29, & 158 \\
225, & 248, & 152, & 17, & 105, & 217, & 142, & 148, & 155, & 30, & 135, & 233, & 206, & 85, & 40, & 223 \\
140, & 161, & 137, & 13, & 191, & 230, & 66, & 104, & 65, & 153, & 45, & 15, & 176, & 84, & 187, & 22
\end{bmatrix}$$

$$S^{-1} = \begin{bmatrix}
82, & 9, & 106, & 213, & 48, & 54, & 165, & 56, & 191, & 64, & 163, & 158, & 129, & 243, & 215, & 251 \\
124, & 227, & 57, & 130, & 155, & 47, & 255, & 135, & 52, & 142, & 67, & 68, & 196, & 222, & 233, & 203 \\
84, & 123, & 148, & 50, & 166, & 194, & 35, & 61, & 238, & 76, & 149, & 11, & 66, & 250, & 195, & 78 \\
8, & 46, & 161, & 102, & 40, & 217, & 36, & 178, & 118, & 91, & 162, & 73, & 109, & 139, & 209, & 37 \\
114, & 248, & 246, & 100, & 134, & 104, & 152, & 22, & 212, & 164, & 92, & 204, & 93, & 101, & 182, & 146 \\
108, & 112, & 72, & 80, & 253, & 237, & 185, & 218, & 94, & 21, & 70, & 87, & 167, & 141, & 157, & 132 \\
144, & 216, & 171, & 0, & 140, & 188, & 211, & 10, & 247, & 228, & 88, & 5, & 184, & 179, & 69, & 6 \\
208, & 44, & 30, & 143, & 202, & 63, & 15, & 2, & 193, & 175, & 189, & 3, & 1, & 19, & 138, & 107 \\
58, & 145, & 17, & 65, & 79, & 103, & 220, & 234, & 151, & 242, & 207, & 206, & 240, & 180, & 230, & 115 \\
150, & 172, & 116, & 34, & 231, & 173, & 53, & 133, & 226, & 249, & 55, & 232, & 28, & 117, & 223, & 110 \\
71, & 241, & 26, & 113, & 29, & 41, & 197, & 137, & 111, & 183, & 98, & 14, & 170, & 24, & 190, & 27 \\
252, & 86, & 62, & 75, & 198, & 210, & 121, & 32, & 154, & 219, & 192, & 254, & 120, & 205, & 90, & 244 \\
31, & 221, & 168, & 51, & 136, & 7, & 199, & 49, & 177, & 18, & 16, & 89, & 39, & 128, & 236, & 95 \\
96, & 81, & 127, & 169, & 25, & 181, & 74, & 13, & 45, & 229, & 122, & 159, & 147, & 201, & 156, & 239 \\
160, & 224, & 59, & 77, & 174, & 42, & 245, & 176, & 200, & 235, & 187, & 60, & 131, & 83, & 153, & 97 \\
23, & 43, & 4, & 126, & 186, & 119, & 214, & 38, & 225, & 105, & 20, & 99, & 85, & 33, & 12, & 125
\end{bmatrix}$$

Rijndael [25] was the name of the winner of the NIST Advanced Encryption Standard competition for block ciphers, and since it has been standardised in 2001 it has became the block cipher most commonly used in practice. It has been given much attention from researchers. For this reason, many of the CAESAR candidates use AES as the underlying block cipher. Another reason is that newer processors that support the AES-NI instruction set extension have AES built into hardware, which nearly guarantees high encryption and decryption speeds when AES is run in parallelisable modes.

AES contains an 8x8 S-box that was designed to minimise the maximum differential propagation probability. $\deg(S) = \deg(S^{-1}) = 7$ and $\mathrm{Nonlin}(S) = \mathrm{Nonlin}(S^{-1}) = 112$. No 8x8 S-boxes are known with the same algebraic degree and a higher nonlinearity, although it has been proven that one should exist [64]. For both S and S', $\delta = 4$.

2.4.2 iSCREAM and SCREAM

$$S = \begin{bmatrix}
0, & 133, & 101, & 210, & 91, & 255, & 122, & 206, & 77, & 226, & 44, & 54, & 146, & 21, & 189, & 173 \\
87, & 243, & 55, & 45, & 136, & 13, & 172, & 188, & 24, & 159, & 126, & 202, & 65, & 238, & 97, & 214 \\
89, & 236, & 120, & 212, & 71, & 249, & 38, & 163, & 144, & 139, & 191, & 48, & 10, & 19, & 111, & 192 \\
43, & 174, & 145, & 138, & 216, & 116, & 11, & 18, & 204, & 99, & 253, & 67, & 178, & 61, & 232, & 93 \\
182, & 28, & 131, & 59, & 200, & 69, & 157, & 36, & 82, & 221, & 228, & 244, & 171, & 8, & 119, & 109 \\
245, & 229, & 72, & 197, & 108, & 118, & 186, & 16, & 153, & 32, & 167, & 4, & 135, & 63, & 208, & 95 \\
165, & 30, & 155, & 57, & 176, & 2, & 234, & 103, & 198, & 223, & 113, & 246, & 84, & 79, & 141, & 46 \\
231, & 106, & 199, & 222, & 53, & 151, & 85, & 78, & 34, & 129, & 6, & 180, & 124, & 251, & 26, & 161 \\
213, & 121, & 252, & 66, & 132, & 1, & 233, & 92, & 20, & 147, & 51, & 41, & 193, & 110, & 168, & 184 \\
40, & 50, & 12, & 137, & 185, & 169, & 217, & 117, & 237, & 88, & 205, & 98, & 248, & 70, & 158, & 25 \\
203, & 127, & 162, & 39, & 215, & 96, & 254, & 90, & 142, & 149, & 227, & 76, & 22, & 15, & 49, & 190 \\
100, & 211, & 60, & 179, & 123, & 207, & 64, & 239, & 143, & 148, & 86, & 242, & 23, & 14, & 175, & 42 \\
47, & 140, & 241, & 225, & 220, & 83, & 104, & 114, & 68, & 201, & 27, & 160, & 56, & 154, & 7, & 181 \\
94, & 209, & 3, & 177, & 35, & 128, & 31, & 164, & 52, & 150, & 224, & 240, & 196, & 73, & 115, & 105 \\
218, & 195, & 9, & 170, & 74, & 81, & 247, & 112, & 62, & 134, & 102, & 235, & 33, & 152, & 29, & 183 \\
219, & 194, & 187, & 17, & 75, & 80, & 107, & 230, & 156, & 37, & 250, & 125, & 130, & 58, & 166, & 5
\end{bmatrix}$$

$$S' = \begin{bmatrix}
30, & 117, & 95, & 225, & 153, & 252, & 137, & 47, & 134, & 238, & 241, & 123, & 35, & 82, & 16, & 148 \\
12, & 183, & 77, & 103, & 216, & 66, & 200, & 214, & 196, & 107, & 170, & 186, & 61, & 165, & 0, & 51 \\
83, & 45, & 11, & 184, & 218, & 168, & 197, & 108, & 202, & 182, & 164, & 34, & 96, & 7, & 93, & 215 \\
79, & 244, & 21, & 50, & 129, & 27, & 156, & 142, & 145, & 63, & 230, & 249, & 112, & 233, & 67, & 126 \\
141, & 243, & 204, & 101, & 8, & 122, & 24, & 171, & 22, & 106, & 119, & 253, & 167, & 192, & 130, & 4 \\
159, & 49, & 222, & 227, & 73, & 208, & 89, & 70, & 84, & 239, & 46, & 60, & 187, & 33, & 146, & 181 \\
85, & 62, & 15, & 169, & 220, & 185, & 193, & 127, & 206, & 166, & 180, & 48, & 114, & 3, & 91, & 209 \\
75, & 228, & 19, & 32, & 133, & 29, & 154, & 138, & 151, & 44, & 246, & 232, & 98, & 248, & 71, & 109 \\
41, & 65, & 104, & 213, & 172, & 203, & 190, & 26, & 176, & 219, & 199, & 78, & 23, & 100, & 38, & 160 \\
57, & 131, & 120, & 81, & 237, & 118, & 255, & 226, & 242, & 92, & 157, & 143, & 10, & 147, & 52, & 5 \\
37, & 88, & 124, & 205, & 175, & 223, & 179, & 25, & 189, & 194, & 210, & 86, & 20, & 113, & 42, & 163 \\
58, & 128, & 97, & 68, & 245, & 110, & 235, & 251, & 231, & 72, & 144, & 140, & 6, & 158, & 55, & 9 \\
152, & 229, & 217, & 115, & 31, & 111, & 13, & 188, & 2, & 125, & 99, & 234, & 177, & 212, & 150, & 18 \\
136, & 39, & 201, & 247, & 94, & 198, & 76, & 80, & 64, & 250, & 59, & 43, & 174, & 53, & 132, & 161 \\
1, & 105, & 90, & 254, & 139, & 236, & 149, & 40, & 155, & 240, & 224, & 102, & 36, & 87, & 14, & 135 \\
28, & 178, & 69, & 116, & 211, & 74, & 207, & 221, & 195, & 121, & 162, & 191, & 54, & 173, & 17, & 56
\end{bmatrix}$$

$$S'^{-1} = \begin{bmatrix}
30, & 224, & 200, & 109, & 79, & 159, & 188, & 45, & 68, & 191, & 156, & 34, & 16, & 198, & 238, & 98 \\
14, & 254, & 207, & 114, & 172, & 50, & 72, & 140, & 70, & 167, & 135, & 53, & 240, & 117, & 0, & 196 \\
115, & 93, & 43, & 12, & 236, & 160, & 142, & 209, & 231, & 128, & 174, & 219, & 121, & 33, & 90, & 7 \\
107, & 81, & 51, & 31, & 158, & 221, & 252, & 190, & 255, & 144, & 176, & 218, & 91, & 28, & 97, & 57 \\
216, & 129, & 21, & 62, & 179, & 242, & 87, & 126, & 185, & 84, & 245, & 112, & 214, & 18, & 139, & 48 \\
215, & 147, & 13, & 32, & 88, & 96, & 171, & 237, & 161, & 86, & 226, & 110, & 153, & 46, & 212, & 2 \\
44, & 178, & 124, & 202, & 141, & 67, & 235, & 19, & 130, & 225, & 73, & 25, & 39, & 127, & 181, & 197 \\
60, & 173, & 108, & 195, & 243, & 1, & 149, & 74, & 146, & 249, & 69, & 11, & 162, & 201, & 63, & 103 \\
177, & 52, & 78, & 145, & 222, & 116, & 8, & 239, & 208, & 6, & 119, & 228, & 187, & 64, & 55, & 155 \\
186, & 56, & 94, & 157, & 15, & 230, & 206, & 120, & 192, & 4, & 118, & 232, & 54, & 154, & 189, & 80 \\
143, & 223, & 250, & 175, & 42, & 29, & 105, & 76, & 37, & 99, & 26, & 71, & 132, & 253, & 220, & 164 \\
136, & 204, & 241, & 166, & 106, & 95, & 41, & 17, & 35, & 101, & 27, & 92, & 199, & 168, & 134, & 251 \\
77, & 102, & 169, & 248, & 24, & 38, & 213, & 138, & 22, & 210, & 40, & 133, & 66, & 163, & 104, & 246 \\
85, & 111, & 170, & 244, & 205, & 131, & 23, & 47, & 20, & 194, & 36, & 137, & 100, & 247, & 82, & 165 \\
234, & 3, & 151, & 83, & 113, & 193, & 58, & 184, & 123, & 61, & 203, & 182, & 229, & 148, & 9, & 89 \\
233, & 10, & 152, & 65, & 49, & 180, & 122, & 211, & 125, & 59, & 217, & 183, & 5, & 75, & 227, & 150
\end{bmatrix}$$

iSCREAM and SCREAM are based on the new tweakable block ciphers iScream and Scream, which only have some small differences [37]. Both use 8x8 S-boxes that are chosen because of their known highly efficient bitsliced implementations. The S-boxes are based on the ones of Robin and Fantomas, respectively, as introduced in [36]. The S-boxes are created by combining two efficient 4-bit S-boxes. An important difference is that the S-box of iScream and Robin is involutory, i.e. $S = S^{-1}$ or $S(S(x)) = x$, which means that it saves space by not requiring an inverse for the inverse cipher. This does not hold for the S-boxes of Scream and Fantomas. All three S-boxes have Nonlin $= 96$ and $\delta = 16$. $\deg(S) = \deg(S'^{-1}) = 6$, while $\deg(S') = 5$. However, SCREAM has made it to the second round of the CAESAR competition, while iSCREAM has not.

2.4.3 Ascon

$$\begin{aligned} S = [&\mathtt{4, 11, 31, 20, 26, 21, 9, 2, 27, 5, 8, 18, 29, 3, 6, 28}, \dots \\ &\dots, \mathtt{30, 19, 7, 14, 0, 13, 17, 24, 16, 12, 1, 25, 22, 10, 15, 23}] \end{aligned}$$

The 5-bit S-box used in Ascon [28] is a lightweight affine transformation of Keccak-χ [5]. When the transformation is seen as part of the mixing layer, the S-box is really the same as in Keccak. It is designed to have an efficient bitsliced software implementation. With $\deg(S) = 2$ it has the lowest possible algebraic degree for a non-linear function, $\text{Nonlin}(S) = 8$, and $\delta(S) = 8$. While the S-box is not an involution, the decryption in Ascon does not require the inverse S-box due to its sponge construction.

2.4.4 ICEPOLE

$$\begin{aligned} S = [&\mathtt{31, 5, 10, 11, 20, 17, 22, 23, 9, 12, 3, 2, 13, 8, 15, 14}, \dots \\ &\dots, \mathtt{18, 21, 24, 27, 6, 1, 4, 7, 26, 29, 16, 19, 30, 25, 28, 0}] \end{aligned}$$

ICEPOLE's 5-bit S-box [52, 53] is also based on Keccak-χ, but it has swapped the values 0 and 31 in order to achieve a higher algebraic degree. This comes at the cost of more gates or more bitwise operations. $\deg(S) = 4$, $\text{Nonlin}(S) = 8$, and $\delta(S) = 8$.

2.4.5 Ketje and Keyak

$$\begin{aligned} S = [&\mathtt{0, 5, 10, 11, 20, 17, 22, 23, 9, 12, 3, 2, 13, 8, 15, 14}, \dots \\ &\dots, \mathtt{18, 21, 24, 27, 6, 1, 4, 7, 26, 29, 16, 19, 30, 25, 28, 31}] \end{aligned}$$

The nonlinear element in both Ketje and Keyak [6, 7] is the same as in the hash function Keccak [5], which was chosen as SHA-3 in 2012. This element is the Keccak-χ step and operates on 5-bit inputs. It has a very efficient hardware implementation. In pseudocode, it can be specified as $A[x, y] = a[x, y] \oplus ((\texttt{NOT } a[x + 1, y]) \texttt{ AND } a[x + 2, y])$ for all $0 \leq x, y < 5$. $\deg(S) = 2$, $\text{Nonlin}(S) = 8$, and $\delta(S) = 8$.

2.4.6 PRIMATE

$$\begin{aligned} S = [&\mathtt{1, 0, 25, 26, 17, 29, 21, 27, 20, 5, 4, 23, 14, 18, 2, 28}, \dots \\ &\dots, \mathtt{15, 8, 6, 3, 13, 7, 24, 16, 30, 9, 31, 10, 22, 12, 11, 19}] \\ S^{-1} = [&\mathtt{1, 0, 14, 19, 10, 9, 18, 21, 17, 25, 27, 30, 29, 20, 12, 16}, \dots \\ &\dots, \mathtt{23, 4, 13, 31, 8, 6, 28, 11, 22, 2, 3, 7, 15, 5, 24, 26}] \end{aligned}$$

With PRIMATE, a new 5-bit S-box was introduced [2]. The S-box is an almost bent function and reaches a relatively high nonlinearity of $\mathrm{Nonlin}(S) = 12$. $\deg(S) = 2$, but $\deg(S^{-1}) = 3$. $\delta(S) = \delta(S^{-1}) = 2$, which is optimal.

