Provably Secure S-Box Implementation Based on Fourier Transform

Emmanuel Prouff[1], Christophe Giraud[2] and Sébastien Aumônier[1]

[1] Oberthur Card Systems,
71-73, rue des Hautes Pâtures, 92 726, Nanterre, France.
{e.prouff,s.aumonier}@oberthurcs.com
[2] Oberthur Card Systems,
4, allée du doyen Georges Brus, 33 600, Pessac, France.
c.giraud@oberthurcs.com

Abstract. Cryptographic algorithms implemented in embedded devices must withstand Side Channel Attacks such as the Differential Power Analysis (DPA). A common method of protecting symmetric cryptographic implementations against DPA is to use masking techniques. However, clever masking of non-linear parts such as S-Boxes is difficult and has been the flaw of many countermeasures. In this article, we take advantage of some remarkable properties of the Fourier Transform to propose a new method to thwart DPA on the implementation of every S-Box. After introducing criteria so that an implementation is qualified as DPA-resistant, we prove the security of our scheme. Finally, we apply the method to FOX and AES S-Boxes and we show in the latter case that the resulting implementation is one of the most efficient.

Keywords: Differential Power Analysis, Provably Secure Countermeasure, Fourier Transform, Symmetric Cryptosystems, S-Box, AES, FOX.

1 Introduction

In 1996, Kocher introduced the concept of *Side Channel Analysis* which utilizes side channel leakage of embedded devices such as timing execution to obtain Information about sensitive data [18]. This concept was pushed one step further in [17]. In this paper, Kocher *et al.* use power consumption measurements of the device during the execution of sensitive operations, allowing two kinds of Power Attacks: the *Simple Power Analysis* (SPA) and the *Differential Power Analysis* (DPA). The first attack consists in directly interpreting power consumption measurements and the second attack also involves statistical tests. From then on, many papers describing either countermeasures or attack improvements have been published (see [1,4,6,21] for example).

In the case of symmetric cryptosystems such as DES [11] and AES [10], the most critical part when securing implementations against DPA is to protect their non-linear operations (*i.e.* the calls to the *S-Boxes*). Indeed, all the other operations are more or less linear and can be protected in a straightforward manner (see [1] for instance). To protect the calculus of the output of an S-Box against DPA, three main kinds of methods have been proposed in the literature. The first one, called the *duplication method* [6,13], consists in randomly splitting every piece of sensitive data in a constant number of blocks. Then, the computation can be securely carried out by performing calculations with these random blocks. The second method, called the *re-computation method* [1,2,29], involves a re-computation of the lookup tables corresponding to the S-Box with one or several random value(s) which must be changed each time the algorithm is executed. The third generic method, that we call here *S-Box secure calculation*, has been essentially applied to protect AES implementations [5,15,28,32] due to the strong algebraic structure of the AES S-Box. In this case, S-Box outputs are not directly obtained by accessing a lookup table but are computed by using a mathematical representation of the S-Box. All the logical operations involved during this calculation are resistant to DPA.

In this paper, we present a new, secure and generic S-Box calculation method based on the discrete Fourier transform. In Section 2, we formalize DPA attacks on S-Boxes and we introduce a model allowing us to measure the efficiency of such attacks. We also exhibit criteria so that an implementation is qualified as DPA-resistant. In Section 3, we briefly present the Fourier transform and we use its properties to introduce a new S-Box secure calculation. We then prove that our method is DPA-resistant in accordance with the criteria established in Section 2. In Section 4, we apply our method to AES and FOX and we compare its efficiency with other existing countermeasures.

2 On the Notion of DPA-resistant Implementation of S-Boxes

S-Boxes aim to ensure confusion of Information in many symmetric cryptosystems. Since they manipulate sensitive data, their implementation in embedded devices must withstand side-channel cryptanalysis such as DPA. During the last decade, several ways of securely implementing S-Boxes have been proposed. For some of them, no proof of resistance to DPA has been established and sometimes Information about the secret is recovered. In such cases, there is a need to quantify the relevance of the leaked Information from the attacker's point of view. In the following, we introduce a new notion called *Advantage* which allows us to measure how much DPA on the S-Box implementation impacts the security of the whole embedded cryptosystem. Even if we focus on block cipher algorithms, our study is valid for every symmetric cryptosystem involving S-Boxes.

A block cipher is the iteration of several *rounds*, each round involving S-Boxes. The rounds are parameterized by *round-keys* which are derived from a secret parameter usually called *master key*. A round-key RK can be viewed as an uplet of small vectors, called *sub-keys*, which are used separately by the S-Boxes. In the following, we denote by n the bit-length of the sub-keys and by N the bit-length of the round-keys.

In a well-designed block cipher algorithm, recovering a sub-key K must be as difficult as recovering the whole round-key RK by brute force attack. An implementation of such an algorithm is said to be *secure* if this fundamental property is not only satisfied by the algorithm but also by its implementation.

When an S-Box implementation thwarts DPA, recovering the sub-key by attacking the S-Box calculus is as difficult as recovering the round-key itself (and thus requires around 2^N suppositions). In this case, the security of the cryptosystem is not impacted by DPA on the S-Boxes implementations.

On the contrary, when the S-Box implementation has some drawbacks from a DPA point of view, some Information about the sub-key is obtained. In this case, the efficiency of the attack depends on the amount of Information on K which has leaked:

- in the best case, the attack allows the attacker to get K completely.
- in less favorable situations, the attacker does not recover the sub-key directly, but some useful Information on it is obtained (for instance the Hamming weight of the sub-key or a linear relation which must be satisfied by some of its bit-coordinates [19]).

Let RK denote the round-key an attacker tries to recover by DPA. By performing a DPA for every S-Box which manipulates a sub-key extracted from RK, the attacker succeeds in isolating the round key in a proper subset of the *key-space*. Therefore, the attacker does not need to test all the N-bit round-keys but only a subset of them to recover RK. Depending on the efficiencies of the localized DPAs, the cardinality ε of this subset ranges from 1 (the best case from the attacker viewpoint) to 2^N (when all the attacks failed).

