# Attack and Improvement of a Secure S-Box Calculation Based on the Fourier Transform

Jean-Sébastien Coron[1], Christophe Giraud[2], Emmanuel Prouff[2], and Matthieu Rivain[1,2]

[1] University of Luxembourg
jean-sebastien.coron@uni.lu
[2] Oberthur Technologies
{c.giraud,e.prouff,m.rivain}@oberthurcs.com

**Abstract.** At CHES 2006, a DPA countermeasure based on the Fourier Transform was published. This generic countermeasure aims at protecting from DPA any S-box calculation used in symmetric cryptosystems implementations. In this paper, we show that this countermeasure has a flaw and that it can be broken by first order DPA. Moreover, we have successfully put into practice our attack on two different S-box implementations. Finally, we propose an improvement of the original countermeasure and we prove its security against first order DPA.

## 1 Introduction

The processing of a cryptographic algorithm on a physical device may leak information about the manipulated data. To exploit this information, Side Channel Attacks (SCA) were introduced in 1996, *cf.* [8]. It is today composed of a large variety of attacks that differ in the attack model, the nature of the side channels they target or the leakage treatments they perform. The *Differential Power Analysis* (DPA) introduced in [9] is probably the one which has received the most attention in the literature. This attack has indeed been demonstrated to be very powerful against unprotected cryptographic implementations, where it allows the attacker to recover the value of a secret key with only a few leakage measurements. Roughly speaking, a DPA is a statistical attack that correlates a physical leakage with the values of particular intermediate variables (called *sensitive variables* in this paper) that depend on both a public value and the secret key. To avoid information leakage and its exploitation by DPA, the manipulation of sensitive variables must be protected by adding countermeasures to the algorithm.

A very common countermeasure to protect block cipher implementations from DPA is to mask every sensitive variable with a randomly generated variable (called *mask*) and then to perform the calculations by only manipulating the masked variable and/or the mask. When such a technique is applied, a problem occurs which is usually referred in the literature as the *mask correction Problem.* It relies on the difficulty of masking the calculation of non-linear sub-functions (*e.g.* the so-called *S-boxes*), without ever manipulating an intermediate variable that depends on sensitive data. Many papers have been published that aim at providing a solution to this problem (see for instance [1,7,10,11,12]). At CHES

E. Oswald and P. Rohatgi (Eds.): CHES 2008, LNCS 5154, pp. 1–14, 2008.

2006, Prouff, Giraud and Aumônier proposed in [11] a solution that may be of particular interest when the input/output dimensions of the function to protect are small and when the masks values are regenerated many times during the algorithm processing. Moreover, the solution is provided together with a proof of security that allows the reader to formally validate its security. In this paper, we show that contrary to what is claimed in [11], a DPA attack can be successfully mounted against this countermeasure. We exhibit the flaw upon which our attack is based and we present how to successfully exploit it to recover the value of a secret parameter. Finally, we propose an improvement of the countermeasure proposed in [11] and we prove its security *versus* DPA in a realistic model.

## 2 Preliminaries

In the rest of the paper, we say that a variable is *sensitive with respect to DPA* (shortened to *sensitive variable* in the context of the present paper) if it is a non-constant function of a plaintext and a secret key. A DPA (also called first order DPA in the literature when it is compared to higher order DPA) exploits the leakage about a single intermediate sensitive variable. Hereafter, we recall the formal definition of the security against DPA (see for instance [2,4,11]).

**Definition 1.** *A cryptographic algorithm is said to be secure against DPA if all its intermediate variables are independent of any sensitive variable.*

Conversely, an algorithm is said to admit a *first order flaw* if one of its intermediate variables depends on a sensitive variable.

A common countermeasure against DPA is to add (by bitwise or modular addition) a random value called the *mask* to each sensitive variable. Masks and masked variables propagate throughout the cipher in such a way that every intermediate variable is independent of any sensitive variable. This strategy, called *first order masking*, ensures that the instantaneous leakage is independent of any sensitive variable, thus rendering DPA ineffective.