2.4.7 Joltik

$$S = [14, 4, 11, 2, 3, 8, 0, 9, 1, 10, 7, 15, 6, 12, 5, 13]$$
$$S^{-1} = [6, 8, 3, 4, 1, 14, 12, 10, 5, 7, 9, 2, 13, 15, 0, 11]$$

Joltik [42] uses the same 4-bit S-box as the lightweight block cipher Piccolo, which is designed to have a very small hardware implementation. The S-box has the optimal nonlinearity, the optimal algebraic degree, and the optimal differential delta uniformity, i.e. $\mathrm{Nonlin}(S) = \mathrm{Nonlin}(S^{-1}) = 4$, $\deg(S) = \deg(S^{-1}) = 3$, and $\delta(S) = \delta(S^{-1}) = 4$.

2.4.8 LAC

$$S = [14, 9, 15, 0, 13, 4, 10, 11, 1, 2, 8, 3, 7, 6, 12, 5]$$

The 4-bit S-box of LAC [72] has the same cryptographic properties as Joltik's. However, it is based on one of the S-boxes of the LBlock, another lightweight block cipher. LAC does not require the inverse S-box for its decryption operation. LAC has not made it to the second round of the CAESAR competition.

2.4.9 Minalpher

$$S = [11, 3, 4, 1, 2, 8, 12, 15, 5, 13, 14, 0, 6, 9, 10, 7]$$

Minalpher's S-box [66] is another 4x4 S-box with the same nonlinearity, algebraic degree, and δ as the S-boxes of Joltik and LAC. Noteworthy is that it is an involution in order to save space with the decryption operation.

2.4.10 Prøst

$$S = [0, 4, 8, 15, 1, 5, 14, 9, 2, 7, 10, 12, 11, 13, 6, 3]$$

The authors of Prøst [43] have chosen another involutory S-box, also with $\mathrm{Nonlin}(S) = 4$, $\deg(S) = 3$, and $\delta(S) = 4$. It is designed to have a very small bitsliced implementation. Still, it has not made it to the second round of the CAESAR competition.

2.4.11 RECTANGLE

$$S = [9, 4, 15, 10, 14, 1, 0, 6, 12, 7, 3, 8, 2, 11, 5, 13]$$
$$S^{-1} = [6, 5, 12, 10, 1, 14, 7, 9, 11, 0, 3, 13, 8, 15, 4, 2]$$

RECTANGLE is not part of the CAESAR competition, but it is a recently proposed lightweight block cipher [73]. It is included because it introduces new design criteria for S-boxes. It contains a 4-bit S-box that not only has the same optimal nonlinearity, algebraic degree, and δ, but also satisfies the following criteria. Let $\mathbb{F}_2^{4*} = \mathbb{F}_2^4 \setminus (0, 0, 0, 0)$. Then the following holds:

- There exist only 2 pairs $(x, y) \in \mathbb{F}_2^{4*} \times \mathbb{F}_2^{4*}$ such that $\mathrm{HW}(x) = \mathrm{HW}(y) = 1$ and $\#\{z \in \mathbb{F}_2^4 | S(z) \oplus S(z \oplus x)) = z\} \neq 0$.
- There exist only 2 pairs $(x, y) \in \mathbb{F}_2^{4*} \times \mathbb{F}_2^{4*}$ such that $\mathrm{HW}(x) = \mathrm{HW}(y) = 1$ and $\left|\#\{z \in \mathbb{F}_2^4 | x \cdot z = y \cdot S(z)\} - 8\right| \neq 0$.

Additionally, RECTANGLE's S-box is chosen to be small in hardware.

2.4.12 Overview

Table 2.1 shows a summary of some cryptographic properties of the S-boxes given in Section 2.4. In this table, '1 enc/dec' means that there are one encryption and one decryption S-box that are used by the authenticated encryption scheme. '1 inv' denotes an involution S-box and '1 enc' denotes that only one S-box is used by both encryption and decryption, although it is not an involution. An algebraic degree of a/b means that $\deg(S) = a$ and $\deg(S^{-1}) = b$.

S-box	Number	Width	Nonlin(S)	deg(S)	$\delta(S)$
AES	1 enc/dec	8	112	7	4
iSCREAM	1 inv	8	96	6	16
SCREAM	1 enc/dec	8	96	5/6	16
Ascon	1 enc	5	8	2	8
ICEPOLE	1 enc	5	8	4	8
Ketje/Keyak	1 enc	5	8	2	8
PRIMATE	1 enc/dec	5	12	2/3	2
Joltik	1 enc/dec	4	4	3	4
LAC	1 enc	4	4	3	4
Minalpher	1 inv	4	4	3	4
Prøst	1 inv	4	4	3	4
RECTANGLE	1 enc/dec	4	4	3	4

Table 2.1: Cryptographic properties of S-boxes w.r.t. linear and differential cryptanalysis attacks.

2.5 Side-Channel Analysis

Over the years, a lot of research has been done on linear and differential cryptanalysis, and on methods to design ciphers that are robust against those kinds of attacks. However, it has turned out that even ciphers that are secure against known attacks, can be broken in the physical reality, due to the emittance of information that is inherent to the physical world. This realisation has raised the field of side-channel analysis (SCA), where a side channel is a physically measurable quantity. It started with timing attacks [45], and has since been extended to, for instance, power analysis [44], SCA using electromagnetic radiance [32], optical properties [31], and acoustic cryptanalysis [33]. This section discusses power analysis in more detail, as this attack has proven to be very powerful and to be relatively easy to mount. In addition, several variants exist.

2.5.1 Power Analysis Attacks

Differential power analysis (DPA) was introduced in [44] and relies on the fact that the power consumption of a device performing an operation is related both to that operation and to the data the operation is performed on. For instance, when switching the value of a bit in a register from 1 to 0 or from 0 to 1 consumes power, an operation that has to switch more values will most likely consume more power. The power consumed by the data is the dominant factor here. An attacker can therefore measure the power consumption of a device and learn information about the potentially secret data in the device.

Assume that a device performs a keyed cryptographic operation on a known part, e.g. encryption of a chosen plaintext or decryption of a chosen ciphertext, and the goal is to extract the correct key. Without loss of generality, it is assumed that the device computes the encryption $E(x, k)$ of a known x and the attacker wants to find k. We will assume that E consists of multiple rounds that each process a part of the key k_j, called a subkey, for which exhaustive search is feasible. This holds for all commonly used block ciphers. In more detail, DPA works as follows [44].

First, collect N *power traces* $p_i(t)$, i.e. the power consumption traces of the device computing $E(x_i, k)$ for N different x_i at time t. Then decide on a selection function $V(x_i, k_j)$ that depends on the plaintext x_i and the subkey k_j and that selects a (bit of a) sensitive intermediate value that leaks information. One example is the least significant bit of the output of the S-box in an AES-like block cipher, i.e. $V(x_i, k_j) = \text{LSB}(S(x_{i,j} \oplus k_j))$, where $x_{i,j}$ is the j^{th} part of the i^{th} plaintext. We start by bruteforcing k_0. For all possible subkey guesses k', compute $V(x_i, k')$. If this yields 0, put the corresponding power trace $p_i(t)$ in the set **Mean0**. Otherwise, put it in **Mean1**. After having split all N traces, compute the differential trace $\Delta(t)$ as the difference between the average of the traces in **Mean0** and the average of the traces in **Mean1**. This is called the *distance-of-means* (DoM) distinguisher.

When $k' \neq k_0$, the output of $V(x_i, k')$ will be 1 for about half the plaintexts x_i, which implies that $\lim_{N\to\infty} \Delta(t) = 0$. However, when $k' = k_0$, $\Delta(t)$ will show spikes in the regions where the LSB is correlated to the values that are processed. The subkey guess k' for which the differential is maximal will be the correct subkey k_0. To get the full key k, this whole process can be repeated for the other subkeys k_j.

2.5.2 Variants

The above selection function and distinguisher were introduced in the original paper of Kocher et al., but other choices can be made as well. One variant that is frequently used is often called correlation power analysis (CPA) [15]. Instead of using one bit as selection function, a more realistic power consumption model is introduced. The idea is that power consumption is only indirectly related to the data that is being processed, but this can be estimated better by taking into account the type of device.

For instance, for register outputs in ASICs, only the number of transitions from 0 to 1 and from 1 to 0 are relevant. All bits contribute equally to the power consumption, both kinds of transitions consume the same amount and static power can be ignored. This is called the Hamming distance model and typically applies to hardware implementations. More precisely, let r denote the previous register value. Then $V(x_i, k') = \text{HD}(S(x_i \oplus k'), r)$.

For software implementations of cryptographic operations including smartcard implementations, the bus usually leaks and the Hamming weight model, $V(x_i, k') = \text{HW}(S(x_i \oplus k'))$, is therefore more suitable. This assumes that the power consumption is proportional to the Hamming weight of the data, thus ignoring data that was processed before. This is typical for pre-charged buses.

Another difference is that CPA uses Pearson's correlation coefficient as a distinguisher instead of the DoM. Let X and Y denote variables, let $\mathbb{E}$ denote the expectation value, μ_X the mean of X, and σ_X the standard deviation of X. Then Pearson's correlation coefficient ρ can be computed as

$$\rho(X,Y) = \frac{\mathbb{E}((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y}.$$

It can be used to measure the linear correlation between the real output of a sensitive intermediate value and the hypothetical output computed with a power consumption model.

More complicated power models are also possible, such as weighted models when not all bits contribute equally, signed models when a 0-to-1 transition consumes more or less power than a 1-to-0 transition, and dedicated models for combinational circuits. For the distinguisher, other statistical tests can also be used, such as Student's t-test, principal component analysis [71], and mutual information analysis [34].

2.5.3 Masking

Countermeasures for power analysis attacks aim to remove the correlation between power consumption and intermediate data. This is done by either hiding the correlation or by masking it. Hiding means that the correlation is reduced by treating the power consumption. Examples of hiding techniques in hardware are pre-charged dual rail logic, power signal filtering, desynchronisation, and noise generation. In software, time randomisation can be used. Such techniques come at a cost of more expensive logic, a bigger total power consumption which can especially be problematic to small devices, or a slower computation.

Masking means that the correlation is reduced by transforming the intermediate data [22, 50]. Masking also comes at a cost of a slower computation or a larger area consumption. A random mask conceals the secret data at the start of the cryptographic operation and all intermediate values are computed using the concealed data. Only at the last step can the random mask be removed to still produce the correct output. How to mask a certain operation depends on the operation. For table lookups, a masked table needs to be computed.

Boolean Masking

Boolean masking means that the secret data x is masked by $x' = x \oplus r_x$, where r_x is a random mask. Bitwise Boolean functions can be masked in this way. This is trivial for the `XOR` operation, where one can simply `XOR` the masked data, $z' = x' \oplus y'$, and the masks, $r_z = r_x \oplus r_y$. For the `AND` operation, $z' = x' \wedge y'$ is easy to compute, but $r_z = (r_x \wedge y') \oplus (r_y \wedge x') \oplus (r_x \wedge r_y)$ requires some more computations. The `AND` operation is therefore, like `OR`, more expensive to mask.

Linear transformations, bitwise permutations, and fixed rotate or shift operations can also be masked by Boolean masking. The operation has to be performed on both the masked data and the mask itself. Data-dependent rotations are more complicated, as the rotation amount needs to be masked by an arithmetic mask.

Arithmetic Masking

Arithmetic masking means that the secret data x is masked by $x' = (x - r_x) \bmod 2^k$, where r is again a random mask. This is more applicable to arithmetic operations such as addition and multiplication modulo 2^k. Masking an addition is straightforward, but multiplication is more difficult, as one has to compute $z' = x'y' \bmod 2^k$ and $r_z = (r_x y' + r_y x' + r_x r_y) \bmod 2^k$. This is analogous to the `AND` operation in the case that $k = 1$.

When masking a cipher, it is sometimes necessary to switch between Boolean and arithmetic masks and vice versa. Several algorithms have been proposed, but some turned out to be insecure or are very inefficient. An overview of some methods can be found in [26].

Masking will protect against first-order DPA attacks, but higher-order attacks that combine multiple samples from within a power consumption trace are still possible and have been quite successful [44, 49]. More than one mask can be added to protect against second-order attacks. In general, an n^{th}-order masking scheme will protect against an n^{th}-order DPA attack (and lower), but can be broken by an $(n+1)^{\text{th}}$-order attack (or higher).

Low-Entropy Masking

As explained before, masking comes at a substantial cost. As often, in practice, a trade-off can be made between efficiency and security. Masking schemes have been proposed that reduce the size of the mask alphabet while retaining security against SCA attacks to a certain extent. Such masking schemes are called low-entropy masking schemes (LEMS). It has been shown that instead of the usual full entropy 256 possible masks for the AES S-box, only 12 values are sufficient to protect against first- and second-order DPA [56].

Chapter 3

Metrics for Side-Channel Leakage Resistance

Although countermeasures for SCA attacks exist, they do not fully prevent all information leakage and come with substantial costs. It is therefore interesting to be able to design a cipher in such a way that it will inherently be as resistant to SCA attacks as possible. This chapter discusses several metrics in chronological order of appearance that have been defined to measure the side-channel leakage or the resistance to SCA attacks.

3.1 Number of Measurements

Two hardware countermeasures against DPA are introducing noise, i.e. reducing the Signal-to-Noise Ratio (SNR), and randomly disarranging the time secret data is being processed. The SNR is here defined as $\mathrm{SNR} = \frac{Var(Q)}{Var(A)}$, where Q is power consumption caused by the computation and A is additive noise. For both countermeasures, its effect can be determined at the design time, even before the implementation has started. This has its benefits, because having a lot of design iterations is costly. There are other factors that are less predictable at the design time, but setting those to the worst case will allow one to compute a lower bound for the number of measurements required to successfully perform DPA using $\rho_{\max}$ [47]:

$$\rho_{\max} = \frac{\rho(H,Q)}{\sqrt{1+\frac{1}{\mathrm{SNR}}}}\hat{p}\sqrt{\frac{Var(P)}{Var(\hat{P})}},$$

where $\hat{p}$ denotes the maximum probability for the probability distribution of the time of processing secret data, $\rho(H,Q)$ is Pearson's correlation coefficient of the hypothetical power consumption H and the actual 'signal' Q, and $F = \sqrt{\frac{Var(P)}{Var(\hat{P})}}$ depends on the device.

The SNR and $\hat{p}$ can be computed at design time and are determined by, for instance, the chosen logic style and chosen way of disarranging the time. $\rho(H,Q)$ on the other hand, depends on how well one knows the power consumption characteristics of a device. Also, F is hard to assess at the design time. In the worst case, $\rho(H,Q) = F = 1$. The minimum number of samples

$n_{\min}$ can then be computed as follows:

$$n_{\min} = 3 + 8\left(\frac{Z_\alpha}{\ln\left(\frac{1+\rho_{\max}}{1-\rho_{\max}}\right)}\right)^2.$$

The quantile Z_α determines the distance between the distributions with $\rho = 0$ and $\rho = \rho_{\max}$. Experiments showed that $\alpha = 0.9$ appears to be a reasonable value.

3.2 Signal-to-Noise Ratio

There are three sources of noise when performing DPA: the activity of the rest of the circuit N_1, the jitter on the attacked gate N_2, and the bias introduced by the S-boxes N_3 [39]. Assuming the first two are low enough, the third makes is possible to extract bits of the (sub)key, although there will also be 'ghost peaks' for wrong candidate keys. They are modelled as noise in the following definition of the SNR, where N_k is the size of the keyspace and $\overline{\text{Sig}} = \frac{1}{N_k}\Sigma_k \text{Sig}(k)$ is the signal mean:

$$\text{SNR}(\text{Sig}) = \frac{\text{Sig}(0) - \overline{\text{Sig}}}{\sqrt{\frac{1}{N_k}\Sigma_k\left(\text{Sig}(k) - \overline{\text{Sig}}\right)^2}}.$$

Computing the SNR of the DPA signal of an S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ then leads to the typical values as in Table 3.1.

SNR	S-box characteristic
$\frac{1}{m}$	Lower bound for unbalanced S-boxes, reached only by rank 1 affine S-boxes.
1	Lower bound for balanced S-boxes, reached only when m is a perfect square.
$\sqrt{m}$	Rank m affine S-boxes; no S-box or linear S-box.
$\sqrt{2^n}/m$	Approximative lower bound for bent S-boxes.
3.6	DES S-Box 1 ($n = 6$, $m = 4$).
9.6	AES SubBytes ($n = m = 8$).
9.8	A Maiorana-McFarland bent S-box ($n = 8$, $m = 4$) [17].
$\sqrt{2^n}$	Upper bound for balanced S-boxes.

Table 3.1: The SNR of the DPA signal of several S-boxes [39].