To compare the efficiency of different countermeasures against DPA, we introduce the notion of *Advantage* which aims to evaluate the attacker's capacity to recover a secret round-key RK manipulated partially by one or several S-Box(es), each of them being implemented through a method $\mathcal{M}$.

Definition 1. *Let RK be a N-bit secret value. The* Advantage of an adversary *in recovering the secret RK by DPA is the value* Adv *defined by:*

$$\mathrm{Adv}(\mathcal{M}) = 2^N - \varepsilon \ , \tag{1}$$

where ε denotes the cardinality of the subset of $\mathbb{F}_2^N$ containing all the candidate round-keys isolated by DPA.

When DPA allows the attacker to unambiguously recover all the sub-keys, the whole round-key is straightforwardly deduced and ε equals 1. On the contrary, when power consumption measurements give no Information on K, then ε equals 2^N. We deduce $0 \leq Adv(\mathcal{M}) \leq 2^N - 1$.

Proposition 1. *An S-Box implementation $\mathcal{M}$ is such that* $\mathrm{Adv}(\mathcal{M}) = 0$ *if and only if $\mathcal{M}$ thwarts DPA. Such an implementation is said to be* DPA-resistant*.*

A first step in securing an S-Box implementation against DPA consists in *masking* the sensitive data manipulated at input and at output of the S-Box calculation. This is usually performed by securely adding random values to these data [6]. Then, while the S-Box computation is performed with the masked input, other operations must be involved (in parallel or as pre-computations) allowing the so-called *mask-correction* and the introduction of a new mask for the output. Let `op` denote either the bitwise-addition or a modular addition and let F denote the S-Box, an S-Box secure calculation $\mathcal{M}$ can be viewed as a process allowing to get the pair $(F(X) \ \mathtt{op} \ R_2, R_2)$ from the input pair $(X \ \mathtt{op} \ R_1, R_1)$. So, a generic solution to securely perform an S-Box calculation can be depicted by the following generic procedure:

Procedure 1 S-Box calculation

INPUTS: A random value R_1, a masked value $\widetilde{X} = X \ \mathtt{op} \ R_1$ (with X a sensitive data), a function F representing the S-Box, a method $\mathcal{M}$
OUTPUT: The pair $(F(X) \ \mathtt{op} \ R_2, R_2)$ with R_2 a random value

1. Generate a random value R_2
2. Compute $result \leftarrow \mathcal{M}(\widetilde{X}, R_1, R_2, F)$ $[result = F(X) \ \mathtt{op} \ R_2]$
3. **Return** $(result, R_2)$

Remark 1. We assume throughout this paper that the distribution of the generated random values is uniform which is a prerequisite for security.

The cost of the mask-correction is usually not negligible compared to the one of the entire block-cipher algorithm. Indeed, as it can be seen in [25] or [31] for instance, it induces a very high (timing and/or memory) overhead, especially because one must always ensure that the sensitive data are securely manipulated and that the mask-correction algorithm itself thwarts DPA.

For an S-Box calculation as depicted in Procedure 1, the method $\mathcal{M}$ is DPA-resistant if the computation of $F(X)$ `op` R_2 from X `op` R_1, R_1 and R_2 is performed without revealing any useful Information for DPA. In order to design such a method, random values independent of the input X are usually involved when manipulating sensitive data during the calculation of the output stored in *result*.

By using and adapting ideas of [3], we decompose $\mathcal{M}$ into d steps at the *unit level* [3]. The intermediate results of the d steps are denoted by $I_1(X, Z_1), \cdots, I_d(X, Z_d)$, where X is the sensitive input of the S-Box and where Z_i denotes an uplet of random variables involved to securely manipulate the i-th intermediate data.

Let us assume that a method $\mathcal{M}$ is such that every intermediate value $I_i(X, Z_i)$ is independent of the sensitive input X. Then, the power consumption resulting from the manipulation of the values $I_i(X, Z_i)$ gives no Information on X. Based on this remark, we introduce a proposition characterizing the methods $\mathcal{M}$ for which $Adv(\mathcal{M}) = 0$:

Proposition 2. *Let d denote the number of steps at the unit level of an S-Box implementation $\mathcal{M}$. Then,* $\mathrm{Adv}(\mathcal{M})$ *is null if and only if for every $i = 1, \cdots, d$, the random variables X and $I_i(X, Z_i)$ are independent.*

Remark 2. The *Mutual Information* can be used to formalize the notion of independency between two variables (see [22] for instance).

In the particular case of the AES algorithm, some S-Box implementations have been proved to be DPA-resistant [3,23]. Nevertheless, as they rely on the algebraic structure of the AES S-Box, they are not generic. In the next section, we present a new DPA-resistant method which can be applied to every S-Box.

---

[3] In hardware terms, this level is based on the contents of registers.

3 A New Method to Protect S-Boxes Access from DPA

In this section we firstly recall some basics about the Fourier transform. Then, we exhibit an interesting property of this function from which a new S-Box secure calculation method is deduced. Finally, we prove that the corresponding implementation is DPA-resistant, *i.e.* the Advantage of an adversary over this implementation is equal to zero.

3.1 Fourier Transform

Let us recall the definition of the Fourier transform of a function defined from an abelian group G into $\mathbb{C}$.

