Impact of Sboxes Size upon Side Channel Resistance and Block Cipher Design

Louis Goubin[1], Ange Martinelli[1,2], and Matthieu Walle[2]

[1] Versailles Saint-Quentin-en-Yvelines University
`Louis.Goubin@uvsq.fr`
[2] Thales Communications
{`Pjean.martinelli,matthieu.walle`}`@thalesgroup.com`

Abstract. Designing a cryptographic algorithm requires to take into account various cryptanalytic threats. Since the 90's, Side Channel Analysis (SCA) has become a major threat against cryptographic algorithms embedded on physical devices. Protecting implementation of ciphers against such attacks is a very dynamic topic of research and many countermeasures have been proposed to thwart these attacks. The most common countermeasure for block cipher implementations is masking, which randomizes the variables by combining them with one or several random values. In this paper, we propose to investigate the impact of the size of the words processed by an algorithm on the security against SCA. For this matter we describe two AES-like algorithms operating respectively on 4 and 16-bit words. We then compare them with the regular AES (8 bits) both in terms of complexity and security with respect to various masking schemes. Our results show that SCA is a determinant criterion for algorithms design and that cryptographers may have various possibilities depending on their security and complexity requirements.

Keywords: Side Channel Analysis (SCA), S-boxes, Word size, Masking Countermeasure, Higher-Order SCA, AES Implementation, FPGA.

1 Introduction

When designing a block cipher, cryptographers take into account various cryptanalytic threats in order to prevent flaws in their algorithm. The most common methods are linear [19] and differential [5] cryptanalysis, interpolation [17] or related key attacks [4]. All these attacks target the mathematical primitive independently of its implementation.

In the 90's, a new kind of cryptanalysis was developed: *Side Channel Analysis* (SCA). SCA is a cryptanalytic method in which an attacker does not attack the algorithm itself, but rather its implementation. Namely, the attacker analyzes the *side channel leakage* (*e.g.* the power consumption, the electromagnetic emanations, . . .) produced during the execution of a cryptographic algorithm embedded on a physical device. SCA exploits the fact that this leakage is statistically dependent on the intermediate variables that are involved in the computation. Some of these variables are called *sensitive* in that they are related to a secret data (*e.g.* the key) and a known data (*e.g.* the plain text), and recovering information on them therefore enables efficient key recovery attacks [18, 6, 14].

However it is a hard task to improve the intrinsic security of a cryptographic algorithm against SCA. Designers can nonetheless foresee the implementation of countermeasures, and design their algorithm in order to help these implementations. As many countermeasures [1, 22, 15] use the arithmetic structure of the AES S-box, it seems a good option for designers to keep such type of structure.

To evaluate the efficiency of a countermeasure, Prouff *et al.* introduce in [23] a methodology to compute the optimal correlation between a leakage measure and a (multivariate) known variable.

They give the optimal correlation for boolean masking. We can observe that this correlation depends on the noise standard variation, the order of the masking, but also on the size of the words targeted by the attack: the longer the words, the better the security. The goal of this paper is to study the impact of the word size on both the complexity and the security of the scheme.

Related work. This paper is mainly related to three kinds of previous works. In [12, 24, 15], the authors propose countermeasures that provide a good security/complexity compromise for some security level, and propose practical results implementing their countermeasures to the AES. In [8], small scale variants of AES are designed in order to compare different cryptanalytic methods. Eventually, various optimized hardware implementations of the AES S-box can be found in [7, 20].

Our contribution. In this paper, we propose an evaluation of the impact of the word size on the security of an algorithm with respect to various masking schemes, namely boolean [22], affine [12] and polynomial masking [24, 15]. We thus define two AES-like algorithms operating respectively on 4 and 16 bits words, and discuss their implementation. Then we compare the security and the complexity of each algorithm depending on the countermeasure scheme. We finally give practical implementation results on a hardware device for equivalent level of security.

Organization of the paper. The remainder of this paper is organized as follows. In the second section we pursue a theoretical analysis about the impact of the size of the words manipulated by an algorithm upon its resistance to SCA. In section 3, we recall the AES algorithm and detail the two AES-like algorithms we have implemented. In section 4 we compare implementation costs of the three algorithms first theoretically, then on practical hardware implementations. Then section 5 details simulations results on the SCA resistance of these algorithms and the AES with respect to various masking schemes. We conclude our work in section 6.

2 Impact of Sboxes size upon Side Channel Resistance

S-boxes are the most sensitive layer with respect to the resistance of a block cipher against higher order side channel attacks. For a matter of implementation, an S-box must have a short dimension n and therefore, the input block is shared in n-bit words for the independent internal computations of the algorithm. The size of these words is determined by designers with respect to the needed properties of the algorithm and to its resistance against known cryptanalysis.

In the state of the art, we can find block ciphers manipulating 8 bits words (*e.g.* the AES [9]), 4 bits words (*e.g. Serpent* [2]) or non-square S-boxes such as those of the DES [11]. In the following of this section we investigate, for various masking schemes, the impact of this dimensions upon the resistance of a block cipher algorithm against SCA.

In what follows, we shall consider that an intermediate variable U_i is associated with a leakage variable L_i representing the information leaking about U_i through side channel. We will assume that the leakage can be expressed as a deterministic *leakage function* φ of the intermediate variable U_i with an independent additive noise B_i. Namely, we will assume that the leakage variable L_i satisfies:

$$L_i = \varphi(U_i) + B_i \ . \tag{1}$$

In the following, we call d^{th}*-order leakage* a tuple $\mathbf{L}$ of d leakage variables L_i corresponding to d different intermediate variables U_i that jointly depend on some sensitive variable. Moreover we place ourselves in the Hamming weight model, *i.e.* $\varphi = HW$.