The less linear an S-box, the higher the DPA SNR becomes, and vice versa. In general, the best shielded against linear or differential cryptanalysis, the more vulnerable to a DPA attack [18, 62].

3.3 Transparency Order

Countermeasures for DPA attacks are frequently added only to the implementation. The transparency order was introduced to quantify the resistance of an S-box against DPA attacks [62]. It is built on the assumption that the previous register value that is replaced by the ciphertext is constant, although in practice it will depend on the type of implementation. In other words, the transparency order assumes the Hamming weight power consumption model and is not trivially extendable to different models. The transparency order $\mathrm{TO}(S)$ of an S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is defined as:

$$\mathrm{TO}(S) = \max_{\beta \in \mathbb{F}_2^m} \left(|m - 2\mathrm{HW}(\beta)| - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{\substack{v \in \mathbb{F}_2^m \\ \mathrm{HW}(v)=1}} (-1)^{v \cdot \beta} W_{D_a S}(0, v) \right| \right),$$

where the dot product $a \cdot b = \bigoplus_{i=0}^{m-1} a_i b_i$, and $W_{D_a S}(u, v)$ denotes the Walsh transform of the derivative of S with respect to a. $W_{D_a S}(0, v)$ equals the correlation coefficient between $x \mapsto v \cdot S(x)$ and $x \mapsto v \cdot S(x + a)$.

The smaller the transparency order of an S-box, the higher its resistance to DPA attacks, assuming the Hamming weight power consumption model is actually suitable. $0 \leq \mathrm{TO}(S) \leq m$. $\mathrm{TO}(S) = m$ if every coordinate function of S is bent. There is a trade-off between nonlinearity and the transparency order [18].

3.4 Success Rate

An adversary performing a side-channel attack is successful if the key is correctly identified at classification. The success rate can be determined by classifying r randomly chosen measurements out of the key extraction measurement series [35]. The random choice is repeated one thousand times and the success rate is then defined as the percentage of success in determining the correct key value. The success rate depends on actual power measurements and is therefore of little use at the design stage of a cipher.

3.5 New Signal-to-Noise Ratio

The previous SNR metric treats traces as random variables, while they are in fact functions of the ciphertext. The previous SNR and transparency order are also suboptimal because they treat ghost peaks as noise, although they are predictable and can be used. A newer definition of the SNR elaborates on the transparency order, but uses the maximum likelihood estimator as DPA distinguisher [38].

Let $\varepsilon = \mathrm{HD}(k_g, k_c)$ be the Hamming distance between the key guess k_g and the correct key k_c. Then the set $\{\Delta(\varepsilon), \varepsilon \neq 0\}$ are the ghost peaks. Let tr denote the trace operator $\mathrm{tr} f = \sum_x f(x)$, and τ_k the translation of vector k being $\tau_k(\cdot) \mapsto k \oplus \cdot$. Then, assuming S-box S is balanced, a DPA attack will be easier when the following metric yields a higher value:

$$\min_{\varepsilon \neq 0} \sum_{e \in \mathbb{F}_2^m} \left(\mathrm{tr} \sum_{b, b' \in \mathbb{N}^m} (-1)^{S_b \circ \tau_e} \cdot \left((-1)^{S_{b'} \circ \tau_\varepsilon} - (-1)^{S_{b'}} \right) \right)^2 .$$

3.6 Guessing Entropy

Previous metrics do not allow claims that one countermeasure is better than the other, as they depend on the attacker model. An information theoretic metric and an actual security metric were proposed to allow to make such claims [68].

The information-theoretic metric evaluates the amount of information in the side-channel leakage, measured by an unbounded adversary in terms of measurement queries. The mutual information between the key class variable C and the random vector denoting the side-channel observations generated with q queries $\mathbf{L}_q$, is then:

$$I(C; \mathbf{L}_q) = H[C] - H[C|\mathbf{L}_q] = \mathbf{E}_c \mathbf{H'}^q_{c,c},$$

where $H[C] = \mathbf{E}_c - \log_2 \Pr[c]$ is the entropy of the key class variable C before any side-channel attack has been performed and $\mathbf{H'}^q_{c,c^*} = H[C] - \mathbf{H}^q_{c,c^*}$ is an entropy reduction matrix.

The latter metric is more flexible than the success rate and measures the average number of key candidates to test after the side-channel attack. In a side-channel key-guessing experiment, let $c \in C$ be the equivalence class of the correct key and let the adversary output a guess vector $[g_1, g_2, \ldots, g_{|C|}]$ with different key candidates sorted according to their likelihood. The output of the experiment is i such that $g_i = c$. Then the guessing entropy metric is defined as the expected value of that experiment. The guessing entropy directly indicates the average remaining workload of the side-channel adversary, in contrast to the success rate.

3.7 Confusion Coefficient

The success rate is the ultimate metric that incorporates the effects of all factors including algorithms, implementations, and attacks. However, it is unclear what kind of intrinsic features instilled in cryptographic algorithms lead to SCA-related properties and to what extent. The confusion coefficient has been introduced to address this [30].

Let the confusion coefficient κ over two subkeys (k_i, k_j) be defined as:

$$\kappa(k_i, k_j) = \mathbb{E}[(V|k_i - V|k_j)^2],$$

where $V(x, k)$ is a selection function that depends on both a known plaintext or ciphertext x, and the key k, and where $V|k_i$ denotes the output of V for a fixed x and the key k_i. For instance, for the classic single-bit DPA attack, $V(x, k) = \text{LSB}(S(x \oplus k))$, having outcomes of either 0 or 1. The confusion coefficients for all possible keys form a discrete distribution and fall in only a few values, called the characteristic confusion values of an S-box. Ideally, in the case of DPA, $\kappa(k_i, k_j) = 0.5$ for any k_i and k_j. For CPA on AES, the selection function is 8 bits, so the ideal value would be 4. It is claimed that when the variance σ^2 of this distribution is lower, key candidates behave more similarly and randomly and the cipher will therefore be more resistant to DPA attacks. However, recent simulation results suggest that the exact opposite is true [40].

It has been demonstrated how the confusion coefficients have a direct effect on the success rate [30]. Let the three-way confusion coefficient $\tilde{\kappa}$ can be defined as:

$$\tilde{\kappa}(k_h, k_i, k_j) = \mathbb{E}[(V|k_h - V|k_i)(V|k_h - V|k_j)].$$

Then the asymptotic success rate of single-bit DPA is:

$$\text{SR} = \Phi_{N_k - 1}\left(\frac{\delta_0}{2} \frac{\kappa\sqrt{n}}{\sqrt{\mathbf{K} + (\delta_0/2)^2(\mathbf{K} - \kappa\kappa^T)}}\right),$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution, δ_0 is the SNR of the implementation, and $\mathbf{K}$ is the $(N_k - 1) \times (N_k - 1)$ confusion matrix for the correct key k_c with elements $\varkappa_{ij} = \tilde{\kappa}(k_c, k_{gi}, k_{gj})$. Lower confusion coefficients imply a higher success rate, and therefore a more resistant S-box.

In [27], confusion coefficients have been extended to higher-order DPA attacks and in the presence of masks.

3.7.1 Applying the Confusion Coefficient

To further understand how the confusion coefficient behaves in practice and to see if it matches our intuitions, we computed all distributions for several scenario's. We use the following power consumption models as selection functions.

- Hamming weight: $V(x, k) = \mathrm{HW}(S(x \oplus k))$.
- Hamming distance: $V(x, k) = \mathrm{HD}(S(x \oplus k), x)$. This corresponds to a RAM-based FGPA implementation that takes one cycle per round, ignoring the linear permutation and attacking the first round [69].
- Value: $V(x, k) = x \oplus k$.
- Weighted: let $n = |k|$, let $X = S(x \oplus k)$, let X_i denote the i^{th} bit of X, and let $w(i)$ denote a function that assigns a weight to a bit i. Then $V(x, k) = \sum_{i=0}^{n-1} w(i)X_i$. Here, $w(i) = 0.8$ if $i < \frac{n-1}{2}$, $w(i) = 1$ if $i = \frac{n-1}{2}$, and $w(i) = 1.2$ if $i > \frac{n-1}{2}$.
- Pairs: $V(x, k) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} X_i X_j$, where again $X = S(x \oplus k)$. Note that this model is nonlinear.

We apply these to all S-boxes of the ciphers mentioned in Section 2.4 for the following scenario's.

- First-order confusion coefficient without any mask.
- First-order confusion coefficient with a low-entropy masking scheme (LEMS).
- Second-order confusion coefficient with a LEMS.
- Second-order confusion coefficient with a full-entropy masking scheme.

The LEMS are taken from [56].

- 4x4 S-boxes: $\{0, 1, 6, 7, 10, 11, 12, 13\}$.
- 5x5 S-boxes: $\{0, 3, 12, 15, 21, 22, 25, 26\}$.
- 8x8 S-boxes: $\{3, 24, 63, 85, 96, 110, 140, 165, 178, 203, 214, 249\}$.

All individual results can be found in Appendix A. To be able to compare the distributions for S-boxes of different sizes, they are normalised according to the coefficient of variation, i.e. $c_v = \frac{\sigma}{\mu}$. A higher c_v implies being more resistant to power analysis attacks, assuming that the effects of the mean and that of the variance are approximately similar.

We first investigate whether the confusion coefficient reflects the expected results for a LEMS. In the case of a first-order attack, the confusion coefficient should be zero for any linear power consumption model. This indeed holds for the Hamming weight, value, and Hamming distance model. However, this is not exactly true for the weighted power model, although the mean is very close to zero. The pairs model also yields a confusion coefficient of zero, which is somewhat unexpected as the model is nonlinear and the LEMS are designed for a linear power consumption model.

For a second-order attack, we expect the confusion coefficient to behave just as good for a LEMS as for a full-entropy masking scheme. It can be seen that this holds for all selection functions except for the nonlinear pairs model, where both the mean and the variance are lower for the LEMS scenario. This is to be expected, as a smaller set of possible mask values will never perform better than the full set of masks.

Then we look at whether the confusion coefficient respects the differences between S-boxes and whether it is possible to say anything meaningful about the S-boxes in practice. Figure 3.1 shows the coefficients of variation, sorted by coefficient of variation for the Hamming weight model in the first-order scenario without any mask. When looking at Figure 3.1a and Figure 3.1b separately, a number of things can be seen.

First, the order of the S-boxes is mostly preserved by different power consumption models and different scenario's. S-boxes that have a small c_v typically have a small c_v in every case. The graph for the value-based power consumption model stands out the most, but this model only really applies to a single-bit setting and is merely included to see how the confusion coefficient responds to it. It is noteworthy that the Hamming distance model shows some spikes with the S-boxes of Ketje, Keyak, and ICEPOLE in the first-order scenario and some negative spikes with all 5-bit S-boxes in the second-order scenario. It remains unclear why this is the case.

Secondly, it can be seen in Figure 3.1 that the 8-bit S-boxes are all nearly at the bottom, implying that either larger S-boxes are easier to attack according to the confusion coefficient, or the mean of the confusion coefficient distribution is more decisive that its variance. However, apparently the S-box of Ketje and Keyak is less DPA-resistant than SCREAM's S-box and the encryption S-box of Joltik is less DPA-resistant than the one used in Ascon, if the confusion coefficient is to be believed. Simulation results and results on real data need to confirm this. The results of our simulations are discussed in Section 3.11.

So far, we have seen how the confusion coefficient reflects the influence of a LEMS and what can be stated about different S-boxes. When Figure 3.1a and Figure 3.1b are combined, it can also be seen how the confusion coefficient is largely unaffected by the order of the attack. The results attained in the first-order case without any mask propagate to the second-order case with a full-entropy masking scheme most fittingly for the Hamming weight power consumption model, as is expected for a Boolean masking scheme.

3.8 Modified Transparency Order

The definition of the transparency order lead to some inconsistencies [20]. First of all, it is redundant in terms of considering the maximum over all $\beta \in \mathbb{F}_2^m$. The transparency order does in fact not depend on β. Furthermore, it was claimed that $\text{TO}(S) = m$ for an S-box S that only has bent and pairwise complement coordinate functions. However, DPA should then not be possible at all under a Hamming weight power model.

Instead, the transparency order should be defined as follows:

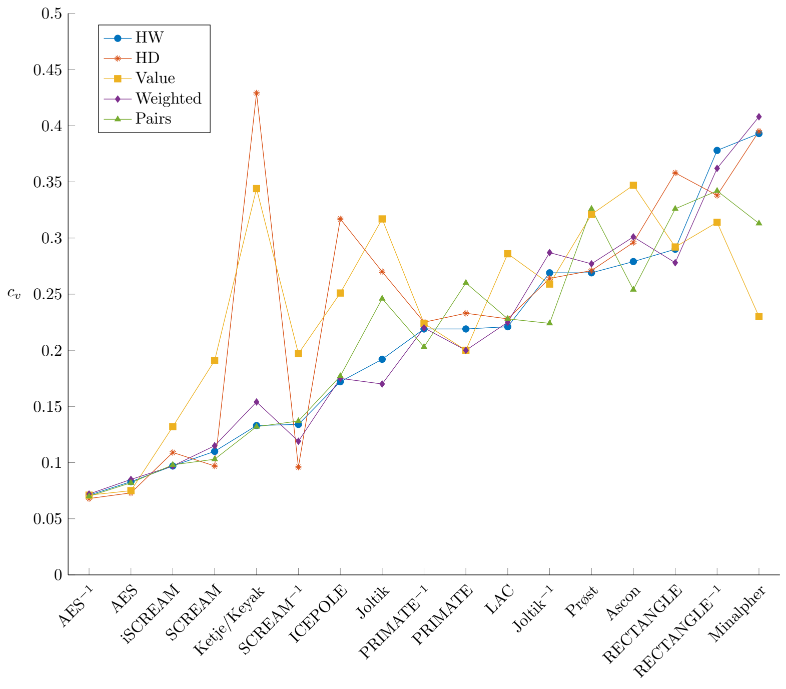
$$\text{TO}(S) = \max_{\beta \in \mathbb{F}_2^m} \left(m - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} \sum_{j=1}^{m} \left| \sum_{i=1}^{m} (-1)^{\beta_i \oplus \beta_j} \mathcal{C}_{S_i, S_j}(a) \right| \right),$$

where $\mathcal{C}_{f_1,f_2}(\omega)$ denotes the cross-correlation spectrum between two Boolean functions and is defined as $\mathcal{C}_{f_1,f_2}(\omega) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f_1(x) \oplus f_2(x \oplus \omega)}$.

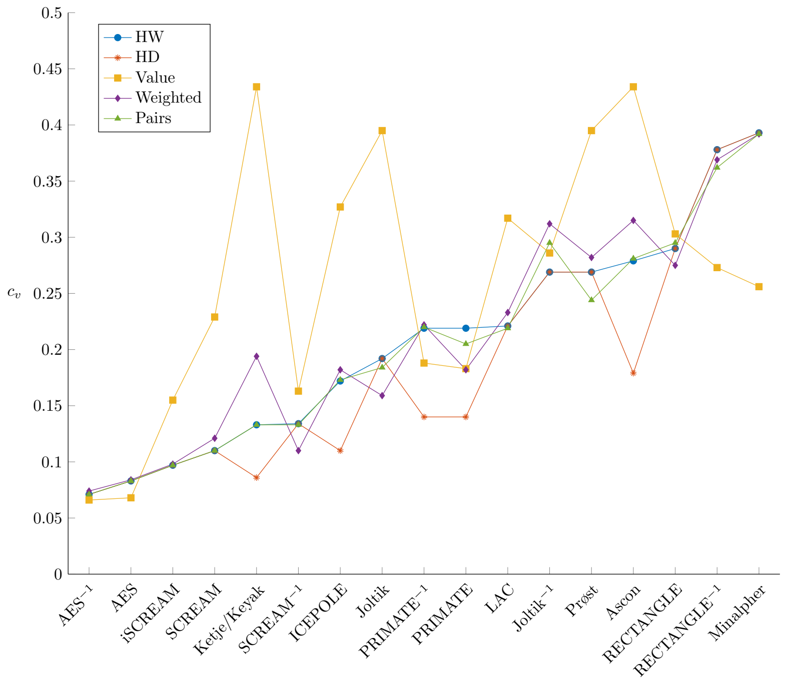
3.9 Second Minimum Distance

Another metric was proposed [65] that only considers permutation S-boxes, implying $n = m$. We call the set of all n-tuples

$$\mathcal{D}_S = \{(D_{k_g,1}(S), \dots, D_{k_g,n}(S))\}_{k_g \in \mathbb{F}_2^n}$$



(a) First-order without any mask.