Definition 2. *Let G be an abelian group and let $\widehat{G}$ denote the dual space of G. Let $\mathbb{C}[G]$ denote the set of applications from G into $\mathbb{C}$. Then, the* Fourier transform *on $\mathbb{C}[G]$, denoted by $\mathcal{F}$, is defined by:*

$$\begin{aligned} \mathcal{F} : \mathbb{C}[G] &\to \mathbb{C}[\widehat{G}] \\ F &\mapsto \widehat{F} \end{aligned} \quad , \tag{2}$$

where $\widehat{F}$ is defined by:

$$\forall \chi \in \widehat{G},\ \widehat{F}(\chi) = \sum_{X \in G} F(X)\chi(X) \quad . \tag{3}$$

In this paper, we use the Fourier transform in the particular case $G = \mathbb{F}_2^n$. When G is a n-dimensional vector space over $\mathbb{F}_2$, its dual group $\widehat{G}$ is the set of characters $\chi_A : X \mapsto (-1)^{A \cdot X}$, where $\cdot$ denotes the scalar product defined by $A \cdot X = \sum_{i \in \{0,\ldots,n-1\}} A_i \cdot X_i \mod 2$. So, if $G = \mathbb{F}_2^n$ then Relation (3) is equivalent to:

$$\forall \chi_A \in \widehat{G},\ \widehat{F}(\chi_A) = \sum_{X \in G} F(X)(-1)^{A \cdot X} \quad . \tag{4}$$

The Fourier transform of a function F defined on $\mathbb{F}_2^n$ satisfies $F = \frac{1}{2^n}\widehat{\widehat{F}}$, that is:

$$\forall X \in \mathbb{F}_2^n,\ F(X) = \frac{1}{2^n} \sum_{\chi_A \in \widehat{G}} \widehat{F}(\chi_A)(-1)^{A \cdot X} \quad . \tag{5}$$

For simplicity reasons, the value $\widehat{F}(\chi_A)$ is denoted by $\widehat{F}(A)$ and the summation in Relation (5) is computed for $A \in G$.

3.2 DPA-resistant Implementation of S-Boxes Access

The new method. Relation (5) is the starting point of our study about how to find a new solution to protect S-Box access from DPA. From this relation and the involutive property of the Fourier transform, we observe that the image of a message X through a function F can be computed from a masked message $\widetilde{X}$ and the corresponding mask.

Let X, R_1 and A be three elements of $\mathbb{F}_2^n$ and let $\widetilde{X}$ denote the vector $X \oplus R_1$. As $A \cdot X \oplus \widetilde{X} \cdot R_1$ equals $A \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus A)$, one can re-write Relation (5) as follows:

$$(-1)^{\widetilde{X} \cdot R_1} F(X) = \frac{1}{2^n} \sum_{A \in \mathbb{F}_2^n} \widehat{F}(A)(-1)^{A \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus A)} \ . \tag{6}$$

The relation above is the core of our solution. When F denotes an S-Box, it provides a way to compute $\pm F(X)$ from a boolean masked input $\widetilde{X}$. The most remarkable fact in Relation (6) is that the mask-correction is performed *on-the-fly* during the computation of $F(X)$. The induced overhead is negligible compared to the whole calculus and the simplicity of the mask-correction operations makes it easy to evaluate the DPA-resistance of the method with the model introduced in Section 2. However, a direct implementation of this relation is not secure since the output and some intermediate results are unmasked. In particular, as $R_1 \cdot \widetilde{X}$ equals $R_1 \cdot \overline{X}$ (where $\overline{X}$ denotes the two-complement of X), the scalar multiplication $R_1 \cdot \widetilde{X}$ has a flaw when X equals the all-one vector (see [12] for the description of an attack exploiting such a flaw). To circumvent this default, the variable $\widetilde{X}$ must be masked by a random value R_2 independent of R_1.

In the following, we present a modified version of Relation (6) whose straightforward implementation is DPA-resistant (as proved in Section 3.3):

$$\begin{aligned}(-1)^{(\widetilde{X} \oplus R_2) \cdot R_1} F(X) + R_3 = \\ \left\lfloor \tfrac{1}{2^n} \left(R' + \textstyle\sum_{A \in \mathbb{F}_2^n} \widehat{F}(A)(-1)^{A \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus A \oplus R_2)} \right) \right\rfloor \ ,\end{aligned} \tag{7}$$

where $R' = 2^n R_3 + R_4$ with $R_3, R_4 \in \mathbb{F}_2^n$. The $2n$-bit vector R' is used to mask the intermediate results of the summation.

Implementation Aspects. In the following, we denote by SP a function which computes $(-1)^{X \cdot Y}$ from a couple (X, Y) and by $AM2BM$ a procedure which transforms an arithmetic masking into a boolean masking:

$$AM2BM : (sign, sign \times X + R, R) \mapsto X \oplus R \ , \tag{8}$$

where $sign = \pm 1$.

From Relation (7), we deduce the following algorithm which computes the boolean masked output of an S-Box from a boolean masked input:

Algorithm 1 Computation of a boolean masked S-Box output from a boolean masked input

INPUTS: A masked input $\tilde{X} = X \oplus R_1$, the input mask R_1 and a lookup table $\widehat{F}$
OUTPUT: The couple $(F(X) \oplus R_3, R_3)$