2.1 Security against HO-DPA

In order to compare various scales of implementation with respect to various masking schemes, we compute, for each case, the optimal correlation value following the methodology described in [23] for decreasing signal-to-noise ratio (SNR). Namely we considered the value of

$$\rho_{\text{opt}} = \sqrt{\frac{\text{Var}\left[\text{E}\left[\mathcal{C}(\mathbf{L})|Z=z\right]\right]}{\text{Var}\left[\mathcal{C}(\mathbf{L})\right]}} \tag{2}$$

where Z is a sensitive variable and $\mathcal{C}$ is a combining function that converts the multivariate leakage $\mathbf{L}$ into a univariate signal. In our evaluations we have chosen $\mathcal{C}$ to be the normalized product. In [25, 12, 15], authors give equations for evaluating the value of ρ_{opt} respectively $\rho_{\text{bool-d}}$ for d-th order boolean masking, ρ_{aff} for affine masking and $\rho_{\text{polynomial}}$ for first order polynomial masking. Let us consider an gaussian noise B_i with 0 mean and standart deviation σ. We have:

$$\rho_{\text{bool-d}} = (-1)^d \frac{\sqrt{n}}{(n+4\sigma^2)^{\frac{d+1}{2}}}, \tag{3}$$

$$\rho_{\text{aff}} = \frac{n}{(4\sigma^2+n)\sqrt{2^n-1}} \tag{4}$$

and

$$\rho_{\text{polynomial}} = \sqrt{\frac{n^3 \cdot (2^{n+1} - 4^n - 1)}{\alpha_2 \cdot \sigma^4 + \alpha_1 \cdot \sigma^2 + \alpha_0}}, \tag{5}$$

where

$$\begin{aligned}
\alpha_2 &= 192 \cdot 2^n - 2^{4n+4} - 64 - 208 \cdot 4^n + 96 \cdot 8^n \\
\alpha_1 &= (40 \cdot 8^n - 64 \cdot 4^n - 8 \cdot 16^n + 32 \cdot 2^n)n^2 \\
&\quad +(88 \cdot 8^n + 128 \cdot 2^n - 2^{4n+4} - 168 \cdot 4^n - 32)n \\
\alpha_0 &= (8^n - 3 \cdot 4^n + 6 \cdot 2^n - 4)n^4 + (-4 \cdot 16^n + 14 \cdot 8^n - 16 \cdot 4^n + 2 \cdot 2^n + 4)n^3 \\
&\quad +(-4 \cdot 16^n + 23 \cdot 8^n - 44 \cdot 4^n + 34 \cdot 2^n - 8)n^2 \\
&\quad +(-3 \cdot 8^n + 10 \cdot 4^n - 9 \cdot 2^n + 2)n.
\end{aligned} \tag{6}$$

As a matter of comparaison, the optimal correlation ρ_{unmasked} for a non-masked implementation is:

$$\rho_{\text{unmasked}} = \frac{\sqrt{n}}{\sqrt{(n+4\sigma^2)}}, \tag{7}$$

We then evaluate these values for any bit size $n \in \{4, 8, 16\}$, any SNR $\in \{1, 1/2, 1/5, 1/10\}$, and for the variables targeted respectively in [25, 12, 24]:

– 1O-boolean masking, with targeted variables $(x \oplus r_1 \ ; \ r_1)$
– 2O-boolean masking, with targeted variables $(x \oplus r_1 \oplus r_2 \ ; \ r_1 \ ; \ r_2)$
– 3O-boolean masking, with targeted variables $(x \oplus r_1 \oplus r_2 \oplus r_3 \ ; \ r_1 \ ; \ r_2 \ ; \ r_3)$
– Affine masking, with targeted variables $(r2 \cdot x \oplus r_1 \ ; \ r_1)$
– 1O-polynomial masking, with targeted variables $((r_1, r3 \cdot r_1 \oplus x) \ ; \ (r_2, r3 \cdot r_2 \oplus x))$ where $r_1 \neq r_2 \neq 0$.

Note that we only consider the best available attacks against these masking. Table 1 summarizes the theoretical correlations ρ_{opt}.

Word length \SNR	$+\infty$	1	1/2	1/5	1/10
2O-DPA against 1O-boolean masking					
4-bits	0.5	0.25	0.1	0.083333	0.045455
8-bits	0.353553	0.176777	0.117851	0.058926	0.032141
16-bits	0.25	0.125	0.083333	0.041667	0.022727
3O-DPA against 2O-boolean masking					
4-bits	0.25	0.088388	0.022361	0.017010	0.006853
8-bits	0.125	0.044194	0.024056	0.008505	0.003426
16-bits	0.0625	0.022097	0.012028	0.004253	0.001713
4O-DPA against 3O-boolean masking					
4-bits	0.125	0.031250	0.005	0.003472	0.001033
8-bits	0.044194	0.011049	0.004910	0.001228	0.000365
16-bits	0.015625	0.003906	0.001736	0.000434	0.000129
2O-DPA against Affine masking [12]					
4-bits	0.258199	0.129099	0.015188	0.009931	0.002556
8-bits	0.062622	0.020874	0.006958	0.001228	0.000312
16-bits	0.003906	0.000781	0.000230	0.000039	0.000010
2O-DPA against 1O-polynomial masking [15]					
4-bits	0.030589	0.023984	0.015187	0.013612	0.009063
8-bits	0.001854	0.001473	0.001243	0.000876	0.000607
16-bits	0.0000074	0.0000060	0.0000051	0.0000037	0.0000027

Table 1. Theoretical correlation values

We can state that the security of each scheme evolves in different ways when the word size increases. Indeed the value of the optimal correlation for boolean masking decreases polynomialy in n, whereas it decreases exponentially for both affine and polynomial masking. Intuitively, this can be explained seeing that, when using boolean masking, every bit of the mask operates on a single bit of the sensitive variable. Thus the security overhead of a larger bit size is roughly linear in the word length. In the case of affine and polynomial maskings, the relation between the targeted values and the sensitive one is much more complex, which highly improves the security when increasing the word length.

Moreover, in the case of boolean masking, the optimal correlation coefficient decrease exponentially in the order d. In order to compare the different implementations of (higher order) boolean masking we represent in Figure 1 the correlation value for various amount of noise. As expected, we can state that a higher order scheme provides a better security in (almost) all cases. Notably $(d+1)$-th order boolean masking applied to the 4-bits AES variation provides better security than d-th order masking applied to the regular 8-bits AES.