(b) Second-order with a full-entropy masking scheme.

Figure 3.1: The c_v of confusion coefficient distributions of S-boxes.

the Relative Power Spectrum (RPS) of S, that contains RPS vectors with differential traces $D_{k_g,j}$ per coordinate function j, for each key candidate k_g.

The RPS vector for the correct key is $((-1)^{\beta_1}, \ldots, (-1)^{\beta_n})$, where $\beta = (\beta_1, \ldots, \beta_n)$ is the precharge logic. In a scenario where there is noise, an attacker should consider the key belonging to the RPS vector with the closest distance from the vector for the correct key. For permutation S-boxes S and S' that are affine equivalent as $S'(x) = S(Ax \oplus b)$, the RPSs of S and S' will be permutations of each other, but this does not hold for (extended) affine equivalence in general.

Consequently, we can create a distance profile of S by computing all Euclidean distances from $((-1)^{\beta_1}, \ldots, (-1)^{\beta_n})$ to all RPS vectors in the RPS of S. The minimum distance occurs for the correct key, but the second minimum distance $\mathcal{D}2_n(S)$ can be used as metric. A lower second minimum distance implies that false candidates are closer to the correct key, which implies a DPA attack will be harder.

3.10 Comparison

Most of these metrics take a different approach at measuring the effectiveness of an SCA attack or the resistance of a given S-box or cipher to an SCA attack. It is therefore not surprising that they are used in different ways and that they yield different results for the CAESAR ciphers.

The number of measurements, the success rate, and the guessing entropy depend on actual measurements. They are therefore on their own not useful at the design stage of a cipher, although they can theoretically be approximated using a model. The other metrics only depend on the S-box and on certain assumptions. For instance, most metrics assume that there is added noise that follows a Gaussian distribution and is key-independent. For the S-boxes given in Section 2.4, the metrics yield the results as given in Table 3.2, assuming a first-order attack scenario without any mask and a Hamming weight power model.

Recall that the SNR, (modified) transparency order, and second minimum distance should be lower to achieve a higher resistance to DPA attacks, while the variance of the confusion coefficient should be higher. The results from Table 3.2 are depicted graphically in Figure 3.2, sorted by the confusion coefficient variance. Note that the blue graphs (SNR, transparency order, modified transparency order) are plotted against the left blue axis and the orange graphs (second minimum distance, confusion coefficient variance) are plotted against the right orange axis, and we are not interested in absolute values. It can clearly be seen that the SNR, the modified transparency order, and the confusion coefficient are consistent in their predictions on which S-box is more resistant to DPA attacks. This helps in making these metrics more credible as being useful.

The original transparency order and the second minimum distance metric are less consistent with these metrics, but it has been pointed out in [20] that the first contains several errors and this could account for the deviation that especially shows for the 5-bit S-boxes. The second minimum distance appears to be fairly consistent in the 5-bit case, but this holds not so much for the other sizes. This metric has only recently been proposed and requires further research.

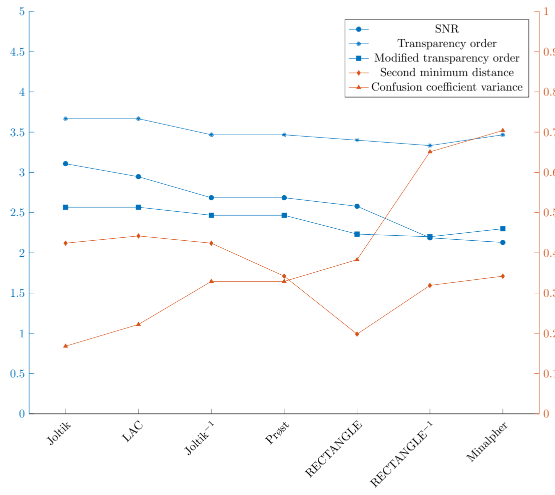
Regarding the CAESAR competition, the SNR, modified transparency order, and confusion coefficient metrics expect Minalpher, Ascon, and the inverse of SCREAM to have the most DPA-resistant S-boxes of sizes 4x4, 5x5, and 8x8, respectively.

3.11 Simulations

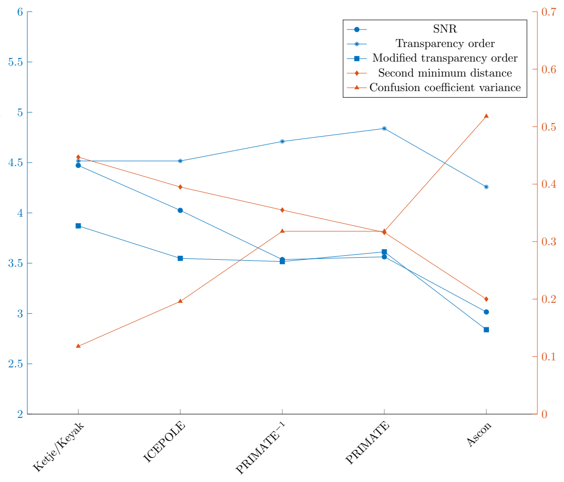
We performed first-order CPA simulations on the ‘best’ and ‘worst’ 4-bit, 5-bit, and 8-bit S-boxes, i.e those of Minalpher, Joltik, Ascon, Ketje/Keyak, and the decryption S-boxes of SCREAM and AES, to see if the results predicted by the metrics can be related to the simulation results.

S-box	SNR	TO	Modified TO	Second minimum distance	Confusion coefficient $(\mu; \sigma^2)$
AES	9.600	7.860	6.916	0.292	(4.016; 0.111)
AES^{-1}	10.583	7.851	6.911	0.278	(4.016; 0.081)
iSCREAM	8.667	7.700	6.763	0.249	(4.016; 0.151)
SCREAM	7.921	7.589	6.605	0.247	(4.016; 0.194)
SCREAM^{-1}	6.779	7.600	6.628	0.247	(4.016; 0.288)
Ascon	3.015	4.258	2.839	0.200	(2.581; 0.518)
ICEPOLE	4.025	4.516	3.548	0.395	(2.581; 0.196)
Ketje/Keyak	4.472	4.516	3.871	0.447	(2.581; 0.118)
PRIMATE	3.563	4.839	3.613	0.316	(2.581; 0.318)
PRIMATE^{-1}	3.536	4.710	3.516	0.355	(2.581; 0.318)
Joltik	3.108	3.667	2.567	0.424	(2.133; 0.168)
Joltik^{-1}	2.685	3.467	2.467	0.424	(2.133; 0.329)
LAC	2.946	3.667	2.567	0.442	(2.133; 0.222)
Minalpher	2.129	3.467	2.300	0.342	(2.133; 0.704)
Prøst	2.685	3.467	2.467	0.342	(2.133; 0.329)
RECTANGLE	2.579	3.400	2.233	0.198	(2.133; 0.383)
RECTANGLE^{-1}	2.187	3.333	2.200	0.319	(2.133; 0.651)

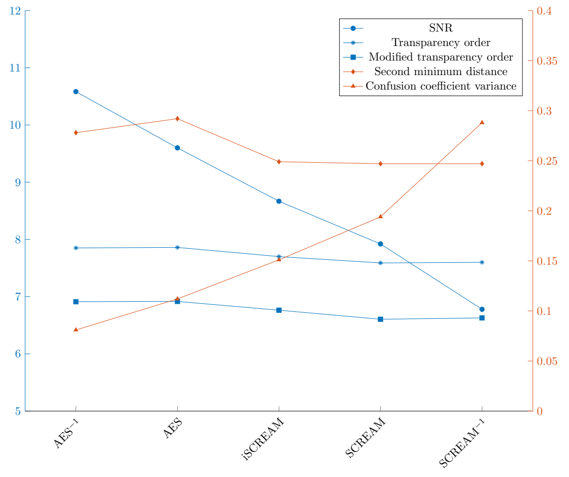
Table 3.2: Cryptographic properties of S-boxes w.r.t. SCA attacks.



(a) 4x4 S-boxes.



(b) 5x5 S-boxes.



(c) 8x8 S-boxes.

Figure 3.2: Cryptographic properties of S-boxes w.r.t. SCA attacks.

We used uniformly random inputs, a Hamming weight power model, and added random white Gaussian noise with SNR = 0.1 to simulate a physical device. We plot the relative rank r' of the correct subkey against the number of traces N in Figure 3.3, where the relative rank r' is defined as $r' = \frac{r-1}{2^{|k|}-1} + 1$, to be able to compare S-boxes of different sizes. Every point on the graph represents the average of 100 iterations to account for the randomness. It can be seen that while the metrics are clear on which S-box should be least and which one should be most resistant to DPA attacks, this can hardly be related to the CPA simulation results, where all S-boxes behave fairly similar. Adding more noise does not change this behaviour. Even the small differences of the simulations between the S-boxes are unfortunately hardly reflected by the metrics. Therefore, it appears that metrics are not of much use at the design stage, even when the power model adequately reflects reality.

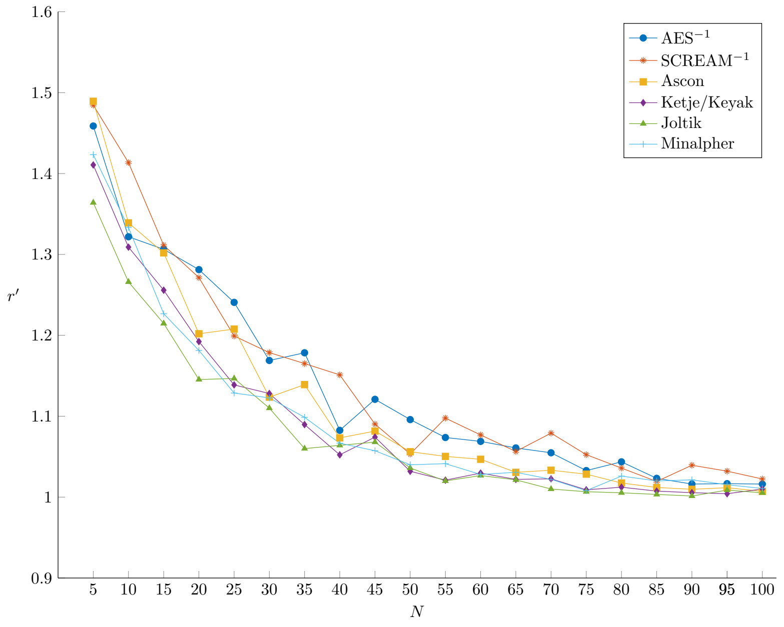


Figure 3.3: CPA simulation results under a Hamming weight power model, $SNR = 0.1$.

Chapter 4

Reducing Nonlinear Operations for Masking

It can be seen that SCA attacks remain a serious threat to the security of cryptographic algorithms in practice and that this is no different for the newly proposed authenticated encryption schemes. It is known that masking can be expensive, although low-entropy masking schemes allow us to make some trade-off, and for that reason, it is worth to investigate how much adding a masking scheme will actually cost for the various CAESAR candidates and for RECTANGLE. For Boolean and polynomial masking schemes, the cost of masking linear operations such as `XOR` gates are linear in the number of operations, while they are quadratic for, for instance, `AND` gates. For this reason, the nonlinear parts are reduced first. Chapter 5 discusses the costs of masking the linear parts.

4.1 Multiplicative Complexity

The fact that masking nonlinear operations comes with larger costs has lead to the notion of multiplicative complexity [10]. Because S-boxes are usually the only nonlinear part in the cryptographic algorithm, it suffices to only look at the S-boxes of the CAESAR candidates for now. The goal is to achieve an S-box implementation with as few `AND`/`OR` gates as possible, i.e. with a multiplicative complexity as low as possible.

For 4x4 S-boxes, it is known by exhaustive search what multiplicative complexity is optimal and which S-boxes achieve this value, while retaining optimal security properties related to traditional cryptanalysis [36, 70]. The so-called ‘Class 13’ S-boxes require only 4 `AND` and 4 `XOR` operations. By combining certain 4-bit S-boxes 8-bit variants were constructed that achieve a fairly low multiplicative complexity (11 `AND`s and 25 `XOR`s, compared to AES’s 32 `AND`s and 83 `XOR`s [9]), but it is not generally known what is optimal for 5x5 or 8x8 S-boxes, or how to find the optimal implementation for a given S-box.

4.2 Logic Synthesis

Finding optimal circuit implementations for a given function is an old problem in electronics. Although the (unbounded) circuit minimisation problem is Σ_2^P-complete, several heuristic algorithms have been successful in industry. After Karnaugh maps and the Quine-McCluskey algorithm, the Espresso tool by IBM became much used [14]. However, the tool only performs two-level minimisation. This means that circuits are viewed as sum-of-products (SOPs) and minimisation does not make use of terms that are shared for multiple outputs. The University of California at Berkeley subsequently created misII [12] that performs multi-level minimisation and the well-known SIS package on top of that [67]. For Windows, the tool Logic Friday provides a GUI on top of Espresso and misII. A more recent tool is Berkeley's ABC [13].

Attempts to design optimal circuits in the context of S-boxes using these tools have not proven very successful. One reason is that the tools are incapable of dealing with `XOR` gates. Minimising Boolean formulas is typically done over the basis $\{\texttt{AND}, \texttt{OR}, \texttt{NOT}\}$, and in hardware `NAND` and `NOR` gates are more efficient and therefore preferred. In cryptography, however, `XOR` gates are linear and therefore cheaper with respect to the cost of masking.

4.3 Multi-Level Minimisation with `XOR`

Sometimes, Boolean formulas can be expressed much more efficiently when `XOR`s are used. A line of research has explored methods to minimise circuits with `XOR` gates [55]. Several heuristic algorithms have been suggested, such as AOXMIN-MV [29] and KGPMIN [21]. They have also been compared to SIS [58].

For exclusive-sums-of-products (ESOPs), more heuristic algorithms exist (e.g. EXORCISM-4 [51]), but there are also some exact minimisation methods, such as min-tau2 [41]. Most heuristic algorithms are aimed at optimising very large functions, as that is a more common goal in practice, and are therefore focussed on speed and memory efficiency. In the S-box case, however, the function is only very small and an exact optimal solution can usually be found in a feasible amount of time.

One remaining problem with this $\{\texttt{AND}, \texttt{OR}, \texttt{XOR}\}$-minimisation[1] is that it is usually not trivial to express constraints such as that you are looking for the minimum number of nonlinear gates. The following completely different approach solves this problem.

4.4 A SAT Solver-Based Approach

The Boolean satisfiability problem, SAT, was one of the first problems proven to be NP-complete. It is the problem of determining whether there exists a valuation for variables that satisfies a given Boolean formula. For example, the formula $a \wedge \neg a$ is unsatisfiable; there does not exist a value for a such that $a \wedge \neg a$ becomes true. Despite that the problem is in NP, there is a large community dedicated to create SAT solvers that are able to find the solution to quite large sets of formulas. International competitions are being held to create the fastest and best SAT solver. If a SAT solver terminates, it will either output `SAT` along with a valuation that satisfies the formulas, or `UNSAT`.

The decisional multiplicative complexity problem (DMC) is defined as: "does there exist a circuit with at most p `AND` gates that implements an S-box S?" Because DMC $\in$ NP[2], a polynomial reduction can be made to SAT. This means that we can use the progress made in the SAT solver community for the benefit of circuit minimisation. This will only be feasible for small S-boxes (e.g. with $n = m \leq 5$), but it has proven to be a successful approach [23, 24, 54]. This approach is also optimal, if we are able to find a value p that outputs `SAT` for which $p - 1$ outputs `UNSAT`.

[1] `NOT` can be implemented by `XOR`, e.g. $\neg a = a \oplus 1$, and is therefore not considered separately.

[2] A formal proof is omitted, but it is not hard to imagine how it would look like. It can clearly be verified in polynomial time that a given circuit has a number of `AND` gates $\leq p$ by counting them. Verifying that the circuit correctly implements a given S-box is similar to the polynomial verification procedure of CSAT.

In our reduction to SAT, we use the following notation for variables. The notation is borrowed from [54]. Note that in [24], q and t are swapped.

- x_i represent S-box inputs.
- y_i represent S-box outputs.
- q_i represent binary gate inputs.
- t_i represent binary gate outputs.
- a_i represent all connections.