1. Pick up three n-bit random R_2, R_3 and R_4
2. $result \leftarrow 2^n R_3 + R_4$
3. **for** A **from** 0 **to** $2^n - 1$ **do**
4. $\quad T_1 \leftarrow \mathrm{SP}(A, \widetilde{X})$ $\quad [T_1 = (-1)^{A \cdot \widetilde{X}}]$
5. $\quad T_2 \leftarrow \widetilde{X} \oplus A$ $\quad [T_2 = \tilde{X} \oplus A]$
6. $\quad T_2 \leftarrow T_2 \oplus R_2$ $\quad [T_2 = \tilde{X} \oplus A \oplus R_2]$
7. $\quad T_2 \leftarrow \mathrm{SP}(R_1, T_2)$ $\quad [T_2 = (-1)^{R_1 \cdot (\widetilde{X} \oplus A \oplus R_2)}]$
8. $\quad T_2 \leftarrow T_1 \times T_2$ $\quad [T_2 = (-1)^{A \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus A \oplus R_2)}]$
9. $\quad T_2 \leftarrow T_2 \times \widehat{F}(A)$ $\quad [T_2 = \widehat{F}(A)(-1)^{A \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus A \oplus R_2)}]$
10. $\quad result \leftarrow result + T_2$ $\quad [result = 2^n R_3 + R_4 + \sum_{i \in \{0, A\}} \widehat{F}(i)(-1)^{i \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus i \oplus R_2)}]$
11. $result \leftarrow result >> n$ $\quad [result = (-1)^{(\widetilde{X} \oplus R_2) \cdot R_1} F(X) + R_3]$
12. $T_1 \leftarrow \widetilde{X} \oplus R_2$ $\quad [T_1 = \tilde{X} \oplus R_2]$
13. $T_1 \leftarrow \mathrm{SP}(T_1, R_1)$ $\quad [T_1 = (-1)^{(\widetilde{X} \oplus R_2) \cdot R_1}]$
14. $result \leftarrow \mathrm{AM2BM}(T_1, result, R_3)$ $\quad [result = F(X) \oplus R_3]$
15. **Return** $(result, R_3)$

In Algorithm 1, the lookup table $\widehat{F}$ is always accessed 2^n times in a way which is independent of the input X. The values X and R_1 only impact the combination of the values $\widehat{F}(A)$.

Random values R_3 and R_4 aim at masking the content of the buffer *result*. Before the right-shift operation of Step 11, the least significant half part of $result$ contains the value R_4. After Step 11, the content of $result$ equals the value $(-1)^{(\widetilde{X} \oplus R_2) \cdot R_1} F(X) + R_3$ left-padded with zeros.

Computation performed in Step 14 (*cf.* Relation (8)) is essentially a transformation of an arithmetic masking into a boolean masking. Goubin [14] and Coron *et al.* [7] proposed DPA-resistant implementations of such a computation. To implement AM2BM we use a slightly modified version of Goubin's method which outputs $(X \oplus R, R)$ from $(X - R, R)$. To take into account the $sign$ parameter, we use Goubin's algorithm with $(sign \times (sign \times X + R), -sign \times R)$ as input.

Efficiency of Algorithm 1 is strongly related to the dimension n of the S-Box since the lookup table $\widehat{F}$ contains 2^n signed integers belonging to $[-2^{2n}; 2^{2n}]$ and is accessed 2^n times. For $n = 8$, $\widehat{F}$ requires at most 544 bytes of ROM (which is reduced to 512 bytes if all the values $\widehat{F}(A)$ are even, which is often the case for cryptographic functions F) and it is accessed 256 times for each execution of Algorithm 1. The overhead becomes significantly smaller when $n = 4$: in this case only 16 access to the 18-byte lookup table $\widehat{F}$ are required (if all the values $\widehat{F}(A)$ are even, $\widehat{F}$ can be stored over 16 bytes).

3.3 Security Analysis

In this section we analyse the security of Algorithm 1. From Proposition 2, Algorithm 1 is DPA-resistant if and only if all the intermediate values $I_i(X, Z_i)$ are independent of the input X.

We do not focus on Step 14 since Goubin shows in [14, §4.3] that all the intermediate values that appear during the execution of his algorithm are independent of the input.

In Table 1, we list the different sensitive intermediate results $I_i(X, Z_i)$ which appear during the execution of Algorithm 1. The values which only depend on the loop counter or on a random value are obviously omitted:

Step i	Instruction	Intermediate results $I_i(X, Z_i)$	Z_i
4.1	$reg \leftarrow \widetilde{X}$	$\widetilde{X}$	R_1
4.2	$T_1 \leftarrow \mathrm{SP}(A, \widetilde{X})$	$(-1)^{A \cdot \widetilde{X}}$	R_1
5	$T_2 \leftarrow \widetilde{X} \oplus A$	$\widetilde{X} \oplus A$	R_1
6	$T_2 \leftarrow T_2 \oplus R_2$	$\widetilde{X} \oplus A \oplus R_2$	(R_1, R_2)
7	$T_2 \leftarrow \mathrm{SP}(R_1, T_2)$	$(-1)^{R_1 \cdot (\widetilde{X} \oplus A \oplus R_2)}$	(R_1, R_2)
8	$T_2 \leftarrow T_1 \times T_2$	$(-1)^{A \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus A \oplus R_2)}$	(R_1, R_2)
9	$T_2 \leftarrow T_2 \times \widehat{F}(A)$	$\widehat{F}(A)(-1)^{A \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus A \oplus R_2)}$	(R_1, R_2)
10	$result \leftarrow result + T_2$	$2^n R_3 + R_4$ $+ \sum_i \widehat{F}(i)(-1)^{i \cdot \widetilde{X} \oplus R_1 \cdot (\widetilde{X} \oplus i \oplus R_2)}$	(R_1, R_2, R_3, R_4)
11	$result \leftarrow result >> n$	$(-1)^{(\widetilde{X} \oplus R_2) \cdot R_1} F(X) + R_3$	(R_1, R_2, R_3)
12	$T_1 \leftarrow \widetilde{X} \oplus R_2$	$\widetilde{X} \oplus R_2$	(R_1, R_2)
13	$T_1 \leftarrow \mathrm{SP}(T_1, R_1)$	$(-1)^{(\widetilde{X} \oplus R_2) \cdot R_1}$	(R_1, R_2)

Table 1. The different sensitive values manipulated during Algorithm 1.

To establish the independency of these intermediate values $I_i(X, Z_i)$ with X, we use the following lemma:

Lemma 1. *Let $\alpha \in \mathbb{F}_2^n$ be arbitrary and let β be uniformly distributed over $\mathbb{F}_2^n$ and independent of α. The variable $\alpha \oplus \beta$ is uniformly distributed and independent of α. The same holds for $(-1)^{\alpha \oplus \beta}$ if $n = 1$.*

The proof of this lemma is straightforward and therefore omitted.