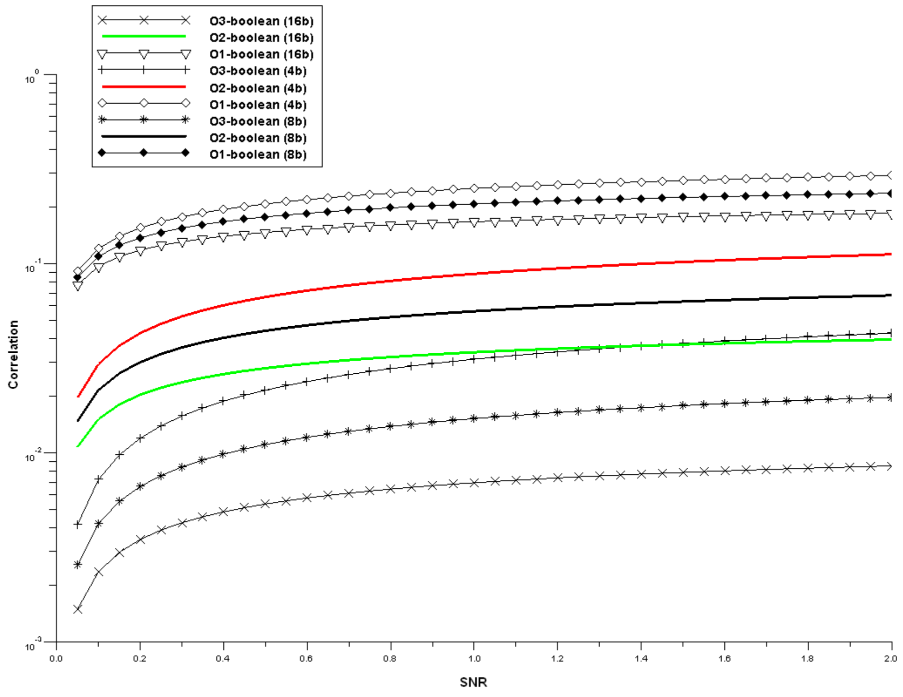

Fig. 1. Correlation value of boolean maskings with respect to SNR

2.2 Information theoretic analysis

The analysis of the optimal correlation allows us to evaluate precisely the security of a countermeasure against CPA but does not give any general informations independently of the chosen distinguisher. In [26] the authors propose to evaluate the mutual information between the leakage vector $\mathbf{L}$ and the sensitive variable Z in order to compute the theoretic leakage induced by the computation. Nevertheless, this metric does not give any direct information of the complexity of an attack but only gives the amount of information leaked during the computation. We can then efficiently compare two countermeasures implemented on the same algorithm in terms of security against SCA, but the comparison between two distinct algorithms does not seems to be so relevant, especially when the word sizes are distinct.

A third security analysis can be the practical attack simulation but it needs the definition of a complete algorithm. Such an analysis is the topic of section 5.

In this section we have shown that non-linear masking techniques applied to large S-boxes (typically 16 bits) provides the best theoratical security among the considered countermeasures. We may wonder what is the practical complexity of the implementation of such countermeasures. In the following, we evaluate the complexity of some implementations in order to emphasize the most interesting one in terms of complexity for a given security level.

3 Design of the Sboxes

The main goal of this article is to evaluate the optimal word size to implement countermeasures against SCA. In order to achieve this goal, we define variants of the AES using different word sizes. For matters of simplicity, we focus on powers of 2. After recalling the processing of the AES, we propose in this section two AES-like algorithms working on respectively 4 and 16 bit words. Both are operating on 128-bit blocks.

3.1 The Advanced Encryption Standard

The Advanced Encryption Standard (AES) is a Substitution-Permutation Network (SPN) introduced by J. Daemen and V. Rijmen in [9]. It iterates 10 times (for the 128-bit version) a transformation involving four steps : `AddRoundKey`, `ShiftRows`, `MixColumn`, and `SubByte`. Details about these steps can be founded in appendix A. In particular the AES S-box is designed as

$$Sb_8[x] = Q(x) + a(x) \cdot P(x) \bmod [X^8 + 1]$$

where $a(x)$ is the inverse of x in the field $\mathbb{F}_{2^8}$, and P and Q are polynomials chosen such that it ensure a complicated algebraic expression when combined with the inverse mapping, and that there is no fixed points ($Sb_8[x] = x$) and no 'opposite fixed points' ($Sb_8[x] = \bar{x}$). This construction ensures a good resistance against linear and differential cryptanalysis. Following the formalism introduced in [10], the AES sbox achieve a prop-ratio and an input-output correlation respectively equal to 2^{-6} and 2^{-3}.

Remark 1. The inversion of $a \in \mathbb{F}_{2^8}$ as described in [22] can be implemented using 4 multiplications, 7 squares and 1 refreshMask operation.

3.2 4-bit variation

Let $\mathbb{F}_{2^4} = \mathbb{F}_2[x]/(x^4 + x + 1)$.

We define the 4-bit AES-like Sbox as follows :

$$Sb_4[x] = Q(x) + a(x) \cdot P(x) \bmod [X^4 + 1]$$

where $a(x)$ is the inverse of x in the field $\mathbb{F}_{2^4}$, and P and Q are polynomials chosen according to [9] such that : $P(x) = x^3 + x + 1$ and $Q(x) = x^3 + x^2 + 1$.

A look-up table for the Sbox Sb_4 is given (notation in hexadecimal) in table 2.

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	D	6	B	9	5	C	F	4	2	A	E	8	7	3	1	0

Table 2. Sbox over $\mathbb{F}_{2^4}$

Remark 2. The inversion of $a \in \mathbb{F}_{2^4}$ can be computed as $a^{-1} = a^{14} = (a \cdot a^2)^4 \cdot a^2$. It can then be implemented using only two multiplications and 3 squares.

Resistance against known attacks : In order to evaluate the security of the S-box against differential and linear cryptanalysis, we have to compute respectively the prop-ratio and the input-output correlation (see appendix D). We can then evaluate the length of an efficient linear or differential trail and adapt the number of round adequately.

- prop-ratio : 2^{-2}
- input-output correlation : 2^{-1}

Keeping the original `ShiftRows` and `MixColumn` operating on 8-bits words, we obtain no 12-round differential trail with a predicted prop ratio above 2^{-150} (which is sufficient for the 128-bit block length), and no 12-round linear trail with a correlation above 2^{-75}. In this case, in order to keep an equivalent security, we have to extend the round number to 30. Moreover this construction is not directly compatible with every masking schemes operating on 4 bits.