As pointed out for instance in [6,1], the tricky part when masking the implementation of an algorithm is to deal with the following problem, called *mask correction Problem*:

*Problem 1.* Let $F$ be a $(n, m)$-function (that is a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$). From a masked input $Z \oplus R_1 \in \mathbb{F}_2^n$, the mask $R_1 \in \mathbb{F}_2^n$ and an output mask $R_2 \in \mathbb{F}_2^m$, compute $F(Z) \oplus R_2$ without introducing any first order flaw.

## 3 Secure S-Box Calculation Based on the Fourier Transform

In [11], an algorithm claimed to solve Problem 1 is proposed. The method is based on the involutivity property of the *Fourier Transform.* Before describing it, let us first recall some basics about the transformation itself.

**Algorithm 1.** Computation of an arithmetically masked S-box output from a boolean masked input

INPUTS: A masked input $\tilde{Z} = Z \oplus R_1$, the input mask $R_1$ and a lookup table $\widehat{F}$

OUTPUT: The 3-tuple $((-1)^{(\widetilde{Z}\oplus R_2)\cdot R_1} F(Z) + R_3 \mod 2^n, R_3, R_2)$ where $R_2$ and $R_3$ are random values.

```
Pick up three $n$-bit randoms $R_2$, $R_3$ and $R_4$
$result \leftarrow 2^n R_3 + R_4$
for $a$ from $0$ to $2^n - 1$ do
    $T_1 \leftarrow \mathrm{SSP}(a, \widetilde{Z})$                  $[T_1 = (-1)^{a\cdot\widetilde{Z}}]$
    $T_2 \leftarrow \widetilde{Z} \oplus a$                          $[T_2 = \widetilde{Z} \oplus a]$
    $T_2 \leftarrow T_2 \oplus R_2$                                  $[T_2 = \widetilde{Z} \oplus a \oplus R_2]$
    $T_2 \leftarrow \mathrm{SSP}(R_1, T_2)$                          $[T_2 = (-1)^{R_1\cdot(\widetilde{Z}\oplus a\oplus R_2)}]$
    $T_2 \leftarrow T_1 \times T_2$                                  $[T_2 = (-1)^{a\cdot\widetilde{Z}\oplus R_1\cdot(\widetilde{Z}\oplus a\oplus R_2)}]$
    $T_2 \leftarrow T_2 \times \widehat{F}(a)$                       $[T_2 = \widehat{F}(a)(-1)^{a\cdot\widetilde{Z}\oplus R_1\cdot(\widetilde{Z}\oplus a\oplus R_2)}]$
    $result \leftarrow result \boxplus T_2$     $[result = (2^n R_3 + R_4) \boxplus \sum_{i\in\{0,a\}} \widehat{F}(i)(-1)^{i\cdot\widetilde{Z}\oplus R_1\cdot(\widetilde{Z}\oplus i\oplus R_2)}]$
end
$result \leftarrow result \gg n$                 $[result = (-1)^{(\widetilde{Z}\oplus R_2)\cdot R_1} F(Z) + R_3 \bmod 2^n]$
return $(result, R_3, R_2)$
```

For every $(n, m)$-function $F$, the Fourier transform $\widehat{F}$ of $F$ is defined for every $Z = (Z_0, \cdots, Z_{n-1}) \in \mathbb{F}_2^n$ by:

$$\widehat{F}(Z) = \sum_{a\in\mathbb{F}_2^n} F(a)(-1)^{a\cdot Z} \quad , \tag{1}$$

where $\cdot$ denotes the scalar product defined by $a \cdot Z = \bigoplus_{i=0}^{n-1} a_i Z_i$.