The sensitive values $I_i(X, Z_i)$ can be divided into two groups:

1. the ones which are masked by adding or by XORing a random value: $\widetilde{X}$, $\widetilde{X} \oplus A$, $\widetilde{X} \oplus A \oplus R_2$, $\widetilde{X} \oplus R_2$ and *result* in Steps 10 and 11,
2. the other values: $(-1)^{A\cdot\widetilde{X}}$, $(-1)^{(\widetilde{X}\oplus R_2)\cdot R_1}$, $(-1)^{R_1\cdot(\widetilde{X}\oplus A\oplus R_2)}$, $(-1)^{A\cdot\widetilde{X}\oplus R_1\cdot(\widetilde{X}\oplus A\oplus R_2)}$ and $(-1)^{A\cdot\widetilde{X}\oplus R_1\cdot(\widetilde{X}\oplus A\oplus R_2)}\widehat{F}(A)$.

The values belonging to the first group have a boolean or an arithmetic mask which is chosen uniformly at random, so it is obvious that they are independent of the input.

Now, let us analyse the values belonging to the second group:

- In the case $A = 0$, the variable $(-1)^{A\cdot\widetilde{X}} = (-1)^{A\cdot(X\oplus R_1)}$ is always equal to 1 and so it is independent of X. When $A \neq 0$, the independency of the variables $(-1)^{A\cdot X\oplus A\cdot R_1}$ and $A \cdot X$ is established by applying Lemma 1 to $\alpha = A \cdot X$ and $\beta = A \cdot R_1$. Since the variable X only appears in the term $A \cdot X$, one deduces that $(-1)^{A\cdot X\oplus A\cdot R_1}$ is independent of X.
- By noticing that $(\widetilde{X}\oplus R_2)\cdot R_1 = \overline{X}\cdot R_1\oplus R_2\cdot R_1$, one deduces in a similar way from Lemma 1 that variables $(-1)^{(\widetilde{X}\oplus R_2)\cdot R_1}$ and $(-1)^{R_1\cdot(\widetilde{X}\oplus A\oplus R_2)}$ are independent of X.
- As $A\cdot\widetilde{X}+R_1\cdot(\widetilde{X}\oplus A\oplus R_2)$ equals $A\cdot X\oplus\overline{X}\cdot R_1\oplus R_2\cdot R_1$, Lemma 1 implies that the variables $(-1)^{A\cdot\widetilde{X}+R_1\cdot(\widetilde{X}\oplus A\oplus R_2)}$ and X are independent. The same conclusion holds for $(-1)^{A\cdot\widetilde{X}+R_1\cdot(\widetilde{X}\oplus A\oplus R_2)}\widehat{F}(A)$.

We proved above that all the values $I_i(X, Z_i)$ manipulated during the execution of Algorithm 1 are independent of the input X. From Proposition 2, we thus deduce that our method is DPA-resistant, *i.e.* that its Advantage is null.

In the next section, we apply our method to protect S-Boxes access of AES and FOX. In the first case, we compare its performances with the ones of two other well-known countermeasures.

4 Applications

4.1 DPA-resistant AES Implementation

Before the final choice for the Advanced Encryption Standard (AES) [10], several papers had investigated the security of the AES candidates against side-channel attacks, especially DPA [5,9,20]. Since 2000, many countermeasures have been proposed to counteract DPA on AES[4].

[4] A survey of the proposed countermeasures is done in [8].

To counteract DPA, Kocher proposed in [18] a very simple and generic solution which can be applied to protect an AES implementation. It consists in using the lookup table F^* defined by $X \mapsto F[X \oplus R_1] \oplus R_2$, where R_1 and R_2 are two random values generated for each new execution of the algorithm. The main drawback of this solution is the large amount of RAM required to store F^*. Indeed, this kind of memory is very limited on embedded devices.

Another method called *Transformed Masking Method* (TMM) has been presented in [1]. However, it has a weakness when computing the AES S-Box (*cf.* [12]). In order to fix this flaw, several papers have been published (*cf.* [12,23] for example).

In the two methods above, the AES S-Box, which performs an inversion in $\mathbb{F}_{2^8}$ with 0 being mapped to 0, is implemented through a lookup table. Rijmen presented in [27] an alternative idea which essentially consists in using efficient combinational logic. In this approach, each element a of $\mathbb{F}_{2^8}$ is represented as a linear polynomial $a_h x + a_l$ over $\mathbb{F}_{2^4}$. The inversion of such a polynomial can be computed as follows when it is different from zero: $(a_h x + a_l)^{-1} = a'_h x + a'_l$ where $a'_h = a_h \times d^{-1}$ and $a'_l = (a_h + a_l) \times d^{-1}$ with $d = (a_h^2 \times \{e\}) + (a_h \times a_l) + a_l^2$ and with $\{e\}$ denoting the hexadecimal value 0x0E (*cf.* [26, §3.3]).

Rijmen's remark has been used in [24,23,32,30] to fix the flaw of TMM when accessing S-Box: the so-called *Tower Field Methods* perform the inversion in $\mathbb{F}_{2^8}$ by using masked multiplications and masked inversions in $\mathbb{F}_{2^4}$ or $\mathbb{F}_{2^2}$.

In the following algorithm, we present a new way to implement the AES S-Box based on the method presented in Section 3. As this method is much faster in $\mathbb{F}_{2^{n/2}}$ than in $\mathbb{F}_{2^n}$, we use Rijmen's remark to perform the computations in $\mathbb{F}_{2^4}$ instead of $\mathbb{F}_{2^8}$. We denote by $\widehat{Inv}_{\mathbb{F}_{2^4}}$ the Fourier transform of the inverse over $\mathbb{F}_{2^4}$ where the element 0 is mapped to itself, and by *map* the isomorphism defined in [26] which takes an element a of $\mathbb{F}_{2^8}$ as input and outputs the coefficients of the corresponding linear polynomial $a_h x + a_l$ over $\mathbb{F}_{2^4}$.