In order to bypass this incompatibility and to optimize the complexity of the overall cipher, we define a diffusion layer composed of a MixColumns operation designed using a 8×8 MDS matrix over $\mathbb{F}_{2^4}$ (see appendix C), combined with a ShiftRows operation as designed for the AES. We then consider the internal state as a 8×4 matrix over $\mathbb{F}_{2^4}$. The branch number is thus 9. For example the matrix involved in the MixColumns operation can be chosen as a circulant matrix with first line equal to:

```
[1 1 2 1 3 4 2 3].
```

We can deduce that there is no 4-round differential trail with a predicted prop ratio above 2^{-98} and no 4-round linear trail with a correlation above 2^{-49}. The round number can thus be fixed to 15 without any security loss.

However, in [8] the authors design simplified version of AES in order to try their security against algebraic attacks. For the 4-bit version the succed only with a small number of round and using a sub-optimal linear layer. Our construction clearly ensure a much better security against such attacks.

3.3 16-bit variation

Let $\mathbb{F}_{2^{16}} = \mathbb{F}_2[x]/(x^{16} + x^{13} + x^{10} + x^9 + x^2 + x + 1)$.

We define the 16 bit AES-like S-box as following :

$$Sb_{16}[x] = Q(x) + a(x) \cdot P(x) \bmod [X^{16} + 1]$$

where $a(x)$ is the inverse of x in the field $\mathbb{F}_{2^{16}}$, and P and Q are polynomials chosen according to [9] such that : $P(x) = x^{15} + x^8 + x^3 + x + 1$ and $Q(x) = x^{15} + x^9 + x^8 + x^7 + x^2 + x + 1$.

Remark 3. The inversion of $a \in \mathbb{F}_{2^{16}}$ can be computed following:

$$\begin{aligned} b &= (a^2.a)^2.a = a^7 \\ c &= \quad b^8.b \quad = a^{63} \\ c &= \quad c^{64}.c \quad = a^{4095} \\ c &= \quad c^{16}.b^2 \quad = a^{65534} \end{aligned}$$

The inversion can then be implemented using only 5 multiplications and 16 squares.

Resistance against known attacks : As previously, we compute respectively the prop-ratio and the input-output correlation and evaluate the length of an efficient linear or differential trail and adapt the number of rounds adequately.

– Sb_{16} prop-ratio : 2^{-14}.
– Sb_{16} input-output correlation : 2^{-7}.

As for the 4 bit case, the original linear layer of the AES is not compatible with every masking schemes operating on 16 bits. A good alternative is to use a 8×8 circulant MDS matrix instead of the `ShiftRows` and `MixColumn` operations. Such a matrix can even be optimized allowing the Hamming weight of each of its component to be 1 and thus leads to the most optimized overall design for this size of words. For example such a matrix can be chosen as a circulant matrix with first line equal to (in hexadecimal):

```
[0001 0001 0020 0001 0100 0400 0200 0040].
```

The branch number of this linear layer is maximal (*i.e.* equal to 9), ensuring that there is no 1-round differential trail with a predicted prop ratio above 2^{-125}, and no 1-round linear trail with a correlation above 2^{-63}. As for the 4 bit case, the round number can thus be lower to 6 rounds without any security loss.

4 Complexity

Previously, we have seen that increasing the word size improves the security of a device against SCA. However, this security improvement should not lead to an unreasonable cost. In this section, we compare several implementations to study the impact of S-boxes sizes on the complexity. Firstly in a theoretical manner, then by doing a comparative analysis of hardware implementations.

4.1 Overall complexity

For each masking scheme, we consider the AES variations described in Section 3 (see appendix B for details about multiplication implementations) :

– 4-bit words (using look-up table),
– 8-bit words (using log/alog tables),
– 16-bit words (using log/alog tables),
– 16-bit words (using tower fields method)

The evolution of the theoretical complexity of each masking scheme according to the word size is given in Table 3.

Remark 4. Each implementation of the affine masking is optimised using the most appropriate variation of the scheme : that is the reference implementation for the original AES and the 4-bit variation, and the least memory expensive variation for the 16-bit (see [12] for details about each variation). Similarly the implementation of the polynomial masking is made using the straightforward adaptation of [3] as explained in [24, 16].

Implementation	XORs/ANDs/shifts	Table look-ups	Random bits	RAM (bits)	ROM (bits)
1-st order Boolean Masking					
4-bit	8580	7920	5760	284	192
8-bit	17640	16144	16896	312	6128
16-bit (log/alog)	7704	9273	13056	368	2097120
16-bit (tower field)	40269	50385	13056	368	1022
2-nd order Boolean Masking					
4-bit	14340	14760	15360	328	192
8-bit	37800	32272	46080	352	6128
16-bit (log/alog)	16200	16257	36086	448	2097120
16-bit (tower field)	72549	90273	36086	448	1022
3-rd order Boolean Masking					
4-bit	26820	23520	28800	328	192
8-bit	65640	54160	87552	400	6128
16-bit (log/alog)	25656	25257	69120	544	2097120
16-bit (tower field)	114429	141969	69120	544	1022
Affine Masking					
4-bit	2176	1224	2400	448	1088
8 bits	3424	1840	1552	4392	8176
16 bits (log/alog)	526560	394456	800	1048912	3145696
16 bits (tower field)	2500080	1971288	800	1048912	1022
1^{st} order polynomial Masking					
4 bits	9480	19440	3840	328	192
8 bits	58560	65824	27792	400	6128
16 bits (log/alog)	39840	57568	18592	544	2097120
16 bits (tower field)	321360	409856	18592	544	1022

Table 3. Theoretical Complexity of cipher implementations

The 8-bit affine masking appears to be a very good option both in terms of security and complexity. The complexity of the 4-bit variation is not as low as it can be expected because of its heavy linear layer. Using a similar S-box in a Feistel scheme could solve this probleme though, but such a construction is not in the scope of this paper. With respect to boolean masking, we can state that, for a high amount of noise, the 4-bit variation provides a very low complexity for a good security level. For instance, with a SNR near 1, the third order boolean masking implemented on a 4-bit algorithm provides a better complexity and a better security than a second order boolean masking implemented on a 8-bit algorithm. The 16-bit variation does not seem an interesting choice because of its huge memory requirements. However the very high security provided by polynomial masking on this variation may justify its implementation on very low restricted devices.