All formulas are first written in algebraic normal form (ANF) and later converted to conjunctive normal form (CNF) using Bard's tool[3] [3]. All binary gate inputs are written as being equal to a linear combinations of S-box inputs and previous gate outputs, to represent an unlimited number of `XOR` gates. Similarly, the S-box outputs are equal to a linear combination of S-box inputs and gate outputs. The p `AND` gates are hard-coded. For instance, for a 4x4 S-box with $p = 3$, the following prototype will be used, where concatenation denotes a logical `AND`, and $+$ a logical `XOR`.

$$\begin{aligned}
q_0 &= a_0 + a_1x_0 + a_2x_1 + a_3x_2 + a_4x_3 \\
q_1 &= a_5 + a_6x_0 + a_7x_1 + a_8x_2 + a_9x_3 \\
t_0 &= q_0q_1 \\
q_2 &= a_{10} + a_{11}x_0 + a_{12}x_1 + a_{13}x_2 + a_{14}x_3 + a_{15}t_0 \\
q_3 &= a_{16} + a_{17}x_0 + a_{18}x_1 + a_{19}x_2 + a_{20}x_3 + a_{21}t_0 \\
t_1 &= q_2q_3 \\
q_4 &= a_{22} + a_{23}x_0 + a_{24}x_1 + a_{25}x_2 + a_{26}x_3 + a_{27}t_0 + a_{28}t_1 \\
q_5 &= a_{29} + a_{30}x_0 + a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}t_0 + a_{35}t_1 \\
t_2 &= q_4q_5 \\
y_0 &= a_{36}x_0 + a_{37}x_1 + a_{38}x_2 + a_{39}x_3 + a_{40}t_0 + a_{41}t_1 + a_{42}t_2 \\
y_1 &= a_{43}x_0 + a_{44}x_1 + a_{45}x_2 + a_{46}x_3 + a_{47}t_0 + a_{48}t_1 + a_{49}t_2 \\
y_2 &= a_{50}x_0 + a_{51}x_1 + a_{52}x_2 + a_{53}x_3 + a_{54}t_0 + a_{55}t_1 + a_{56}t_2 \\
y_3 &= a_{57}x_0 + a_{58}x_1 + a_{59}x_2 + a_{60}x_3 + a_{61}t_0 + a_{62}t_1 + a_{63}t_2
\end{aligned}$$

Now we want this to be true for all input-output pairs of the S-box, so 2^n copies of this formula set are concatenated, in which all a_i remain the same, but all other variables get renamed with new indices. For example, the first formula of the second copy will be $q_6 = a_0+a_1x_4+a_2x_5+a_3x_6+a_4x_7$.

Finally, to bind the actual S-box to this description, it can be added as a simple truth table, although an algebraic description is also possible. For instance, if $n = m = 4$ and $S(3) = 12$, the corresponding line of the truth table can be added with the following set of equations, as $3_{10} = 0011_2$ and $12_{10} = 1100_2$.

[3]A link to a download can be found at http://www.cryptosystem.net/aes/tools.html.

$$
\begin{aligned}
0 &= x_{12}\\
0 &= x_{13}\\
0 &= 1 + x_{14}\\
0 &= 1 + x_{15}\\
0 &= 1 + y_{12}\\
0 &= 1 + y_{13}\\
0 &= y_{14}\\
0 &= y_{15}
\end{aligned}
$$

Various SAT solvers can be used to attempt to find a solution to this set of formulas. Recent versions of MiniSat and CryptoMiniSat[4] were used to obtain the results in Table 4.1 on the multiplicative complexity of CAESAR S-boxes and RECTANGLE. Their proofs can be found in Section 4.5.

S-box	Multiplicative complexity
AES	≤ 32 [9]
AES^{-1}	≤ 32 [9]
iSCREAM	≤ 12 [37]
SCREAM	≤ 11 [37]
SCREAM^{-1}	≤ 11 [37]
Ascon	5
ICEPOLE	6*
Ketje/Keyak	5
PRIMATE	$\in \{6, 7\}$*
PRIMATE^{-1}	$\in \{6, 7, 8, 9, 10\}$*
Joltik	4
Joltik^{-1}	4*
LAC	4*
Minalpher	5*
Prøst	4
RECTANGLE	4
RECTANGLE^{-1}	4*

Table 4.1: Multiplicative complexity of CAESAR S-boxes and RECTANGLE. The * indicates that Section 4.5 provides a new S-box implementation with less nonlinear gates compared to implementations published by the designers.

It can be seen that Minalpher has the only 4-bit S-box among these for which an implementation using 4 `AND` operations does not exist. Unfortunately, over 1000 CPU hours were insufficient to

[4]MiniSat 2.2.0 and CryptoMiniSat 4.2.0, mostly using default settings.

further narrow down the multiplicative complexity of the S-boxes used in PRIMATE or of any of the 8x8 S-boxes.

4.5 Multiplicative Complexity Proofs

4.5.1 Ascon

Theorem 1. *The S-box of Ascon has a multiplicative complexity of 5.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 4. For a multiplicative complexity of 5, the following implementation can be obtained, although the specification of course already provides a more efficient implementation with 5 `AND` operations [28].

$$\begin{aligned}
q_0 &= 1 + x_3 + x_4 & q_7 &= x_0 \\
q_1 &= 1 + x_4 & t_3 &= q_6 * q_7 \\
t_0 &= q_0 * q_1 & q_8 &= x_3 + t_1 + t_2 \\
q_2 &= x_0 + x_2 + x_4 & q_9 &= x_1 + x_2 \\
q_3 &= x_1 & t_4 &= q_8 * q_9 \\
t_1 &= q_2 * q_3 & y_0 &= x_0 + x_1 + x_2 + x_3 + t_1 \\
q_4 &= x_0 + x_1 + x_4 & y_1 &= x_0 + x_2 + x_3 + x_4 + t_4 \\
q_5 &= x_1 & y_2 &= x_1 + x_2 + x_3 + t_0 \\
t_2 &= q_4 * q_5 & y_3 &= x_0 + x_1 + x_2 + x_3 + x_4 + t_3 \\
q_6 &= x_3 + x_4 & y_4 &= x_3 + x_4 + t_2
\end{aligned}$$

□

4.5.2 ICEPOLE

Theorem 2. *The S-box of ICEPOLE has a multiplicative complexity of 6.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 5. For a multiplicative complexity of 6, the following implementation can be obtained:

$$\begin{aligned}
q_0 &= x_0 + x_3 + x_4 \\
q_1 &= x_0 + x_3 \\
t_0 &= q_0 * q_1 \\
q_2 &= 1 + x_2 + x_4 \\
q_3 &= x_2 + x_3 + x_4 \\
t_1 &= q_2 * q_3 \\
q_4 &= x_2 + t_0 + t_1 \\
q_5 &= x_0 + x_2 + x_3 + x_4 + t_1 \\
t_2 &= q_4 * q_5 \\
q_6 &= x_0 + x_1 + x_4 \\
q_7 &= x_1 + x_4 \\
t_3 &= q_6 * q_7 \\
q_8 &= x_1 + x_2 + t_0 + t_1 + t_2
\end{aligned}$$

$$\begin{aligned}
q_9 &= x_0 + x_1 + t_0 + t_1 + t_2 \\
t_4 &= q_8 * q_9 \\
q_{10} &= 1 + x_2 + t_1 + t_3 + t_4 \\
q_{11} &= 1 + x_0 + t_4 \\
t_5 &= q_{10} * q_{11} \\
y_0 &= x_0 + t_0 + t_1 + t_2 + t_5 \\
y_1 &= x_0 + x_1 + x_2 + x_3 + x_4 + t_2 + \cdots \\
&\quad \cdots + t_3 + t_4 + t_5 \\
y_2 &= x_0 + x_3 + t_1 + t_2 + t_3 + t_4 + t_5 \\
y_3 &= x_0 + x_2 + x_3 + x_4 + \cdots \\
&\quad \cdots + t_0 + t_1 + t_2 + t_3 + t_4 + t_5 \\
y_4 &= x_2 + x_4 + t_0 + t_1 + t_2 + t_4 + t_5
\end{aligned}$$

□

4.5.3 Ketje and Keyak

Theorem 3. *The S-box of Ketje and Keyak has a multiplicative complexity of 5.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 4. For a multiplicative complexity of 5, the following implementation can be obtained, although the specification of course already provides an implementation with 5 `AND` operations [7, 6].

$$\begin{aligned}
q_0 &= x_0 + x_1 \\
q_1 &= x_1 \\
t_0 &= q_0 * q_1 \\
q_2 &= x_0 + x_3 + x_4 \\
q_3 &= x_0 + x_3 \\
t_1 &= q_2 * q_3 \\
q_4 &= x_2 + x_3 \\
q_5 &= x_3 \\
t_2 &= q_4 * q_5 \\
q_6 &= x_2
\end{aligned}$$

$$\begin{aligned}
q_7 &= x_1 + x_2 \\
t_3 &= q_6 * q_7 \\
q_8 &= x_3 \\
q_9 &= x_4 \\
t_4 &= q_8 * q_9 \\
y_0 &= x_0 + t_3 \\
y_1 &= x_1 + t_2 \\
y_2 &= x_2 + x_4 + t_4 \\
y_3 &= t_1 + t_4 \\
y_4 &= x_4 + t_0
\end{aligned}$$

□

4.5.4 PRIMATE

Theorem 4. *The S-box of PRIMATE has a multiplicative complexity* $\in \{6, 7\}$.

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 5. For a multiplicative complexity of 7, the following implementation can be obtained:

$$
\begin{aligned}
q_0 &= 1 + x_0 + x_3 \\
q_1 &= 1 + x_1 \\
t_0 &= q_0 * q_1 \\
q_2 &= 1 + x_1 + x_3 \\
q_3 &= x_0 + x_2 \\
t_1 &= q_2 * q_3 \\
q_4 &= x_0 + x_1 + x_4 \\
q_5 &= x_0 + x_2 + x_3 \\
t_2 &= q_4 * q_5 \\
q_6 &= x_0 + x_2 + x_3 + x_4 \\
q_7 &= 1 + x_1 + x_2 + x_4 \\
t_3 &= q_6 * q_7 \\
q_8 &= x_0 + x_1 + x_2 + x_3 + x_4
\end{aligned}
$$

$$
\begin{aligned}
q_9 &= x_2 + t_0 + t_3 \\
t_4 &= q_8 * q_9 \\
q_{10} &= 1 + x_0 + x_3 + x_4 \\
q_{11} &= x_0 + x_4 \\
t_5 &= q_{10} * q_{11} \\
q_{12} &= 1 + x_1 + x_2 + t_0 + t_2 + t_3 + t_4 \\
q_{13} &= x_2 + x_3 \\
t_6 &= q_{12} * q_{13} \\
y_0 &= x_1 + x_3 + t_2 + t_3 + t_5 + t_6 \\
y_1 &= x_0 + x_4 + t_1 + t_2 + t_3 + t_4 + t_5 + t_6 \\
y_2 &= x_1 + x_2 + x_4 + t_1 + t_3 + t_4 + t_5 \\
y_3 &= x_0 + x_2 + x_3 + x_4 + t_3 + t_4 + t_5 + t_6 \\
y_4 &= x_2 + t_0 + t_2 + t_3 + t_4 + t_5 + t_6
\end{aligned}
$$

□

Theorem 5. *The inverse S-box of PRIMATE has a multiplicative complexity* $\in \{6, 7, 8, 9, 10\}$.

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 5. For a multiplicative complexity of 10, the following implementation can be obtained:

$$
\begin{aligned}
q_0 &= x_0 + x_2 + x_3 \\
q_1 &= 1 + x_2 + x_4 \\
t_0 &= q_0 * q_1 \\
q_2 &= x_0 \\
q_3 &= x_1 \\
t_1 &= q_2 * q_3 \\
q_4 &= x_2 + x_3 + t_0 \\
q_5 &= 1 + x_1 \\
t_2 &= q_4 * q_5 \\
q_6 &= x_1 + t_1 + t_2 \\
q_7 &= x_2 + x_4 \\
t_3 &= q_6 * q_7 \\
q_8 &= x_2 + t_0 + t_2 + t_3 \\
q_9 &= x_0 + x_3 + x_4 + \cdots \\
&\quad \cdots + t_1 + t_2 + t_3 \\
t_4 &= q_8 * q_9 \\
q_{10} &= x_0 + x_2 + x_3 + \cdots \\
&\quad \cdots + t_1 + t_2 + t_3 \\
q_{11} &= x_1 + x_3 + t_0 + t_2 \\
t_5 &= q_{10} * q_{11}
\end{aligned}
$$

$$
\begin{aligned}
q_{12} &= x_0 + x_4 \\
q_{13} &= t_0 + t_3 + t_4 + t_5 \\
t_6 &= q_{12} * q_{13} \\
q_{14} &= 1 + x_0 + x_1 + x_2 + x_4 + t_0 + t_1 + \cdots \\
&\quad \cdots + t_3 + t_4 + t_5 + t_6 \\
q_{15} &= x_0 + x_3 + t_0 + t_1 + t_2 + t_4 + t_6 \\
t_7 &= q_{14} * q_{15} \\
q_{16} &= 1 + x_2 + x_3 + t_2 + t_5 \\
q_{17} &= 1 + x_0 + x_1 + x_4 + t_0 + t_1 + t_2 + \cdots \\
&\quad \cdots + t_3 + t_6 + t_7 \\
t_8 &= q_{16} * q_{17} \\
q_{18} &= x_4 + t_2 + t_5 + t_6 + t_8 \\
q_{19} &= 1 + x_0 + x_1 + x_4 + t_4 + t_7 + t_8 \\
t_9 &= q_{18} * q_{19} \\
y_0 &= x_0 + x_1 + t_0 + t_6 + t_7 + t_9 \\
y_1 &= t_0 + t_3 + t_6 \\
y_2 &= t_3 + t_5 + t_6 + t_7 \\
y_3 &= t_1 + t_2 + t_4 \\
y_4 &= x_1 + t_0 + t_4 + t_8
\end{aligned}
$$

□

4.5.5 Joltik

Theorem 6. *The S-box of Joltik has a multiplicative complexity of 4.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 3. For a multiplicative complexity of 4, the following implementation can be obtained:

$$\begin{array}{ll}
q_0 = 1 + x_1 & t_2 = q_4 * q_5 \\
q_1 = 1 + x_0 & q_6 = x_0 + x_1 + x_2 + t_1 + t_2 \\
t_0 = q_0 * q_1 & q_7 = x_1 + x_3 + t_0 \\
q_2 = 1 + x_1 + x_3 + t_0 & t_3 = q_6 * q_7 \\
q_3 = 1 + x_2 & y_0 = x_3 + t_0 \\
t_1 = q_2 * q_3 & y_1 = x_0 + x_2 + x_3 + t_0 + t_1 + t_2 \\
q_4 = x_2 & y_2 = x_1 + x_3 + t_0 + t_2 \\
q_5 = x_2 + x_3 + t_0 & y_3 = x_3 + t_1 + t_2 + t_3
\end{array}$$

□

Theorem 7. *The inverse S-box of Joltik has a multiplicative complexity of 4.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 3. For a multiplicative complexity of 4, the following implementation can be obtained:

$$\begin{array}{ll}
q_0 = 1 + x_0 & t_2 = q_4 * q_5 \\
q_1 = 1 + x_1 & q_6 = x_1 + x_2 + t_2 \\
t_0 = q_0 * q_1 & q_7 = x_1 \\
q_2 = 1 + x_0 + x_2 + t_0 & t_3 = q_6 * q_7 \\
q_3 = x_1 + x_3 + t_0 & y_0 = x_2 + t_1 + t_2 + t_3 \\
t_1 = q_2 * q_3 & y_1 = x_0 + x_2 + t_0 + t_2 \\
q_4 = x_0 + x_3 & y_2 = x_3 + t_0 \\
q_5 = x_3 & y_3 = x_0 + t_1
\end{array}$$

□

4.5.6 LAC

Theorem 8. *The S-box of LAC has a multiplicative complexity of 4.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 3. For a multiplicative complexity of 4, the following implementation can be obtained:

$$
\begin{aligned}
q_0 &= x_0 & t_2 &= q_4 * q_5\\
q_1 &= x_0 + x_1 & q_6 &= 1 + x_0\\
t_0 &= q_0 * q_1 & q_7 &= x_1 + t_1 + t_2\\
q_2 &= 1 + x_1 + x_3 + t_0 & t_3 &= q_6 * q_7\\
q_3 &= x_0 + x_2 & y_0 &= x_2 + x_3 + t_1 + t_3\\
t_1 &= q_2 * q_3 & y_1 &= x_0 + t_0 + t_3\\
q_4 &= 1 + x_0 + x_1 + x_2 & y_2 &= t_1 + t_2\\
q_5 &= 1 + x_3 + t_0 & y_3 &= x_1 + x_2 + x_3 + t_0
\end{aligned}
$$