Algorithm 2 Inversion of a masked element $\widetilde{a} = a \oplus m_a$ in $\mathbb{F}_{2^8}$

INPUTS: $(\widetilde{a} = a \oplus m_a, m_a) \in {\mathbb{F}_{2^8}}^2$
OUTPUT: $(\widetilde{a^{-1}} = a^{-1} \oplus m'_a, m'_a)$

---

1. Pick up three 4-bit random m_d, m'_h and m'_l
2. $(m_h, m_l) \in \mathbb{F}_{2^4}^2 \leftarrow map(m_a)$
3. $(\widetilde{a_h}, \widetilde{a_l}) \in \mathbb{F}_{2^4}^2 \leftarrow map(\widetilde{a})$ $\quad [(\widetilde{a_h}, \widetilde{a_l}) = (a_h \oplus m_h, a_l \oplus m_l)]$
4. $\widetilde{d} \leftarrow \widetilde{a_h}^2 \otimes \{e\} \oplus \widetilde{a_h} \otimes \widetilde{a_l} \oplus \widetilde{a_l}^2 \oplus m_d \oplus \widetilde{a_h} \otimes m_l$ $\quad [\widetilde{d} = d \oplus m_d]$

$\oplus \widetilde{a_l} \otimes m_h \oplus m_h^2 \otimes \{e\} \oplus m_l^2 \oplus m_h \otimes m_l$

5. $(\widetilde{d^{-1}}, m_{d^{-1}}) \leftarrow$ Algorithm 1$(\widetilde{d}, m_d, \widetilde{Inv_{\mathbb{F}_{2^4}}})$ $[\widetilde{d^{-1}} = d^{-1} \oplus m_{d^{-1}}]$

6. $\widetilde{a'_h} \leftarrow \widetilde{a_h} \otimes \widetilde{d^{-1}} \oplus m'_h \oplus m_h \otimes \widetilde{d^{-1}} \oplus m_{d^{-1}} \otimes \widetilde{a_h} \oplus m_{d^{-1}} \otimes m_h$ $[\widetilde{a'_h} = a'_h \oplus m'_h]$

7. $\widetilde{a'_l} \leftarrow \widetilde{a_l} \otimes \widetilde{d^{-1}} \oplus m'_l \oplus \widetilde{a'_h} \oplus \widetilde{d^{-1}} \otimes m_l \oplus \widetilde{a_l} \otimes m_{d^{-1}} \oplus m'_h \oplus m_l \otimes m_{d^{-1}}$ $[\widetilde{a'_l} = a'_l \oplus m'_l]$

8. $m'_a \leftarrow map^{-1}(m'_h, m'_l)$

9. $\widetilde{a^{-1}} \leftarrow map^{-1}(\widetilde{a'_h}, \widetilde{a'_l})$ $[\widetilde{a^{-1}} = a^{-1} \oplus m'_a]$

10. **Return** $(\widetilde{a^{-1}}, m'_a)$

Steps 1 to 4 and Steps 6 to 9 have been proved to be DPA-resistant in [23].

In the following table, we compare our method applied to AES with two other countermeasures. The three implementations use boolean masking of the intermediate results except when accessing S-Boxes where we use:

- Algorithm 2,
- Oswald *et al.*'s method [23,24]. It only differs from Algorithm 2 in its approach to compute the inversion of $\widetilde{d} \in \mathbb{F}_{16}$. In [23,24], the inversion is performed by going down to $\mathbb{F}_4$ and its complexity approximatively equals the one of Algorithm 2 excluding the 5$^{\text{th}}$ Step which is replaced by a square operation (since the inversion operation in $\mathbb{F}_4$ is equivalent to squaring),
- Trichina *et al.*'s method [31] which uses log- and alog-tables.[5]

The timings were obtained with a CPU running at 8 MHz.

Method	Timings (ms)	RAM (bytes)	ROM (bytes)
Straightforward implementation	5	32	1150
This paper (Algo. 2)	32	39	3100
Oswald *et al.* [23,24]	26	42	3400
Trichina *et al.* [31]	21	291	3050

Table 2. Comparison of several methods to protect AES against DPA.

To test the DPA-resistance of our method in practice, we mount a DPA attack on the implementation described above. The results are given in Appendix A.

4.2 DPA-resistant FOX Implementation

In [16], Junod and Vaudenay introduce a new family of block ciphers called FOX. The non-linear part of a FOX-algorithm is ensured by an

[5] Trichina *et al.*'s method seems to have a flaw with regard to the Zero Value Attack (*cf.* [25]). Thus, its DPA-resistance is not well-established yet.

S-Box S. It consists in a Laï-Massey scheme with three rounds taking three different small S-Boxes as round functions; these functions, denoted by S_1, S_2 and S_3, operate on 4-bit words.

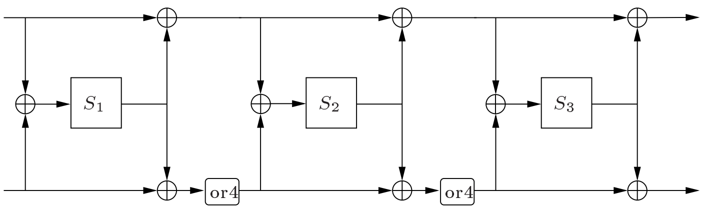

Fig. 1. Structure of the S-Box of FOX

In Figure 1, the `or4` operation consists in a single round of a 4-bit Feistel scheme with the identity function as round function: for every $X = (x_0, x_1, x_2, x_3) \in \mathbb{F}_2^4$, we have $\texttt{or4}(X) = (x_2, x_3, x_0 \oplus x_2, x_1 \oplus x_3)$.