4.2 Complexity of chosen hardware implementations

The theoretical complexities given in the previous section provide a good overview of the implementations' security. However we want to evaluate the practical feasability of some chosen implementations on hardware devices. As boolean masking is the most widely implemented scheme, we limited ourselves to implement Rivain and Prouff's scheme from [22] at orders 1, 2 and 3.

We developed in VHDL a small system on chip (SoC) embedding a simple serial interface and a 128-bits masked AES implementation running in ECB mode. The implementations are fully parallelized, notably all Sboxes are processed simultaneously. As proposed in [22], the multiplicative inverse is computed using a d-order secure square-and-multiply algorithm. To do so, each Sbox encompasses a secure multiplier as well as a square operator, both working on d shares. Then, alternating the square and the multiply module in a sequential way ensure to use the minimal area.

The SoC is built on an Altera Cyclone III EP3C25 (24,624 Logic Elements, Speed grade -7, -8) with no particular optimization technique and an automated place-and-route stage. The resulting maximal clock frequency is 125 Mhz for all implementations. Following this fully parallelized design, no protected version of the 16-bits scheme can be realistically implemented on the SoC.

We implemented multiplier and square blocks for GF(2^8) and GF(2^4) in the same way, that is fully combinatorial with input/output register. In that case, a secure multiplication takes 3 cycles since some variables have to be processed at different time as explained in [22], squaring is done in 2 clock cycles and only one cycle is needed to refresh the masks. Eventually, the linear layer is also combinatorial so the total number of cycles to process the whole round is equal to the the number of cycles required for the inversion.

Word Size	Global System (LEs)	S-box (LEs)	SecMult (LEs)	Clock cycles	Throughput (MB/s)
1-st order Boolean Masking					
4 bits	4350	112	54	197	18.87
8 bits	8089	380	212	282	7.09
2-nd order Boolean Masking					
4 bits	7500	207	117	197	18.87
8 bits	13435	690	487	282	7.09
3-rd order Boolean Masking					
4 bits	11300	350	212	197	18.87
8 bits	21299	1170	870	282	7.09

Table 4. Implementations areas (in logic elements) and performances

For the 8-bit version, we have 4 secMult + 7 Square + 2 refreshMask $= 4 \times 3 + 7 \times 2 + 2 = 28$ cycles per round. Since there are 10 rounds, we obtain 280 cycles for the encryption + 2 cycles to handle the I/O, hence 282 cycles in total.

For the 4-bit version, we have 2 secMult + 3 Square + 1 refreshMask $= 2 \times 3 + 3 \times 2 + 1 = 13$ cycles per round. Since there are 15 rounds, we obtain 195 cycles for the encryption + 2 cycles to handle the I/O, hence 197 cycles in total.

As expected, the resulting 8-bit S-box is at least 3 times bigger than the 4-bit version for a given order of masking. The interesting fact is that this inequality still hold for different order of masking : a d-order 8-bit S-box is *bigger* than a $(d+1)$-order 4-bit Sbox. Now if we look at the theoretical correlation of each of this implementations (see Table 1), we observe that any $(d+1)$-order 4-bit AES is *more* secure the d-order 8-bit version. Notably the 4-bit 2^{nd}-order AES is more secure *and* smaller than the regular 8-bit AES using only one mask.

As a matter of fact, we can observe that between the two implementations, the difference of size of the global circuit is not so important for 1-st order masking but increases with the order. It can be explained by noticing that the expensive layer for the 8-bit scheme is clearly the S-box (and in particular the SecMult operation), while it is the permutation layer in the case of the 4 bit variation. Indeed, in this case, the cost of the S-box is roughly 4 times lower than the one operating on 8 bits. Moreover the cost of the S-box transformation is quadratic in d while the linear layer is only linear in d, and so the difference increases with the order.

By taking into account the Sbox size during the design of an implementation, it is possible to improve the security of the device without increasing the size of the circuit excessively. Actually the linear layer may take a non-negligible place. Indeed, in order to avoid to increase excessively the number of round, this layer has to be improved. This leads to a bigger linear layer and the global system size increases. Anyway, the $(d+1)$-order 4-bit variation is wholly more secure and smaller and faster than the d-order 8-bit variation.

5 Attack simulations

To confirm the theoretical analysis conducted in section 2, we performed several attacks simulations. Formally we applied several side-channel distinguishers to simulated leakages. The leakage measurements have been simulated as samples of the random variables L_i defined according to equation (1) with $\varphi = \mathrm{HW}$ and $B_i \sim \mathcal{N}(0, \sigma^2)$ where the different B_i's are assumed mutually independent. For all the attacks, the sensitive variable Z was chosen to be an S-box output of the targeted algorithm of the form $\mathsf{S}(M \oplus k^\star)$ where M represents a varying plaintext byte and $k^\star$ represents the key byte to recover.

Side-channel distinguishers. We applied two kind of side-channel distinguishers: higher-order DPA such as described in section 2.1 and higher-order MIA [21, 13]. In a HO-MIA, the distinguisher is the mutual information: the guess k is tested by estimating $\mathrm{I}(\hat{\varphi}(Z(k)); \mathbf{L})$. As mutual information is a multivariate operator, this approach does not involve a combining function.

Targeted variables. Each attack was applied against leakage values associated to boolean masking, affine masking and polynomial masking. The target variables are those listed in section 2.1 where $x = \mathsf{S}(X \oplus k^\star)$:

Prediction functions. For each DPA, we choose $\hat{\varphi}$ to be the optimal prediction function :

$$\hat{\varphi} : z \mapsto \mathrm{E}\left[\mathcal{C}(\mathbf{L}) | Z = z\right]. \tag{8}$$

This leads us to select the Hamming weight function in the attacks against both $1O$-polynomial masking and dO-boolean masking and the Dirac function δ_0 for the affine masking.