□

4.5.7 Minalpher

Theorem 9. *The S-box of Minalpher has a multiplicative complexity of 5.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 4. For a multiplicative complexity of 5, the following implementation can be obtained:

$$
\begin{aligned}
q_0 &= x_1 + x_2 + x_3 & q_7 &= 1 + x_0 + x_1 + x_2 + t_0 + t_2\\
q_1 &= x_1 & t_3 &= q_6 * q_7\\
t_0 &= q_0 * q_1 & q_8 &= x_0 + x_2 + x_3 + t_0 + t_1 + t_2 + t_3\\
q_2 &= x_0 + x_1 + x_3 & q_9 &= x_1 + x_2 + t_0 + t_2 + t_3\\
q_3 &= x_1 + x_2 + t_0 & t_4 &= q_8 * q_9\\
t_1 &= q_2 * q_3 & y_0 &= x_2 + t_4\\
q_4 &= x_0 + t_0 & y_1 &= x_0 + x_2 + t_1\\
q_5 &= x_0 + x_1 + x_2 + t_0 & y_2 &= t_0 + t_3\\
t_2 &= q_4 * q_5 & y_3 &= t_1 + t_2 + t_3\\
q_6 &= 1 + x_0 + x_1 + t_1 & &
\end{aligned}
$$

□

4.5.8 Prøst

Theorem 10. *The S-box of Prøst has a multiplicative complexity of 4.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 3. For a multiplicative complexity of 4, the following implementation can be obtained:

$$
\begin{aligned}
q_0 &= x_0 & t_2 &= q_4 * q_5\\
q_1 &= x_1 & q_6 &= x_1 + x_3 + t_0\\
t_0 &= q_0 * q_1 & q_7 &= x_2 + t_0\\
q_2 &= x_0 + x_2 + t_0 & t_3 &= q_6 * q_7\\
q_3 &= 1 + x_0 + x_1 + \cdots & y_0 &= x_2 + t_0\\
&\quad \cdots + x_2 + x_3 + t_0 & y_1 &= t_1 + t_2\\
t_1 &= q_2 * q_3 & y_2 &= x_0 + t_3\\
q_4 &= x_3 & y_3 &= x_1 + t_1\\
q_5 &= x_0 + x_2 + x_3 + t_0
\end{aligned}
$$

□

4.5.9 RECTANGLE

Theorem 11. *The S-box of RECTANGLE has a multiplicative complexity of 4.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 3. For a multiplicative complexity of 4, the following implementation can be obtained:

$$
\begin{aligned}
q_0 &= x_0 + x_1 & t_2 &= q_4 * q_5\\
q_1 &= x_0 + x_1 + x_3 & q_6 &= x_0 + x_1 + x_2 + t_0 + t_2\\
t_0 &= q_0 * q_1 & q_7 &= x_1 + t_2\\
q_2 &= x_3 & t_3 &= q_6 * q_7\\
q_3 &= 1 + x_1 & y_0 &= x_2 + x_3 + t_0 + t_1 + t_3\\
t_1 &= q_2 * q_3 & y_1 &= x_2 + t_0 + t_1\\
q_4 &= 1 + x_0 + x_3 + t_1 & y_2 &= x_0 + x_2 + t_0\\
q_5 &= 1 + x_0 + x_2 + x_3 + t_0 & y_3 &= x_0 + x_1 + x_2 + t_0 + t_2
\end{aligned}
$$

□

Theorem 12. *The inverse S-box of RECTANGLE has a multiplicative complexity of 4.*

Proof. SAT solvers output `UNSAT` for a multiplicative complexity of 3. For a multiplicative complexity of 4, the following implementation can be obtained:

$$
\begin{aligned}
q_0 &= 1 + x_2 & t_2 &= q_4 * q_5\\
q_1 &= 1 + x_0 + x_3 & q_6 &= x_1 + x_3 + t_0\\
t_0 &= q_0 * q_1 & q_7 &= x_1 + t_1 + t_2\\
q_2 &= x_0 + x_1 + x_3 & t_3 &= q_6 * q_7\\
q_3 &= 1 + x_2 + x_3 + t_0 & y_0 &= x_2 + t_1 + t_2 + t_3\\
t_1 &= q_2 * q_3 & y_1 &= x_0 + x_2 + x_3 + t_1 + t_3\\
q_4 &= x_2 + x_3 + t_0 & y_2 &= t_1 + t_2\\
q_5 &= x_0 + x_1 + t_0 & y_3 &= x_3 + t_0 + t_1 + t_2
\end{aligned}
$$

□

Chapter 5

Choosing Efficient Operations for Masking

Some high-level operations commonly performed in ciphers are more suitable for a Boolean masking scheme as opposed to an arithmetic masking scheme. The amount of overhead that comes with masking differs per operation, as was demonstrated by Messerges in [50]. Implementing masked versions of all CAESAR candidates is beyond the scope of this thesis, but this chapter sheds some light on how efficient it could be for certain candidates. In a way very similar to [50][1], fundamental operations in the core primitives of CAESAR candidates are identified. This means that for LAC, only LBlock-s is considered, and on an algorithmic level. On an implementation level, the exact operations that are used also depend on the data structure chosen to store the cipher state.

5.1 Table Lookups

Table lookups are expensive to mask. For each execution of the algorithm, a Boolean masked table has to be computed using an input mask and an output mask. The table also has to be stored somewhere in memory. This storage requirement can become a problem for smartcard implementations with little memory available. It is typically the S-box that is implemented using a lookup table. In AES, for instance, the 8x8 S-box takes 256 bytes. It can also be described arithmetically, but then the implementation will become significantly slower. PRIMATE, Joltik, LAC, and Minalpher all suggest in their proposals and their reference implementations to implement the S-box with a lookup table. This costs PRIMATE 32 bytes of storage and 16 bytes for Joltik, LAC, and Minalpher. The other CAESAR candidates selected in this thesis suggest a bitsliced implementation for the S-box, which prevents the need of a lookup table, or, in the case of ICEPOLE, Ketje, and Keyak, provide simple Boolean operations to implement the nonlinear function.

Lookup tables can be used for operations other than S-boxes as well. iSCREAM and SCREAM use a lookup table of 512 bytes for the linear layer. ICEPOLE needs 96 bytes for its round constants, assuming 12 rounds, and Joltik requires another 16 bytes for its subtweakey and 64 bytes for round constants, assuming 32 rounds. For PRIMATE and RECTANGLE, it is assumed

[1]In [50], linear transformations are discussed separately, but as these are usually implemented with bitwise Boolean functions and shifts and rotates, they are here omitted as an operation category. All notes on masking bitwise Boolean functions and shifts and rotates also apply if they are part of a linear transformation.

that the LFSR that computes the round constants is implemented using XORs and rotates. Ascon, Ketje, Keyak, Prøst, and RECTANGLE do not need any lookup table. An overview of the memory requirements per cipher is given in Table 5.1.

It is unlikely that lookup tables of these sizes will raise problems with implementations for smartcards or other small devices.

5.2 Bitwise Boolean Functions

Bitwise Boolean functions such as AND, OR and XOR are all fast to compute. They can be masked with a Boolean masking scheme, as explained in Section 2.5.3. For linear operations such as XOR, the additional cost of masking is linear in the number of operations or gates. The XOR operation is heavily used by all CAESAR candidates. The costs of masking are quadratic in the number of inputs for nonlinear operations such as AND and OR.

In the previous chapter, we minimised the number of nonlinear operations in the computation of the S-box. This is highly useful especially in the context of hardware implementations and bitsliced software implementations. However, some cipher designers suggested to use lookup tables to implement their S-boxes. For this overview, we will assume that the implementations are in line with the suggestions of the designers. In that case, AES, PRIMATE, Joltik, LAC, and Minalpher do not require nonlinear operations. All their nonlinearity is captured by the S-box that is implemented as a lookup table.

The core primitives in iSCREAM, SCREAM, Ascon, ICEPOLE, Ketje, Keyak, Prøst, and RECTANGLE do need AND and/or OR operations. Most of these are part of the implementation of the S-boxes, although some arithmetic operations also sometimes use an AND or an OR.

5.3 Shifts and Rotates

Fixed shifts and rotates are easy to mask, also using a Boolean masking scheme. The masks shift and rotate along with the data. To be more precise, if we want to mask a fixed right rotate, denoted $\ggg$, by n, the masked output becomes $x' \ggg n$ and the new mask becomes $r_x \ggg n$. This means that the additional cost is again linear in the number of shifts and rotates. These operations are used by the majority of the CAESAR candidates that are discussed. Only SCREAM and Minalpher do not make use of them. For AES-like designs, it is typically the ShiftRows operation where shifts/rotates are used[2]. Linear transformations are also frequently implemented using bitwise Boolean functions and shifts/rotates. Ketje and Keyak use shifts/rotates in the θ operation as well. In iSCREAM, it is with the computation of the tweakey, while in Ascon it is with the round constants. LAC uses them in its key schedule.

(Sensitive) data dependent shifts and rotates are a lot less efficient to mask. The shift/rotation distance now also needs to be masked. For this, an arithmetic masking scheme is more suitable. This requires some extra effort. However, such data dependent shifts and rotates do not occur in the previously mentioned authenticated encryption schemes.

5.4 Modular Additions and Multiplications

For modular additions and multiplications, an arithmetic masking scheme is more suitable, as explained in Section 2.5.3. The cost of masking modular additions then becomes linear in the number of additions, while the cost of masking modular multiplications is quadratic and should

[2]A similar operation is called ShiftPlanes in Prøst and ρ in ICEPOLE, Ketje and Keyak.

therefore be avoided. However, how efficient this is in practice depends more on how well the modulus aligns with powers of two, or the CPU word size in the case of a software implementation. Whether switching between different types of masking is necessary is also significant.

iSCREAM and SCREAM use multiplication modulo 256 for their round constants and Joltik multiplication modulo 16 for its subtweakey, although that parts needs does not need to be masked. ICEPOLE, Ketje and Keyak use a less CPU-friendly modulus. ICEPOLE needs addition modulo 4 and 5, while Ketje and Keyak need both addition and multiplication modulo 5, but only for indexing so it does not have to be masked.

5.5 Modular Polynomial Multiplications

Whether masking modular polynomial multiplications can be done efficiently depends on the way it is implemented. Lookup table approaches are possible, but usually inefficient. AES, PRIMATE and Joltik use modular polynomial multiplications over $\mathbb{F}_2[x]/x^8 + x^4 + x^3 + x + 1$, $\mathbb{F}_2[x]/x^5 + x^2 + 1$, and $\mathbb{F}_2[x]/x^4 + x + 1$, respectively. Masking modular polynomial multiplications is then similar to masking regular modular multiplications.

5.6 Results

The observations above are summarised in Table 5.1. It can be seen that the expected cost of masking will be within reasonable limits for all discussed candidates. iSCREAM and SCREAM are most likely to be most expensive to mask, as they use a relatively large lookup table, the expensive `AND`/`OR` operations, and expensive modular multiplications, which also require a switch from a Boolean to an arithmetic masking scheme that causes additional overhead. Ascon, LAC, Minalpher, Prøst, and RECTANGLE all stand out when it comes to the expected cost of masking. They do not need switching to an arithmetic masking scheme and they use only very small lookup tables or a low amount of `AND`/`OR` operations.

Operation	Table lookups	Bitwise Boolean operations	Shifts and rotates	Modular additions and multiplications	Modular polynomial multiplications
AES	256 bytes	`XOR`	Fixed		✓
iSCREAM	512 bytes	`AND`,`OR`,`XOR`	Fixed	$\times$ mod 256	
SCREAM	512 bytes	`AND`,`OR`,`XOR`		$\times$ mod 256	
Ascon		`AND`,`OR`,`XOR`	Fixed		
ICEPOLE	96 bytes	`AND`,`XOR`	Fixed		
Ketje/Keyak		`AND`,`XOR`	Fixed		
PRIMATE	25 bytes	`XOR`	Fixed		✓
Joltik	64 bytes	`XOR`	Fixed	$+$ mod 16	✓
LAC	16 bytes	`XOR`	Fixed		
Minalpher	16 bytes	`XOR`			
Prøst		`AND`,`XOR`	Fixed		
RECTANGLE		`AND`,`OR`,`XOR`	Fixed		

Table 5.1: Fundamental operations in CAESAR candidates and RECTANGLE.

Chapter 6

Conclusion

Within the context of the CAESAR candidates, we started out by asking ourselves two questions. The first was about the use of metrics to assess the intrinsic resistance of ciphers to implementation- and device-dependent attacks at the design stage. We have discussed the metrics that have appeared in literature and we compared the metrics that are usable at the design stage. It appears that the SNR, the modified transparency order, and the confusion coefficient are all consistent in their predictions of which S-boxes are most resistant to attacks that makes use of side-channels such as differential power analysis. According to them, the S-boxes of Minalpher, Ascon, and the inverse S-box of SCREAM are the most SCA-resistant 4-bit, 5-bit, and 8-bit S-boxes in the CAESAR competition, respectively. The S-boxes of Joltik, Ketje and Keyak, and the inverse S-box of AES are the least resistant S-boxes, if the metrics are to be believed.

For the confusion coefficient, we have shown in more detail how it behaves in practice. It nicely propagates to higher-order attacks and it responds to masking schemes as is to be expected. It becomes clear once again that the chosen power model and selection function are of high importance.

Although the SNR, modified transparency order, and the confusion coefficient all look very credible as metrics and behave as they should under various circumstances, CPA simulation results do not reflect the expectations suggested by the metrics. Even the small differences that the S-boxes show are in conflict with the metrics.

The second question we asked ourselves was on design strategies to reduce the additional costs of applying masking countermeasures. The additional costs are linear in the number of linear gates and quadratic in the number of nonlinear gates. We therefore first attempted to reduce the number of nonlinear gates in S-box implementations. We calculated the multiplicative complexity for the S-boxes by encoding them in logic and by feeding the problem to a SAT solver. For most S-boxes, we were able to obtain an implementation with a provably minimum number of nonlinear gates. In particular, we provide implementations for the 4-bit S-boxes except Minalpher with only 4 `AND` gates, for the S-boxes of Ascon, Ketje, Keyak, and Minalpher with 5 `AND` gates, and for ICEPOLE's S-box with 6 `AND` gates.

We then zoomed out and looked at the high-level design of the ciphers. We identified the fundamental operations that are used and looked at the costs of masking them. We also looked at the memory requirements and at overhead caused by switching between Boolean and arithmetic masking schemes. It appears that the costs of masking are expected to be the least for Ascon, LAC, Minalpher, Prøst, and RECTANGLE.

From these results, a general design strategy can be distilled to only use fundamental operations that are cheap to mask using a Boolean masking scheme, and to minimise the number of required nonlinear operations by choosing a secure S-box with a low multiplicative complexity.

Chapter 7

Future Work

In the process of dealing with the problems as stated in Sections 1.1 and 1.2, more questions were raised that, despite being interesting, could unfortunately not be answered within the given amount of time or within the scope of this thesis. They are put here as ideas for future research.

In this work, we have seen that metrics are not always useful to assess the intrinsic resistance to side-channel attacks of an S-box at the design stage. While the predictions are hard to relate to simulation results, the metrics are mostly consistent. It is not immediately clear why this is the case, as they take different approaches. In the case of the transparency order line of work and the confusion coefficients, it is unknown how they are precisely related. More theoretical analysis is required to understand the exact nature of this relation. This will lead to a better insight of which metric is most useful, if any.

Secondly, it is unclear if a 5-bit S-box should be more or less resistant to power analysis attacks compared to S-boxes of different sizes, or in general, how metric results for differently sized S-boxes should be interpreted and compared. In Section 3.7, it turned out that the confusion coefficient behaves oddly with 5-bit S-boxes in a Hamming distance power model. There appears to be little reason for this, yet 5-bit S-boxes, or S-boxes of other odd sizes, have not nearly received as much attention from researchers as even sizes.

Thirdly, some work has been done to compare the predictions given by metrics to actual success rates of actual SCA attacks for certain S-boxes, e.g. in [30]. However, in other cases such as in our work and in [20], the comparison is performed on artificial simulation data generated according to a certain leakage function. More research using real data is required to better understand the use of design-time metrics in practice for general S-boxes.