For every n-bit vector X, let us denote by X^l and X^r the two $\frac{n}{2}$-bit vectors such that $X = X^l || X^r$. To thwart DPA attack during the S-Box calculations, we perform the following algorithm which inherits the security of Algorithm 1:

Algorithm 3 Secure computation of FOX S-Box

INPUTS: $\widetilde{X} = X \oplus R_1$ and R_1 in $\mathbb{F}_2^8$
OUTPUT: $S(X) \oplus R_2$ and R_2, where R_2 is a random vector

$\mathrm{T}_1 \leftarrow \widetilde{X}^l$; $\mathrm{T}_2 \leftarrow \widetilde{X}^r$; $\mathrm{T}_3 \leftarrow R_1^l$; $\mathrm{T}_4 \leftarrow R_1^r$
for i **from** 1 **to** 3 **do**
  $(\widetilde{\text{result}}, \text{mask}) \leftarrow$ **Algorithm 1**$(\mathrm{T}_1 \oplus \mathrm{T}_2, \mathrm{T}_3 \oplus \mathrm{T}_4, \widehat{S}_i)$
  $\mathrm{T}_2 \leftarrow \widetilde{\text{result}} \oplus \mathrm{T}_2$
  $\mathrm{T}_4 \leftarrow \text{mask} \oplus \mathrm{T}_4$
  if $i \neq 3$ **then**
    $\mathrm{T}_1 \leftarrow$ **or4**$(\widetilde{\text{result}} \oplus \mathrm{T}_1)$; $\mathrm{T}_3 \leftarrow$ **or4**$(\text{mask} \oplus \mathrm{T}_3)$
  else
    $\mathrm{T}_1 \leftarrow \widetilde{\text{result}} \oplus \mathrm{T}_1$; $\mathrm{T}_3 \leftarrow \text{mask} \oplus \mathrm{T}_3$
$\widetilde{\text{result}} \leftarrow (\mathrm{T}_1 << 4) \oplus \mathrm{T}_2$; $\text{mask} \leftarrow (\mathrm{T}_3 << 4) \oplus \mathrm{T}_4$
Return $(\widetilde{\text{result}}, \text{mask})$

Because the S-Boxes S_i of FOX operate on 4-bit vectors, computing their outputs by use of Algorithm 1 only implies 16 lookup table's access for each S_i. It is possible to check that this overhead is much smaller than the overhead induced by previous S-Box secure calculation methods.

5 Conclusion and Perspectives

In this paper, we describe a new and generic method based on the Fourier transform to obtain DPA-resistant S-Box implementations. After introducing a security model to resist DPA, we prove the resistance of our proposal. Since our method does not rely on specific S-Box properties, it can be applied to any symmetric cryptosystem. It is very efficient when the S-Box is applied to small fields such as FOX's or when the computations can be performed in vector spaces of small dimensions. In particular, we apply our method to AES and we evaluate in practice the efficiency and the resistance of the corresponding implementation.

This work raises two interesting open problems. The first one is to upgrade our security model and our method to take into account high-order DPA attacks. The second one is to find other transformations or operators which allow us to compute a masked output of an S-Box from a masked input, without revealing information on the sensitive data.

References

1. M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In *CHES 2001*, vol. 2162 of *LNCS*, pages 309–318. Springer, 2001.
2. M.-L. Akkar and L. Goubin. A Generic Protection against High-Order Differential Power Analysis. In *FSE 2003*, vol. 2887 of *LNCS*, pages 192–205. Springer, 2003.
3. J. Blömer, J. G. Merchan, and V. Krummel. Provably Secure Masking of AES. In *SAC 2004*, vol. 3357 of *LNCS*, pages 69–83. Springer, 2004.
4. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *CHES 2004*, vol. 3156 of *LNCS*, pages 16–29. Springer, 2004.
5. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards. In *AES 2*, March 1999.
6. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO '99*, vol. 1666 of *LNCS*, pages 398–412. Springer, 1999.
7. J.-S. Coron and A. Tchulkine. A New Algorithm for Switching from Arithmetic to Boolean Masking. In *CHES 2003*, vol. 2779 of *LNCS*, pages 89–97. Springer, 2003.
8. N. Courtois and L. Goubin. An Algebraic Masking Method to Protect Against Power Attacks. In *ICISC 2005*, vol. 3935 of *LNCS*. Springer, 2006.
9. J. Daemen and V. Rijmen. Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals. In *AES 2*, March 1999.
10. FIPS PUB 197. *Advanced Encryption Standard.* National Institute of Standards and Technology, 2001.
11. FIPS PUB 46. *The Data Encryption Standard.* National Bureau of Standards, January 1977.
12. J. Golić and C. Tymen. Multiplicative Masking and Power Analysis of AES. In *CHES 2002*, vol. 2523 of *LNCS*, pages 198–212. Springer, 2002.