For the MIA attacks, we choose $\hat{\varphi}$ such that it maximizes the mutual information $\mathrm{I}(\hat{\varphi}(Z(k)); \mathbf{L})$ for $k = k^\star$ while ensuring that the mutual information is lower for $k \neq k^\star$. In our case, every HO-MIA against both polynomial and Boolean masking is performed with $\hat{\varphi} = \mathrm{HW}$ since the distribution of $(\mathrm{HW}(Z \oplus m_0), \mathrm{HW}(m_0))$ (resp. $(\mathrm{HW}(Z \oplus a_0 \cdot x_0, x_0), \mathrm{HW}(Z \oplus a_0 \cdot x_1, x_1))$) only depends on $\mathrm{HW}(Z)$. Therefore

$$\mathrm{I}\big(Z; (\mathrm{HW}(Z \oplus m_0), \mathrm{HW}(m_0))\big) = \mathrm{I}\big(\mathrm{HW}(Z); (\mathrm{HW}(Z \oplus m_0), \mathrm{HW}(m_0))\big).$$

Note that the same relation holds at every masking order. Every HO-MIA against affine masking is performed using $\hat{\varphi} = \delta_0$ since the distribution of the leakage functions is identically distributed for any $Z \neq 0$, and is only remarkable for $Z = 0$ [12].

Pdf estimation method. For the (HO-)MIA attacks, we use the histogram estimation method with rule of [14] for the *bin-widths* selection.

Attack simulation results. Each attack simulation is performed 100 times for various SNR values ($+\infty$, 1, 1/2, 1/5 and 1/10). Table 5 summarizes the number of leakage measurements required to observe a success rate of 90% in retrieving $k^\star$ for the different attacks.

Word size \ SNR	$+\infty$	1	1/2	1/5	1/10
2O-CPA against 1O-boolean masking					
4-bit	70	160	400	1 800	13 000
8-bit	150	500	1 500	6 000	20 000
16-bit	400	1 400	2 000	10 000	25 000
2O-MIA against 1O-boolean masking					
4-bit	80	1 000	5 000	10 000	33 000
8-bit	100	5 000	15 000	50 000	160 000
3O-CPA against 2O-boolean masking					
4-bit	370	1 700	20 000	50 000	300 000
8-bit	1 500	9000	35 000	280 000	$> 10^6$
16-bit	6 500	20 000	85 000	900 000	$> 10^6$
3O-MIA against 2O-boolean masking					
4-bit	120	10 000	200 000	800 000	$> 10^6$
8-bit	160	160 000	650 000	$> 10^6$	$> 10^6$
2O-CPA against affine masking					
4-bit	300	1400	20 000	100 000	400 000
8-bit	6500	20 000	45 000	170 000	650 000
16-bit	55 000	200 000	800 000	$> 10^6$	$> 10^6$
2O-MIA against affine masking					
4-bit	270	10 000	100 000	800 000	$> 10^6$
8-bit	5500	100 000	600 000	$> 10^6$	$> 10^6$
2O-CPA against 1O-polynomial masking					
4-bit	15 000	40 000	100 000	150 000	250 000
8-bit	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
16-bit	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
2O-MIA against 1O-polynomial masking					
4-bit	100 000	300 000	600 000	$> 10^6$	$> 10^6$
8-bit	500 000	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$

Table 5. Number of leakage measurements for a 90% success rate against 4, 8 and 16-bits algorithms

Remark 5. No MIA processed against an implementation of the 16-bits algorithm had succeeded. This can be explained by the complexity of estimation of the probability density functions needed by the attack.

The simulation results confirm the security intuition introduced in section 2 that the security of an algorithm is highly dependant of its word size. We can indeed state that the number of measurements needed for a 90% success rate increase with the word size. In particular these results show that the security improvement induced by boolean masking on longer words increase more slowly than that induced by non-linear masking scheme. Moreover we are able to give practical results for the efficiency of MIA upon the considred implementations. These results shows that the security improvement of a longer word size has the same kind of impact on both CPA and MIA.

6 Conclusion

In this paper, we investigated the influence of the size of the words in an cryptographic algorithm on the efficiency and the security of the scheme against side channel analysis. We designed for this matter two algorithms operating respectively on 4 and 16-bit words, and compared them to the original 8-bits AES both in terms of complexity and SCA resistance.

The 16-bit variation provides a very good security, particularly assiciated with a non-linear masking, but the complexity overhead is consequent. On the contrary, we have shown that in some situations, using smaller Sboxes associated with higher order masking technique improves the security of a device with almost no extra cost. Our results show that indeed, a 2^{nd} order boolean masking applied on the 4-bits AES provides both a better resistance as well as better performances than 1^{st} order boolean masking applied on the 8-bit AES.

The S-boxes size and the masking order can be viewed as two complementary parameters. By choosing these parameters, one can adapt the performances (area, thoughput, security) of a device to match a specific need. Table 6 recall implementations complexities and the corresponding CPA simulation results for a realistic amount of noise (SNR= 1/2).

CPA (traces)	Global System (LEs)	S-box (LEs)	Clock cycles	Throughput (MB/s)
8-bits AES secured by 1-st order Boolean Masking				
1 500	8089	380	282	7.09
4-bits AES secured by 2-nd order Boolean Masking				
20 000	7500	207	197	18.87

Table 6. Comparison of two distinct implementations

References

1. Mehdi-Laurent Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.
2. Ross Anderson, Eli Biham, and Lars Knudsen. *Serpent: A Proposal for the Advanced Encryption Standard*, 1998.
3. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM, 1988.
4. Eli Biham. New types of cryptanalytic attacks using related keys. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 398–409. Springer-Verlag New York, Inc., 1993.
5. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
6. É. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
7. David Canright. A Very Compact S-Box for AES. In J.R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.
8. C. Cid, S. Murphy, and M.J.B. Robshaw. Small scale variants of the aes. In *FSE*, Lecture Notes in Computer Science. Springer, 2005.
9. J. Daemen and V. Rijmen. *AES Proposal: Rijndael*, September 1999.
10. Joan Daemen. *Cipher and hash function design strategies based on linear and differential cryptanalysis.* PhD thesis, K.U.Leuven, 1995.