It is well known that this transformation is involutive, which means that $\widehat{\widehat{F}} = 2^n F$ or equivalently that:

$$F(Z) = \frac{1}{2^n} \sum_{a\in\mathbb{F}_2^n} \widehat{F}(a)(-1)^{a\cdot Z},\ Z \in \mathbb{F}_2^n \quad . \tag{2}$$

Let $R_1$, $R_2$, $R_3$ and $R_4$ be 4 random masks belonging to $\mathbb{F}_2^n$, and let $Z$ denotes a sensitive variable. The algorithm proposed in [11] to process $F(Z)+R_3 \bmod 2^n$ securely from $\tilde{Z} = Z \oplus R_1$ and $R_1$, implements the right-hand side calculus of the following relation (which is a slightly modified version of Relation (2)):

$$\begin{aligned}&(-1)^{(\widetilde{Z}\oplus R_2)\cdot R_1} F(Z) + R_3 \bmod 2^n \\ &\qquad = \left\lfloor \frac{1}{2^n}\left(R' + \sum_{a\in\mathbb{F}_2^n} \widehat{F}(a)(-1)^{a\cdot\widetilde{Z}\oplus R_1\cdot(\widetilde{Z}\oplus a\oplus R_2)} \bmod 2^{2n}\right)\right\rfloor \quad ,\end{aligned} \tag{3}$$

where $R' = 2^n R_3 + R_4$.

Let $SSP$ denote the signed scalar product $X, Y \mapsto (-1)^{X \cdot Y}$, let $\boxplus$ denote the addition modulo $2^{2n}$ and let $\times$ denote the multiplication of two values belonging to $\{-1, 1\}$. We recall hereafter the algorithm proposed in [11] to process the right-hand side of (3) securely.

Finally, it is proposed in [11] to use the method described in [5] in order to transform the arithmetic masking of the output of Algorithm 1 into a boolean masking.

The authors of [11] had proposed a proof of security *versus* DPA for the countermeasure defined by Algorithm 1, but as we will see in the next section, the proof is flawed and the countermeasure is not secure against DPA.

## 4 DPA against the Fourier Transform Based S-Box Calculation

### 4.1 First Order Flaw

Unlike what is claimed in [11], the implementation of Algorithm 1 is not immune against DPA. Indeed, the variable $V = a \cdot \widetilde{Z} \oplus R_1 \cdot (\widetilde{Z} \oplus a \oplus R_2)$ processed at Step 8 brings information about the sensitive variable $Z$ (recalling $\widetilde{Z} = Z \oplus R_1$). To exhibit the dependency between $V$ and $Z$, let us first rewrite $V$ as follows:

$$\begin{aligned} V &= a \cdot \widetilde{Z} \oplus R_1 \cdot (\widetilde{Z} \oplus a \oplus R_2) \\ &= a \cdot (Z \oplus R_1) \oplus R_1 \cdot (\widetilde{Z} \oplus a \oplus R_2) \\ &= a \cdot Z \oplus R_1 \cdot (\widetilde{Z} \oplus R_2) \ . \end{aligned}$$

The relation above shows that the intermediate variable $V$ equals the sensitive variable $a \cdot Z$ ($a$ being a loop index) masked with the scalar product $R_1 \cdot (\widetilde{Z} \oplus R_2)$. Since $R_2$ is uniformly distributed and is independent of both $Z$ and $R_1$, then so does the variable $\widetilde{Z} \oplus R_2$. The flaw of the method proposed in [11] comes from the fact that the scalar product of two uniformly distributed random variables does not output an uniformly distributed random variable. For example, the product $b_1 \cdot b_2$ of two random bits $b_1$ and $b_2$ equals 0 with probability 3/4, and equals 1 with probability 1/4. More generally, for $n$-bit random variables we have the following lemma.

**Lemma 1.** *Let $X$ and $Y$ be two random variables uniformly distributed over $\mathbb{F}_2^n$ and mutually independent. Then the scalar product $X \cdot Y$ satisfies*

$$\Pr[X \cdot Y = 0] = \frac{1}{2} + \frac{1}{2^{n+1}} \ . \qquad (4)$$

*Proof.* We have:

$$P[X \cdot Y = 0] = P[X \neq 0] \cdot P[X \cdot Y = 0 | X \neq 0] + P[X = 0] \cdot P[X \cdot Y = 0 | X = 0] \ .$$