Furthermore, methods exist to generally minimise the number of operations to perform a certain function, and methods exist to minimise the number of nonlinear operations. For efficiency and speed reasons, it would be interesting to know if there exist general strategies to combine such methods to reach optimal implementations with respect to masking. Additional overhead is always the argument against implementing a masking scheme, so when this can be minimised and the overhead in practice becomes small enough, it will lead to more secure implementations. Our results in Sections 4.4 and 4.5 yield a minimum number of nonlinear gates, but at the cost of being generous with the number of `XOR` gates. It would be interesting to learn more about the best way to further minimise the number of linear operations.

Finally, identifying fundamental operations in cipher constructions and making predictions about the additional costs of masking, as done in Chapter 5, are only as good as that. To learn about the actual costs in practice, masked versions of all CAESAR candidates need to be implemented, as was done in [50]. However, it is more convenient to do so once the number of candidates has been drastically reduced in subsequent rounds, to reduce the amount of work.

Bibliography

[1] Farzaneh Abed, Christian Forler, and Stefan Lucks. General overview of the first-round CAESAR candidates for authenticated encryption. Cryptology ePrint Archive, Report 2014/792, 2014. http://eprint.iacr.org/.

[2] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. PRIMATEs v1. CAESAR submission, 2014. http://competitions.cr.yp.to/round1/primatesv1.pdf.

[3] Gregory V. Bard, Nicolas T. Courtois, and Chris Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-solvers. Cryptology ePrint Archive, Report 2007/024.

[4] G.V. Bard. *Algebraic Cryptanalysis*. Springer Science & Business Media, 2009.

[5] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles van Assche. The Keccak reference, January 2011. http://keccak.noekeon.org/.

[6] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles van Assche, and Ronny van Keer. CAESAR submission: Ketje v1, 2014. http://competitions.cr.yp.to/round1/keyakv1.pdf.

[7] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles van Assche, and Ronny van Keer. CAESAR submission: Keyak v1, 2014. http://competitions.cr.yp.to/round1/ketjev1.pdf.

[8] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[9] Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In Paola Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 178–189. Springer Berlin Heidelberg, 2010.

[10] Joan Boyar, René Peralta, and Denis Pochuev. On the multiplicative complexity of Boolean functions over the basis $(\wedge, \oplus, 1)$. *Theoretical Computer Science*, 235(1):43–57, 2000.

[11] An Braeken. *Cryptographic Properties of Boolean Functions and S-boxes.* PhD thesis, Katholieke Universiteit Leuven, March 2006.

[12] R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A.R. Wang. MIS: A multiple-level logic optimization system. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 6(6):1062–1081, November 1987.

[13] Robert Brayton and Alan Mishchenko. ABC: An academic industrial-strength verification tool. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *Computer Aided Verification*, volume 6174 of *Lecture Notes in Computer Science*, pages 24–40. Springer Berlin Heidelberg, 2010.

[14] Robert K Brayton. *Logic minimization algorithms for VLSI synthesis*, volume 2. Springer Science & Business Media, 1984.

[15] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer Berlin Heidelberg, 2004.

[16] A. Canteaut and P. Charpin. Decomposing bent functions. *Information Theory, IEEE Transactions on*, 49(8):2004–2019, Aug 2003.

[17] Claude Carlet. On the confusion and diffusion properties of Maiorana-McFarland's and extended Maiorana-McFarland's functions. *Journal of Complexity*, 20(2-3):182–204, 2004. Festschrift for Harald Niederreiter, Special Issue on Coding and Cryptography.

[18] Claude Carlet. On highly nonlinear S-boxes and their inability to thwart DPA attacks. In *Progress in Cryptology – INDOCRYPT 2005*, pages 49–62. Springer, 2005.

[19] Florent Chabaud and Serge Vaudenay. Links between differential and linear cryptanalysis. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 356–365. Springer Berlin Heidelberg, 1994.

[20] Kaushik Chakraborty, Sumanta Sarkar, Subhamoy Maitra, Bodhisatwa Mazumdar, Debdeep Mukhopadhyay, and Emmanuel Prouff. Redefining the transparency order. Cryptology ePrint Archive, Report 2014/367, 2014. http://eprint.iacr.org/.

[21] Santanu Chattopadhyay, Samir Roy, and Parimal Pal Chaudhuri. KGPMIN: An efficient multilevel multioutput AND-OR-XOR minimizer. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 16(3):257–265, 1997.

[22] Jean-Sébastien Coron and Louis Goubin. On Boolean and arithmetic masking against differential power analysis. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 231–237. Springer Berlin Heidelberg, 2000.

[23] Nicolas Courtois, Daniel Hulme, and Theodosis Mourouzis. Solving circuit optimisation problems in cryptography and cryptanalysis, 2011. http://eprint.iacr.org/.

[24] Nicolas Courtois, Theodosis Mourouzis, and Daniel Hulme. Exact logic minimization and multiplicative complexity of concrete algebraic and cryptographic circuits. *International Journal On Advances in Intelligent Systems*, 6(3 and 4):165–176, 2013.

[25] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[26] Blandine Debraize. Efficient and provably secure methods for switching from arithmetic to boolean masking. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 107–121. Springer Berlin Heidelberg, 2012.

[27] Aidong Adam Ding, Liwei Zhang, Yunsi Fei, and Pei Luo. A statistical model for higher order DPA on masked devices. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 147–169. Springer Berlin Heidelberg, 2014.

[28] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon. CAESAR submission, 2014. http://ascon.iaik.tugraz.at.

[29] E. V. Dubrova, D. M. Miller, and J. C. Muzio. AOXMIN-MV: A heuristic algorithm for AND-OR-XOR minimization. In *Proc. 4th International Workshop on the Applications of the Reed-Muller Expansion in Circuit Design*, pages 37–53, 1999.

[30] Yunsi Fei, Aidong Adam Ding, Jian Lao, and Liwei Zhang. A statistics-based fundamental model for side-channel attack analysis. Cryptology ePrint Archive, Report 2014/152, 2014. http://eprint.iacr.org/.

[31] Julie Ferrigno and Martin Hlavác. When AES blinks: introducing optical side channel. *Information Security, IET*, 2(3):94–98, 2008.

[32] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer Berlin Heidelberg, 2001.

[33] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461. Springer Berlin Heidelberg, 2014.

[34] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer Berlin Heidelberg, 2008.

[35] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In *Cryptographic Hardware and Embedded Systems – CHES 2006*, pages 15–29. Springer, 2006.

[36] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varıcı. LS-designs: Bitslice encryption for efficient masked software implementations. In *Fast Software Encryption – FSE 2014*, 2014.

[37] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varıcı, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. SCREAM & iSCREAM. CAESAR submissions, 2014. http://competitions.cr.yp.to/round1/screamv1.pdf.

[38] Sylvain Guilley, Philippe Hoogvorst, Renaud Pacalet, and Johannes Schmidt. Improving side-channel attacks by exploiting substitution boxes properties. *International Conference on Boolean Functions: Cryptography and Applications (BFCA)*, pages 1–25, 2007.

[39] Sylvain Guilley and Renaud Pacalet. Differential power analysis model and some results. In *CARDIS 2004*, pages 127–142. Kluwer Academic Publishers, 2004.

[40] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. A theoretical study of Kolmogorov-Smirnov distinguishers. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design – COSADE 2014*, volume 8622 of *Lecture Notes in Computer Science*, pages 9–28. Springer International Publishing, 2014.

[41] Takashi Hirayama and Yasuaki Nishitani. Exact minimization of AND–EXOR expressions of practical benchmark functions. *Journal of Circuits, Systems, and Computers*, 18(03):465–486, 2009.

[42] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Joltik v1. CAESAR submission, 2014. http://competitions.cr.yp.to/round1/joltikv1.pdf.

[43] Elif Bilge Kavun, Martin M. Lauridsen, Gregor Leander, Christian Rechberger, Peter Schwabe, Tolga Yalçın, and DTU Compute. Prøst v1. CAESAR submission, 2014. http://competitions.cr.yp.to/round1/proestv11.pdf.

[44] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology – CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer Berlin Heidelberg, 1999.

[45] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer Berlin Heidelberg, 1996.

[46] Gregor Leander and Axel Y. Poschmann. On the classification of 4 bit S-boxes. In Claude Carlet and Berk Sunar, editors, *Arithmetic of Finite Fields*, volume 4547 of *Lecture Notes in Computer Science*, pages 159–176. Springer Berlin Heidelberg, 2007.

[47] Stefan Mangard. Hardware countermeasures against DPA – a statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer Berlin Heidelberg, 2004.

[48] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Berlin Heidelberg, 1994.

[49] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer Berlin Heidelberg, 2000.

[50] Thomas S. Messerges. Securing the AES finalists against power analysis attacks. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer Berlin Heidelberg, 2001.

[51] Alan Mishchenko and Marek Perkowski. Fast heuristic minimization of exclusive-sums-of-products. 2001.

[52] Paweł Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. ICEPOLE: High-speed, hardware-oriented authenticated encryption. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 392–413. Springer Berlin Heidelberg, 2014.

[53] Paweł Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. Icepole v1. CAESAR submission, 2014. http://competitions.cr.yp.to/round1/icepolev1.pdf.

[54] Theodosis Mourouzis. *Optimizations in Algebraic and Differential Cryptanalysis.* PhD thesis, UCL (University College London), 2015.

[55] Unni Narayanan and C. L. Liu. Low power logic synthesis for XOR based circuits. In *ICCAD*, pages 570–574, 1997.

[56] Maxime Nassar, Sylvain Guilley, and Jean-Luc Danger. Formal analysis of the entropy / security trade-off in first-order masking countermeasures against side-channel attacks. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *Progress in Cryptology – INDOCRYPT 2011*, volume 7107 of *Lecture Notes in Computer Science*, pages 22–39. Springer Berlin Heidelberg, 2011.

[57] Kaisa Nyberg. Perfect nonlinear S-boxes. In DonaldW. Davies, editor, *Advances in Cryptology – EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 378–386. Springer Berlin Heidelberg, 1991.

[58] Ritesh Parikh and Santanu Chattopadhyay. Power-aware multi-level AND-XOR network synthesis. *International Journal of Computers and Applications*, 33(1):22, 2011.

[59] Stjepan Picek. *Applications of Evolutionary Computation to Cryptology.* PhD thesis, Radboud University, July 2015.

[60] Stjepan Picek, Lejla Batina, Domagoj Jakobovic, Barış Ege, and Marin Golub. S-box, SET, match: A toolbox for S-box analysis. In *Information Security Theory and Practice. Securing the Internet of Things – 8th IFIP WG 11.2 International Workshop, WISTP 2014, Heraklion, Crete, Greece, June 30 - July 2, 2014. Proceedings*, pages 140–149, 2014.

[61] Stjepan Picek, Elena Marchiori, Lejla Batina, and Domagoj Jakobovic. Combining evolutionary computation and algebraic constructions to find cryptography-relevant boolean functions. In *Parallel Problem Solving from Nature – PPSN XIII*, pages 822–831. Springer, 2014.

[62] Emmanuel Prouff. DPA attacks and S-boxes. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption*, volume 3557 of *Lecture Notes in Computer Science*, pages 424–441. Springer Berlin Heidelberg, 2005.

[63] O.S Rothaus. On "bent" functions. *Journal of Combinatorial Theory, Series A*, 20(3):300–305, 1976.

[64] Palash Sarkar and Subhamoy Maitra. Nonlinearity bounds and constructions of resilient Boolean functions. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 515–532. Springer Berlin Heidelberg, 2000.

[65] Sumanta Sarkar, Subhamoy Maitra, and Kaushik Chakraborty. Differential power analysis in Hamming weight model: How to choose among (extended) affine equivalent S-boxes. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology – INDOCRYPT 2014*, Lecture Notes in Computer Science, pages 360–373. Springer International Publishing, 2014.

[66] Yu Sasaki, Yosuke Todo, Kazumaro Aoki, Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui, and Shoichi Hirose. Minalpher v1. CAESAR submission, 2014. http://competitions.cr.yp.to/round1/minalpherv1.pdf.

[67] Ellen M. Sentovich, Kanwar Jit Singh, Luciano Lavagno, Cho Moon, Rajeev Murgai, Alexander Saldanha, Hamid Savoj, Paul R. Stephan, Robert K. Brayton, and Alberto Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. Technical report, University of California, Berkeley, 1992.

[68] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology – EUROCRYPT 2009*, pages 443–461. Springer, 2009.

[69] François-Xavier Standaert, Sıddıka Berna Örs, Jean-Jacques Quisquater, and Bart Preneel. Power analysis attacks against FPGA implementations of the DES. In Jrgen Becker, Marco Platzner, and Serge Vernalde, editors, *Field Programmable Logic and Application*, volume 3203 of *Lecture Notes in Computer Science*, pages 84–94. Springer Berlin Heidelberg, 2004.

[70] Markus Ullrich, Christophe De Canniere, Sebastian Indesteege, Özgül Küçük, Nicky Mouha, and Bart Preneel. Finding optimal bitsliced implementations of 4x4-bit S-boxes. In *SKEW 2011 Symmetric Key Encryption Workshop, Copenhagen, Denmark*, pages 16–17, 2011.

[71] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[72] Lei Zhang, Wenling Wu, Yanfeng Wang, Shengbao Wu, and Jian Zhang. LAC: A lightweight authenticated encryption cipher. CAESAR submission, 2014. http://competitions.cr.yp.to/round1/lacv1.pdf.

[73] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. RECTANGLE: A bit-slice ultra-lightweight block cipher suitable for multiple platforms., 2014. http://eprint.iacr.org/.

Appendix A

Confusion Coefficient Results

A.1 Original Data

S-box	First-order, no mask $(\mu; \sigma^2)$	First-order, LEMS $(\mu; \sigma^2)$	Second-order, LEMS $(\mu; \sigma^2)$	Second-order, 1 full mask $(\mu; \sigma^2)$
AES	(4.016; 0.112)	(0.000; 0.000)	(1.004; 0.007)	(1.004; 0.007)
AES^{-1}	(4.016; 0.081)	(0.000; 0.000)	(1.004; 0.005)	(1.004; 0.005)
iSCREAM	(4.016; 0.151)	(0.000; 0.000)	(1.004; 0.009)	(1.004; 0.009)
SCREAM	(4.016; 0.194)	(0.000; 0.000)	(1.004; 0.012)	(1.004; 0.012)
SCREAM^{-1}	(4.016; 0.288)	(0.000; 0.000)	(1.004; 0.018)	(1.004; 0.018)
Ascon	(2.581; 0.518)	(0.000; 0.000)	(0.645; 0.032)	(0.645; 0.032)
ICEPOLE	(2.581; 0.196)	(0.000; 0.000)	(0.645; 0.012)	(0.645; 0.012)
Ketje/Keyak	(2.581; 0.118)	(0.000; 0.000)	(0.645; 0.007)	(0.645; 0.007)
PRIMATE	(2.581; 0.318)	(0.000; 0.000)	(0.645; 0.020)	(0.645; 0.020)
PRIMATE^{-1}	(2.581; 0.318)	(0.000; 0.000)	(0.645; 0.020)	(0.645; 0.020)
Joltik	(2.133; 0.168)	(0.000; 0.000)	(0.533; 0.011)	(0.533; 0.011)
Joltik^{-1}	(2.133; 0.329)	(0.000; 0.000)	(0.533; 0.021)	(0.533; 0.021)
LAC	(2.133; 0.222)	(0.000; 0.000)	(0.533; 0.014)	(0.533; 0.014)
Minalpher	(2.133; 0.704)	(0.000; 0.000)	(0.533; 0.044)	(0.533; 0.044)
Prøst	(2.133; 0.329)	(0.000; 0.000)	(0.533; 0.021)	(0.533; 0.021)
RECTANGLE	(2.133; 0.383)	(0.000; 0.000)	(0.533; 0.024)	(0.533; 0.024)
RECTANGLE^{-1}	(2.133; 0.651)	(0.000; 0.000)	(0.533; 0.041)	(0.533; 0.041)

Table A.1: Confusion coefficient properties using the Hamming weight model.