11. FIPS PUB 46-3. *Data Encryption Standard (DES)*. National Institute of Standards and Technology, October 1999.
12. Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*. Springer, 2010.
13. Benedikt Gierlichs, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis. Cryptology ePrint Archive, Report 2009/228, 2009. http://eprint.iacr.org/.
14. 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, 2008.
15. Louis Goubin and Ange Martinelli. Protecting aes with shamir's secret sharing scheme. In *Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science. Springer, 2011.
16. Louis Goubin and Ange Martinelli. Protecting aes with shamir's secret sharing scheme - extended version. *IACR Cryptology ePrint Archive*, 2011:516, 2011.
17. Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In E. Biham, editor, *Fast Software Encryption – FSE '97*, volume 1367 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 1997.
18. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M.J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
19. M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.
20. Sumio Morioka and Akashi Satoh. An optimized s-box circuit architecture for low power aes design. In *Cryptographic Hardware and Embedded Systems*, CHES '02, 2002.
21. Emmanuel Prouff and Matthieu Rivain. Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security – ANCS 2009*, volume 5536 of *Lecture Notes in Computer Science*, pages 499–518. Springer, 2009.
22. Emmanuel Prouff and Matthieu Rivain. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and Franois-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.
23. Emmanuel Prouff, Matthieu Rivain, and Régis Bévan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Trans. Comput.*, 58(6):799–811, 2009.
24. Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the aes using secure multi-party computation protocols. In *Cryptographic Hardware and Embedded Systems – CHES 2011*, Lecture Notes in Computer Science. Springer, 2011.
25. Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.
26. François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.
27. Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. An ASIC Implementation of the AES SBoxes. In B. Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2002.

A Original AES Steps

In this section, we recall the four main operations involved in each round of the AES encryption Algorithm. For each of them, we denote by $\mathbf{s} = (s_{i,j})_{0 \le i,j \le 3}$ the 16-byte state at the input of the transformation, and by $\mathbf{s}' = (s'_{i,j})_{0 \le i,j \le 3}$ the state at the output of the transformation.

1. AddRoundKey: Let $\mathbf{k} = (k_{i,j})_{0 \le i,j \le 3}$ denote the round key. Each byte of the state is XOR-ed with the corresponding round key byte:

$$(s'_{i,j}) \leftarrow (s_{i,j}) \oplus (k_{i,j}).$$

2. $\mathsf{SubBytes}$: each byte of the state passes through the 8-bit S-box S:

$$s'_{i,j} \leftarrow \mathsf{S}(s_{i,j}).$$

For all x in $\mathrm{GF}(2^8)$, the AES S-box is defined as follows :

$$S[x] = Q(x) + a(x) \cdot P(x) \bmod [X^8 + 1]$$

where $a(x)$ is the inversion function in the field $\mathrm{GF}(2^8)$, $P(x) = x^7 + x^6 + x^5 + x^4 + 1$ coprime to the modulus, and $Q(x) = x^7 + x^6 + x^2 + x$ chosen such that the S-box has no fixed points ($S(x) = x$) and no "opposite fixed point" ($S(x) = \bar{x}$).

3. $\mathsf{ShiftRows}$: each row of the state is cyclically shifted by a certain offset:

$$s'_{i,j} \leftarrow s_{i,j-i \bmod 4}.$$

4. $\mathsf{MixColumns}$: each column of the state is modified as follows:

$$(s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}) \leftarrow \mathsf{MixColumns}_c(s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c})$$

where $\mathsf{MixColumns}_c$ implements the following operations:

$$\begin{cases} s'_{0,c} \leftarrow (\mathsf{02} \cdot s_{0,c}) \oplus (\mathsf{03} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} \leftarrow s_{0,c} \oplus (\mathsf{02} \cdot s_{1,c}) \oplus (\mathsf{03} \cdot s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} \leftarrow s_{0,c} \oplus s_{1,c} \oplus (\mathsf{02} \cdot s_{2,c}) \oplus (\mathsf{03} \cdot s_{3,c}) \\ s'_{3,c} \leftarrow (\mathsf{03} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\mathsf{02} \cdot s_{3,c}), \end{cases}$$

where $\cdot$ and $\oplus$ respectively denote the multiplication and the addition in the field $\mathrm{GF}(2)[X]/p(X)$ with $p(X) = X^8 + X^4 + X^3 + X + 1$, and where $\mathsf{02}$ and $\mathsf{03}$ respectively denote the elements X and $X + 1$.

B Implementations of the field multiplication

The main problem encountered when implementing AES is the implementation of the field multiplication. In this section we will view several possibilities of implementations with respect to the words bit-size.

B.1 4-bit Multiplication :

In the case of a 4-bit implementation, a natural idea is to pre-compute the field multiplications and store them in a 16×16 entries table. The multiplication is then resumed to 1 table look-up. Such table can be stored on 128 bytes.

B.2 8-bit Multiplication :

A classical method to implement the multiplication over GF(256) in software is to use *log/alog* tables. These tables are constructed using the fact that all non-zero elements in a finite field $\mathrm{GF}(2^n)$ can be obtained by exponentiation of a generator in this field. For a generator α of

GF(256)* we define $\log(\alpha^i) = i$ and $\mathrm{alog}(i) = \alpha^i$. This results are stored in two tables of $2^n - 1$ words of n bits.

If a, b are non-zero, then the product $a \cdot b$ can be computed using log/alog tables as

$$a \cdot b = \mathrm{alog}[(\log(a) + \log(b)) \bmod (2^n - 1)]. \tag{9}$$

With this representation, computing a product over GF(256) requires 3 table look-ups, and two additions modulo 256. Both tables can be stored in ROM on 510 bytes.

On hardware systems, the multiplication can easily be implemented using composite field method using the methodology given in [27], or simple combinatorial multipliers.

B.3 16-bits Multiplication :

In order to compute multiplication over GF(2^{16}), two tools can be used: log/alog tables or the tower field method (see [27]).
Using log/alog table requires 3 table look-ups, and two additions mod(2^{16}). Both tables can be stored in ROM on 262140 bytes.
For more memory-restricted implementations, the tower field methodology can be applied. It consists in considering GF(2^{16}) as GF(2^8) $\times$ GF(2^8), thus making product in the smaller field GF(2^8).

In [27], Wolkerstorfer *et al.* give an efficient hardware implementation of multiplications and inversions in GF(2^8). They represent GF(2^8) as a quadratic extension of the field GF(2^4) then exhibit an isomorphism between GF(2^8) and GF(2^4) $\times$ GF(2^4). The multiplication can thus be implemented in GF(2^4) $\times$ GF(2^4), instead of GF(2^8). We want to develop here the same methodology in order to implement the multiplication in GF(2^{16}) using multiplication in GF(2^8).