13. L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In *CHES '99*, vol. 1717 of *LNCS*, pages 158–172. Springer, 1999.
14. L. Goubin. A Sound Method for Switching between Boolean and Arithmetic Masking. In *CHES 2001*, vol. 2162 of *LNCS*, pages 3–15. Springer, 2001.
15. S. Gueron, O. Parzanchevsky, and O. Zuk. Masked Inversion in $GF(2^n)$ Using Mixed Field Representations and its Efficient Implementation for AES. In *Embedded Cryptographic Hardware: Methodologies and Architectures*, pages 213–228. Nova Science Publishers, 2004.
16. P. Junod and S. Vaudenay. FOX: a new family of block ciphers. In *SAC 2004*, vol. 3357 of *LNCS*, pages 114–129. Springer, 2004.
17. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO '99*, vol. 1666 of *LNCS*, pages 388–397. Springer, 1999.
18. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO '96*, vol. 1109 of *LNCS*, pages 104–113. Springer, 1996.
19. S. Kunz-Jacques, F. Muller, and F. Valette. The Davies-Murphy Power Attack. In *ASIACRYPT 2004*, vol. 3329 of *LNCS*, pages 451–467. Springer, 2004.
20. T. Messerges. Securing the AES Finalists Against Power Analysis Attacks. In *FSE 2000*, vol. 1978 of *LNCS*, pages 150–164. Springer, 2000.
21. T. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant software. In *CHES 2000*, vol. 1965 of *LNCS*, pages 238–251. Springer, 2000.
22. R. Oppligern. *Contemporary Cryptography.* ARTECH House, 2005.
23. E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-box. In *FSE 2005*, vol. 3557 of *LNCS*, pages 413–423. Springer, 2005.
24. E. Oswald, S. Mangard, and N. Pramstaller. Secure and Efficient Masking of AES – A Mission Impossible ? Cryptology ePrint Archive, Report 2004/134, 2004. `http://eprint.iacr.org/`.
25. E. Oswald and K. Schramm. An Efficient Masking Scheme for AES Software Implementations. In *WISA 2005*, vol. 3786 of *LNCS*, pages 292–305. Springer, 2006.
26. C. Paar. *VLSI Architectures for Bit Parallel Computations in Galois Fields.* PhD thesis, Universität Essen, 1994.
27. V. Rijmen. Efficient Implementation of the Rijndael S-box, 2000. Available at `http://www.esat.kuleuwen.ac.be/~rijmen/rijndael/sbox.pdf`.
28. A. Rudra, P. K. Bubey, C. S. Jutla, V. Kumar, J. Rao, and P. Rohatgi. Efficient Rijndael Encryption Implementation with Composite Field Arithmetic. In *CHES 2001*, vol. 2162 of *LNCS*, pages 171–184. Springer, 2001.
29. E. Trichina, D. DeSeta, and L. Germani. Simplified Adaptive Multiplicative Masking for AES. In *CHES 2002*, vol. 2523 of *LNCS*, pages 187–197. Springer, 2002.
30. E. Trichina, L. Korkishko, and K. H. Lee. Small Size, Low Power, Side Channel-Immune AES Coprocessor, Design and Synthesis Results. In *AES 4*, vol. 3373 of *LNCS*, pages 113–127. Springer, 2005.
31. E. Trichina and L. Korkishko. Secure and Efficient AES Software Implementation for Smart Cards. In *WISA 2004*, vol. 3325 of *LNCS*, pages 425–439. Springer, 2004.
32. E. Trichina. Combinatorial Logic Design for AES SubByte Transformation on Masked Data. Cryptology ePrint Archive, Report 2003/236, 2003. `http://eprint.iacr.org/`.

A Practical Evaluation of Our Method Applied to AES

In this section we present the results of a practical evaluation of our method applied to AES (*cf.* Section 4.1). The implementation was done on a 8-bit smart card on which we do not activate the different hardware countermeasures. Concerning the statistical treatment, we use an improvement of traditional DPA called *Correlation Power Analysis* (CPA) (*cf.* [4]).

Firstly, we attack a straightforward implementation of the AES when accessing the first S-Box during the first round. By using the selection function equal to the Hamming weight of the output of the S-Box, we obtain the result depicted in Figure 2 after 100 executions of the algorithm. The value of the sub-key used by the S-Box is recovered with only 30 executions of the algorithm.

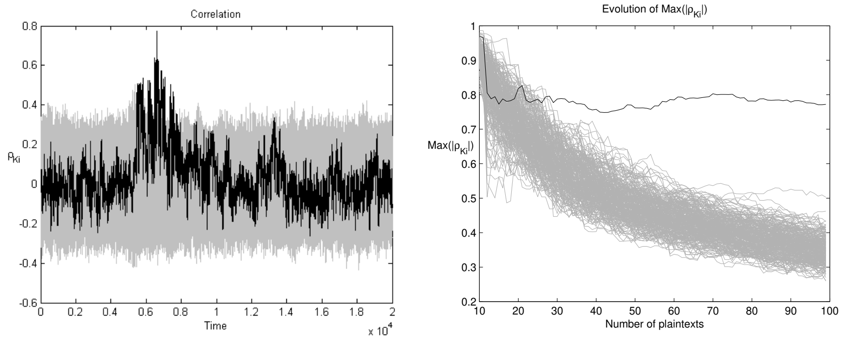

Fig. 2. CPA on non-masked AES S-Box implementation using 100 random plaintexts.

Secondly, we perform the same attack against our DPA-resistant method (*cf.* Algorithm 2) by using 20 000 executions of the algorithm. As shown in Figure 3, the attack fails. We also apply CPA by using several other selection functions such as the Hamming weight of the input of the S-Box. All these attacks fail in the same way.

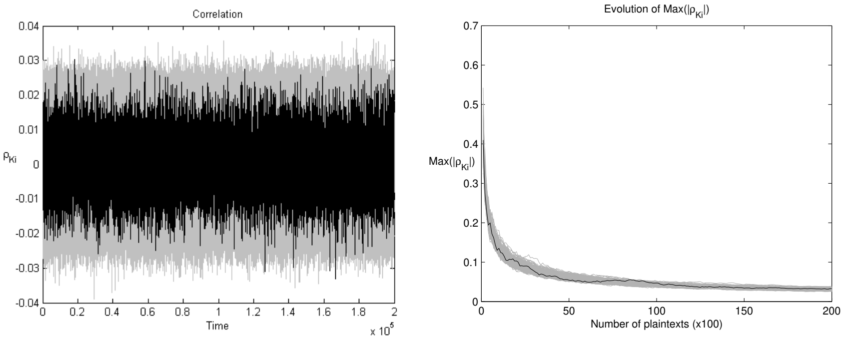

Fig. 3. CPA on Algorithm 2 using 20 000 random plaintexts.