S-box	First-order, no mask ($\mu; \sigma^2$)	First-order, LEMS ($\mu; \sigma^2$)	Second-order, LEMS ($\mu; \sigma^2$)	Second-order, 1 full mask ($\mu; \sigma^2$)
AES	(4.086; 0.089)	(0.000; 0.000)	(1.004; 0.007)	(1.004; 0.007)
AES^{-1}	(4.086; 0.077)	(0.000; 0.000)	(1.004; 0.005)	(1.004; 0.005)
iSCREAM	(4.580; 0.249)	(0.000; 0.000)	(1.004; 0.009)	(1.004; 0.009)
SCREAM	(3.702; 0.128)	(0.000; 0.000)	(1.004; 0.012)	(1.004; 0.012)
SCREAM^{-1}	(3.702; 0.127)	(0.000; 0.000)	(1.004; 0.018)	(1.004; 0.018)
Ascon	(2.581; 0.585)	(0.000; 0.000)	(0.645; 0.032)	(0.645; 0.032)
ICEPOLE	(3.226; 1.044)	(0.000; 0.000)	(0.645; 0.012)	(0.645; 0.012)
Ketje/Keyak	(3.226; 1.914)	(0.000; 0.000)	(0.645; 0.007)	(0.645; 0.007)
PRIMATE	(2.323; 0.292)	(0.000; 0.000)	(0.645; 0.020)	(0.645; 0.020)
PRIMATE^{-1}	(2.323; 0.274)	(0.000; 0.000)	(0.645; 0.020)	(0.645; 0.020)
Joltik	(1.733; 0.218)	(0.000; 0.000)	(0.533; 0.011)	(0.533; 0.011)
Joltik^{-1}	(1.733; 0.210)	(0.000; 0.000)	(0.533; 0.021)	(0.533; 0.021)
LAC	(1.733; 0.156)	(0.000; 0.000)	(0.533; 0.014)	(0.533; 0.014)
Minalpher	(1.867; 0.543)	(0.000; 0.000)	(0.533; 0.044)	(0.533; 0.044)
Prøst	(2.533; 0.472)	(0.000; 0.000)	(0.533; 0.021)	(0.533; 0.021)
RECTANGLE	(1.600; 0.329)	(0.000; 0.000)	(0.533; 0.024)	(0.533; 0.024)
RECTANGLE^{-1}	(1.600; 0.293)	(0.000; 0.000)	(0.533; 0.041)	(0.533; 0.041)

Table A.2: Confusion coefficient properties using the Hamming distance model.

S-box	First-order, no mask ($\mu; \sigma^2$)	First-order, LEMS ($\mu; \sigma^2$)	Second-order, LEMS ($\mu; \sigma^2$)	Second-order, 1 full mask ($\mu; \sigma^2$)
AES	(10965; 684557)	(0.00; 0.00)	(3.59×10^7; 5.93×10^{12})	(3.59×10^7; 5.93×10^{12})
AES^{-1}	(10965; 605787)	(0.00; 0.00)	(3.59×10^7; 5.63×10^{12})	(3.59×10^7; 5.63×10^{12})
iSCREAM	(10965; 2092633)	(0.00; 0.00)	(3.59×10^7; 3.12×10^{13})	(3.59×10^7; 3.12×10^{13})
SCREAM	(10965; 4383174)	(0.00; 0.00)	(3.59×10^7; 6.76×10^{13})	(3.59×10^7; 6.76×10^{13})
SCREAM^{-1}	(10965; 4643440)	(0.00; 0.00)	(3.59×10^7; 3.42×10^{13})	(3.59×10^7; 3.42×10^{13})
Ascon	(176; 3724)	(0.00; 0.00)	(9020; 1.53×10^7)	(9020; 1.53×10^7)
ICEPOLE	(176; 1957)	(0.00; 0.00)	(9020; 8.70×10^6)	(9020; 8.70×10^6)
Ketje/Keyak	(176; 3660)	(0.00; 0.00)	(9020; 1.53×10^7)	(9020; 1.53×10^7)
PRIMATE	(176; 1245)	(0.00; 0.00)	(9020; 2.73×10^6)	(9020; 2.73×10^6)
PRIMATE^{-1}	(176; 1550)	(0.00; 0.00)	(9020; 2.89×10^6)	(9020; 2.89×10^6)
Joltik	(45.33; 205.9)	(0.00; 0.00)	(582.53; 52921)	(582.53; 52921)
Joltik^{-1}	(45.33; 138.0)	(0.00; 0.00)	(582.53; 27799)	(582.53; 27799)
LAC	(45.33; 167.6)	(0.00; 0.00)	(582.53; 34201)	(582.53; 34201)
Minalpher	(45.33; 109.1)	(0.00; 0.00)	(582.53; 22257)	(582.53; 22257)
Prøst	(45.33; 211.6)	(0.00; 0.00)	(582.53; 52944)	(582.53; 52944)
RECTANGLE	(45.33; 175.3)	(0.00; 0.00)	(582.53; 31071)	(582.53; 31071)
RECTANGLE^{-1}	(45.33; 202.7)	(0.00; 0.00)	(582.53; 25263)	(582.53; 25263)

Table A.3: Confusion coefficient properties using the value model, as defined in Section 3.7.1.

S-box	First-order, no mask $(\mu;\sigma^2)$	First-order, LEMS $(\mu;\sigma^2)$	Second-order, LEMS $(\mu;\sigma^2)$	Second-order, 1 full mask $(\mu;\sigma^2)$
AES	(4.176; 0.125)	$(4.446\times10^{-31}; 1.334\times10^{-63})$	(1.246; 0.011)	(1.246; 0.011)
AES^{-1}	(4.176; 0.092)	$(4.446\times10^{-31}; 1.460\times10^{-63})$	(1.246; 0.009)	(1.246; 0.009)
iSCREAM	(4.176; 0.165)	$(4.446\times10^{-31}; 1.514\times10^{-63})$	(1.246; 0.015)	(1.246; 0.015)
SCREAM	(4.176; 0.231)	$(4.446\times10^{-31}; 1.456\times10^{-63})$	(1.246; 0.023)	(1.246; 0.023)
SCREAM^{-1}	(4.176; 0.247)	$(4.446\times10^{-31}; 1.404\times10^{-63})$	(1.246; 0.019)	(1.246; 0.019)
Ascon	(2.663; 0.641)	$(5.368\times10^{-32}; 1.372\times10^{-64})$	(0.770; 0.059)	(0.770; 0.059)
ICEPOLE	(2.663; 0.217)	$(5.368\times10^{-32}; 1.372\times10^{-64})$	(0.770; 0.020)	(0.770; 0.020)
Ketje/Keyak	(2.663; 0.169)	$(5.368\times10^{-32}; 1.372\times10^{-64})$	(0.770; 0.022)	(0.770; 0.022)
PRIMATE	(2.663; 0.283)	$(5.368\times10^{-32}; 2.588\times10^{-64})$	(0.770; 0.020)	(0.770; 0.020)
PRIMATE^{-1}	(2.663; 0.342)	$(5.368\times10^{-32}; 1.372\times10^{-64})$	(0.770; 0.029)	(0.770; 0.029)
Joltik	(2.219; 0.142)	$(4.602\times10^{-32}; 1.621\times10^{-64})$	(0.662; 0.011)	(0.662; 0.011)
Joltik^{-1}	(2.219; 0.404)	$(4.602\times10^{-32}; 1.621\times10^{-64})$	(0.662; 0.043)	(0.662; 0.043)
LAC	(2.219; 0.250)	$(4.602\times10^{-32}; 1.621\times10^{-64})$	(0.662; 0.024)	(0.662; 0.024)
Minalpher	(2.219; 0.818)	$(4.602\times10^{-32}; 1.621\times10^{-64})$	(0.662; 0.067)	(0.662; 0.067)
Prøst	(2.219; 0.377)	$(4.602\times10^{-32}; 1.621\times10^{-64})$	(0.662; 0.035)	(0.662; 0.035)
RECTANGLE	(2.219; 0.381)	$(4.602\times10^{-32}; 1.621\times10^{-64})$	(0.662; 0.033)	(0.662; 0.033)
RECTANGLE^{-1}	(2.219; 0.645)	$(4.602\times10^{-32}; 1.621\times10^{-64})$	(0.662; 0.060)	(0.662; 0.060)

Table A.4: Confusion coefficient properties using the weighted model, as defined in Section 3.7.1.

S-box	First-order, no mask $(\mu;\sigma^2)$	First-order, LEMS $(\mu;\sigma^2)$	Second-order, LEMS $(\mu;\sigma^2)$	Second-order, 1 full mask $(\mu;\sigma^2)$
AES	(52.706; 18.760)	(0.000; 0.000)	(149.159; 159.439)	(150.871; 158.191)
AES^{-1}	(52.706; 13.705)	(0.000; 0.000)	(149.159; 113.001)	(150.871; 114.985)
iSCREAM	(52.706; 26.498)	(0.000; 0.000)	(149.159; 187.999)	(150.871; 212.469)
SCREAM	(52.706; 29.550)	(0.000; 0.000)	(149.159; 280.230)	(150.871; 276.535)
SCREAM^{-1}	(52.706; 51.785)	(0.000; 0.000)	(149.159; 329.428)	(150.871; 400.291)
Ascon	(11.613; 8.705)	(0.000; 0.000)	(5.645; 2.417)	(10.403; 8.549)
ICEPOLE	(11.613; 4.239)	(0.000; 0.000)	(5.645; 1.294)	(10.403; 3.231)
Ketje/Keyak	(11.613; 2.364)	(0.000; 0.000)	(5.645; 0.671)	(10.403; 1.917)
PRIMATE	(11.613; 9.126)	(0.000; 0.000)	(5.645; 1.442)	(10.403; 4.531)
PRIMATE^{-1}	(11.613; 5.545)	(0.000; 0.000)	(5.645; 1.024)	(10.403; 5.242)
Joltik	(5.600; 1.891)	(0.000; 0.000)	(1.700; 0.118)	(2.750; 0.256)
Joltik^{-1}	(5.600; 1.570)	(0.000; 0.000)	(1.700; 0.238)	(2.750; 0.657)
LAC	(5.600; 1.623)	(0.000; 0.000)	(1.700; 0.136)	(2.750; 0.363)
Minalpher	(5.600; 3.070)	(0.000; 0.000)	(1.700; 0.291)	(2.750; 1.160)
Prøst	(5.600; 3.338)	(0.000; 0.000)	(1.700; 0.207)	(2.750; 0.450)
RECTANGLE	(5.600; 3.338)	(0.000; 0.000)	(1.700; 0.375)	(2.750; 0.660)
RECTANGLE^{-1}	(5.600; 3.659)	(0.000; 0.000)	(1.700; 0.189)	(2.750; 0.992)

Table A.5: Confusion coefficient properties using the pairs model, as defined in Section 3.7.1.

A.2 Normalised Data

The tables in this section show the coefficient of variation $c_v = \frac{\sigma}{\mu}$ for the various distributions. By calculating this, S-boxes of different sizes can be compared.

S-box	First-order, no mask	First-order, LEMS	Second-order, LEMS	Second-order, 1 full mask
AES	0.083	N/A	0.083	0.083
AES^{-1}	0.071	N/A	0.071	0.071
iSCREAM	0.097	N/A	0.097	0.097
SCREAM	0.110	N/A	0.110	0.110
SCREAM^{-1}	0.134	N/A	0.134	0.134
Ascon	0.279	N/A	0.279	0.279
ICEPOLE	0.172	N/A	0.172	0.172
Ketje/Keyak	0.133	N/A	0.133	0.133
PRIMATE	0.219	N/A	0.219	0.219
PRIMATE^{-1}	0.219	N/A	0.219	0.219
Joltik	0.192	N/A	0.192	0.192
Joltik^{-1}	0.269	N/A	0.269	0.269
LAC	0.221	N/A	0.221	0.221
Minalpher	0.393	N/A	0.393	0.393
Prøst	0.269	N/A	0.269	0.269
RECTANGLE	0.290	N/A	0.290	0.290
RECTANGLE^{-1}	0.378	N/A	0.378	0.378

Table A.6: c_v of the confusion coefficients using the Hamming weight model.

S-box	First-order, no mask	First-order, LEMS	Second-order, LEMS	Second-order, 1 full mask
AES	0.073	N/A	0.083	0.083
AES^{-1}	0.068	N/A	0.071	0.071
iSCREAM	0.109	N/A	0.097	0.097
SCREAM	0.097	N/A	0.110	0.110
SCREAM^{-1}	0.096	N/A	0.134	0.134
Ascon	0.296	N/A	0.179	0.179
ICEPOLE	0.317	N/A	0.110	0.110
Ketje/Keyak	0.429	N/A	0.086	0.086
PRIMATE	0.233	N/A	0.140	0.140
PRIMATE^{-1}	0.225	N/A	0.140	0.140
Joltik	0.270	N/A	0.192	0.192
Joltik^{-1}	0.264	N/A	0.269	0.269
LAC	0.228	N/A	0.221	0.221
Minalpher	0.395	N/A	0.393	0.393
Prøst	0.271	N/A	0.269	0.269
RECTANGLE	0.358	N/A	0.290	0.290
RECTANGLE^{-1}	0.338	N/A	0.378	0.378

Table A.7: c_v of the confusion coefficients using the Hamming distance model.

S-box	First-order, no mask	First-order, LEMS	Second-order, LEMS	Second-order, 1 full mask
AES	0.075	N/A	0.068	0.068
AES^{-1}	0.071	N/A	0.066	0.066
iSCREAM	0.132	N/A	0.155	0.155
SCREAM	0.191	N/A	0.229	0.229
SCREAM^{-1}	0.197	N/A	0.163	0.163
Ascon	0.347	N/A	0.434	0.434
ICEPOLE	0.251	N/A	0.327	0.327
Ketje/Keyak	0.344	N/A	0.434	0.434
PRIMATE	0.200	N/A	0.183	0.183
PRIMATE^{-1}	0.224	N/A	0.188	0.188
Joltik	0.317	N/A	0.395	0.395
Joltik^{-1}	0.259	N/A	0.286	0.286
LAC	0.286	N/A	0.317	0.317
Minalpher	0.230	N/A	0.256	0.256
Prøst	0.321	N/A	0.395	0.395
RECTANGLE	0.292	N/A	0.303	0.303
RECTANGLE^{-1}	0.314	N/A	0.273	0.273

Table A.8: c_v of the confusion coefficients using the value model.

S-box	First-order, no mask	First-order, LEMS	Second-order, LEMS	Second-order, 1 full mask
AES	0.085	0.082	0.084	0.084
AES^{-1}	0.072	0.086	0.074	0.074
iSCREAM	0.097	0.087	0.098	0.098
SCREAM	0.115	0.086	0.121	0.121
SCREAM^{-1}	0.119	0.084	0.110	0.110
Ascon	0.301	0.218	0.315	0.315
ICEPOLE	0.175	0.218	0.182	0.182
Ketje/Keyak	0.154	0.218	0.194	0.194
PRIMATE	0.200	0.300	0.182	0.182
PRIMATE^{-1}	0.220	0.218	0.222	0.222
Joltik	0.170	0.277	0.159	0.159
Joltik^{-1}	0.287	0.277	0.312	0.312
LAC	0.225	0.277	0.233	0.233
Minalpher	0.408	0.277	0.392	0.392
Prøst	0.277	0.277	0.282	0.282
RECTANGLE	0.278	0.277	0.275	0.275
RECTANGLE^{-1}	0.362	0.277	0.369	0.369

Table A.9: c_v of the confusion coefficients using the weighted model.

S-box	First-order, no mask	First-order, LEMS	Second-order, LEMS	Second-order, 1 full mask
AES	0.082	N/A	0.085	0.083
AES^{-1}	0.070	N/A	0.071	0.071
iSCREAM	0.098	N/A	0.092	0.097
SCREAM	0.103	N/A	0.112	0.110
SCREAM^{-1}	0.137	N/A	0.122	0.133
Ascon	0.254	N/A	0.275	0.019
ICEPOLE	0.177	N/A	0.202	0.173
Ketje/Keyak	0.132	N/A	0.145	0.133
PRIMATE	0.260	N/A	0.213	0.205
PRIMATE^{-1}	0.203	N/A	0.179	0.220
Joltik	0.246	N/A	0.202	0.184
Joltik^{-1}	0.224	N/A	0.287	0.295
LAC	0.228	N/A	0.217	0.219
Minalpher	0.313	N/A	0.317	0.392
Prøst	0.326	N/A	0.268	0.244
RECTANGLE	0.326	N/A	0.360	0.295
RECTANGLE^{-1}	0.342	N/A	0.256	0.362

Table A.10: c_v of the confusion coefficients using the pairs model.