Let α be the class of X in GF(2^{16}). Let $Q(X) = X^2 + X + \alpha^7$ be an irreducible polynomial over GF(2^8). Let us consider the field GF(2^8) $\times$ GF(2^8) = GF(2^8)$[X]/Q(X)$. If β is the class of X in GF(2^8)^2, then every element of GF(2^8)^2 can be written as $a \cdot \beta + b$ with a and b in GF(2^8).

Let η and ν be two elements of GF(2^8)^2 such that $\eta = u_1\beta + u_0$ and $\nu = v_1\beta + v_0$. Then we have :

$$\begin{aligned} \eta \cdot \nu &= (u_1\beta + u_0)(v_1\beta + v_0) \\ &= u_1v_1\beta^2 + (u_1v_0 + u_0v_1)\beta + u_0v_0 \\ &= (u_1v_0 + u_0v_1 + u_1v_1)\beta + (u_1v_1\alpha^7 + u_0v_0) \end{aligned} \tag{10}$$

Hence the product in GF(2^8)^2 can be performed using 5 multiplications in GF(2^8) and 3 XORs. In order to compute the isomorphism $I : \mathrm{GF}(2^{16}) \longrightarrow \mathrm{GF}(2^8) \times \mathrm{GF}(2^8)$ and its inverse, we simply have to define base changing equations from the relation $I(\alpha) = \mathtt{2A}\beta + \mathtt{1C}$. Base changing can then be computed following algorithm 1.

Algorithm 1 Base changing

INPUT: An element a in the input field F, $(\mu_0, \ldots, \mu_{15})$ the base changing value

OUTPUT: The corresponding element a' in the ouput field G

```
1. $a' \leftarrow 0$
2. for $i = 0$ to 15 do
```

```
3.    $a' \leftarrow a' \oplus (a_i \cdot \mu_i)$
4. return $a'$
```

where a_i is the i^{th} bit of a.

Remark 6. As both words in GF(2^8) depend on every 16 bits of the input, there is no security loss in this implementation.

Using this method, each multiplication in GF(2^8) $\times$ GF(2^8) can be performed using 5 multiplications in GF(2^8) (using log/alog tables) and 3 XORs. Both isomorphisms I and I^{-1} can be computed using 16 XORs and 16 ANDs (and 16 shifts in software) knowing both 32-bytes tables of base changing.

C Linear Layer and MDS Matrix

We have seen that the linear layer of the AES is composed of two operations: `ShiftRows` and `MixColumn`. This linear layer allows a reasonable diffusion entwined with a very low complexity. However we can define optimal diffusion function using MDS matrices as follows.

Let `C` be an (m, k, d)-error correcting code over $\mathbb{F}_{2^n}$. Then m is the word size of `C`, k is its dimension, and d is the minimal distance between two words of the code (or the minimal weight of a non-zero word of the code). Let us have the following definition:

Definition 1 (MDS code). *`C` is said MDS (Maximum Distance Separable) if it is a linear code that reaches the Singleton bound, i.e. if and only if $d = m - k + 1$.*

An important parameter in the security of a block cipher against linear and differential cryptanalysis is its *branch number B*.

Let b be the linear (respectively differential) bias (see section D) associated to a transformation $S : \mathrm{GF}(q) \to \mathrm{GF}(q)$ of the substitution layer, then the global resistance provided by N rounds of the algorithm can be evaluated by

$$b^{B \cdot \frac{N}{2}}.$$

Let θ be a diffusion layer with input k elements of GF(q) and with output m elements of GF(q) then

$$\theta : \left| \begin{array}{ccc} \mathrm{GF}(q)^k & \to & \mathrm{GF}(q)^m \\ x & \mapsto & \theta(x). \end{array} \right.$$

Then θ's branch number is given by

$$\mathcal{B}(\theta) = \min_{\xi \in (\mathrm{GF}(q)^k)^\star} \{\omega(\xi) + \omega(\theta(\xi))\}.$$

Proposition 1. *We have $\mathcal{B}(\theta) \leq m + 1$.*

Let now `C` be a $(2k, k, k+1)$-MDS code over $\mathbb{F}_{2^n}$ with generator matrix $G = (I \parallel M)$ with I the identity and $I, M \in \mathcal{M}_{k \times k}(\mathbb{F}_{2^n})$. M is then called an MDS matrix. Let us have the following proposition:

Proposition 2. *Let M be an MDS matrix over $\mathbb{F}_{2^n}$. We can then define an optimal, i.e having the maximal branch number, invertible SPN-diffusion layer θ_C as*

$$\theta_C : \left| \begin{array}{l} \mathbb{F}_{2^n}^k \to \mathbb{F}_{2^n}^k \\ x \;\mapsto Mx. \end{array} \right.$$

In this case, the branch number of the linear layer is maximal, and equal to $k + 1$.

D Linear and Differential cryptanalysis

Differential and Linear cryptanalysis were first described respectively by Eli Biham and Adi Shamir [5] in 1991 and by Mitsuru Matsui [19] in 1993. Both attacks aims to recover the last round's subkey of the algorithm by exploiting statistical bias in the propagation of the message through the algorithm called linear or differential trails. The efficiency of these attacks depends of the length of the trails, *i.e.* the round number. Basically, the round number can be derived from the branch number of the linear layer and both the prop ratio and the input-output correlation of the S-boxes [10].

In practice we evaluate the security of an S-box S against differential cryptanalysis by computing the prop-ratio R_S. Let (a', b') be a pair where a' is a difference of input values and b' the difference of the corresponding outputs. The prop-ratio $R_S(a', b')$ of S associated to (a', b') is:

$$R_S(a', b') = 2^{-n} \sum_a \delta(b' \oplus S(a \oplus a') \oplus S(a)) \tag{11}$$

where δ is the Dirac function.

Similarly, we can evaluate the security of S against linear cryptanalysis by computing its input-output correlation. Let (a', b') be an input-output pair, then the correlation $c_S(a', b')$ of S associated to (a', b') is:

$$c_S(a', b') = 2 \cdot \mathrm{p}_X \left[a' \cdot S(x) = b' \cdot x \right] - 1 \tag{12}$$

Formally, for a cipher operating on n bits blocks to be resistant against Differential Cryptanalysis, it is a necessary condition that there is no differential trail with a predicted prop ratio higher than 2^{1-n}.
Similarly, to be resistant against Linear Cryptanalysis, it is a necessary condition that there is no linear trail with a input-output correlation coefficient higher than $2^{n/2}$.