Since the Boolean function $y \in \mathbb{F}_2^n \mapsto x \cdot y$ is linear and not null for every $x \neq 0$, we have $\#\{x \cdot y = 1\} = \#\{x \cdot y = 0\} = 2^{n-1}$. This, together with the fact that $X$ and $Y$ are independent, implies $P[X \cdot Y = 0 | X \neq 0] = \frac{1}{2}$. Since $P[X \cdot Y = 0 | X = 0] = 1$ and $P[X \neq 0] = \frac{2^n - 1}{2^n}$, we deduce (4). ◇

*Remark 1.* In the security proof conducted in [11], it is stated that the uniform distribution of $X$ and $Y$ implies the one of $X \cdot Y$. We show in Lemma 1 that this assertion is actually wrong.

Lemma 1 implies that the distribution of $R_1 \cdot (\widetilde{Z} \oplus R_2)$ has a bias $\frac{1}{2^{n+1}}$ with respect to the uniform distribution. Since the sensitive variable $a \cdot Z$ is masked with a biased mask, the variable $V$ defined in (4) leaks information on $a \cdot Z$. This information can be used to recover $Z$ by DPA.

### 4.2 DPA Attack

A DPA attack [9] targets the leakage $L(b)$ generated by the processing of a sensitive bit $b$ in order to recover information about a secret which we denote here by $k^\star$. It can be performed with only a few information about the leakage and it actually only assumes that the expectation of $L(b)$ depends on the value of $b$. Let us first recall the outlines of the attack in the general case where $b$ can be expressed as:

$$b = f(X, k^\star) \ , \tag{5}$$

where $f$ is a Boolean function and $X$ is a public variable.

**Description.** To perform a DPA, the target algorithm is executed several times, say $N$, for a sequence of values $(x_i)_{i \le N}$ taken by $X$. For each execution, the attacker measures the leakage $l_i$ generated by the processing of $b$. Then, the resulting *leakage measurement sequence* $(l_i)_{i \le N}$ is involved to (in)validate a key hypothesis $k$ on $k^\star$. For such a purpose, the attacker first computes the *sequence of guesses* $(b_i)_{i \le N}$ which are the predicted values of the bit $b$ processed in the successive executions: namely, for every $i \le N$ we have $b_i = f(x_i, k)$. Then, the leakage measurements are separated in two categories: the ones for which the predicted bit $b_i$ is equal to 1, and the ones for which it is equal to 0. Finally, the so-called differential $\Delta_k$ corresponding to the difference between the mean values of the two sets is computed:

$$\Delta_k = \frac{\sum_{i=1}^{N} b_i \times l_i}{\sum_{i=1}^{N} b_i} - \frac{\sum_{i=1}^{N} (1 - b_i) \times l_i}{\sum_{i=1}^{N} (1 - b_i)} \ . \tag{6}$$

If the key hypothesis is correct then the expectation satisfies:

$$E[\Delta_{k^\star}] = E[L(1)] - E[L(0)] \ . \tag{7}$$

If the key hypothesis is incorrect then a *ratio* $\alpha \in [0, 1]$ of the $b_i$'s is wrongly predicted and the expectation of the differential satisfies:

$$E[\Delta_k] = (1 - 2\alpha)\big(E[L(1)] - E[L(0)]\big) \ . \tag{8}$$

Since $\alpha$ is usually around $\frac{1}{2}$, we have $E[\Delta_{k \neq k^\star}] \simeq 0$. This implies that, for a sufficiently large $N$, the correct key hypothesis is such that $\Delta_k$ is of maximum amplitude.

*Remark 2.* Depending on the function $f$, it may happen that the correct key hypothesis is not the single one for which $\Delta_k$ is of maximum amplitude. Indeed, a key hypothesis such that $\alpha = 1$ also results in a differential of maximal amplitude. According to (6), this differential and the one corresponding to the correct key hypothesis have exactly the same amplitude but have opposite signs. To differentiate them the attacker needs to determine the polarity of $E[L(1)] - E[L(0)]$.

**DPA Attack Exploiting a Biased Mask.** Let us now consider the case where the target bit $b$ is masked, namely:

$$b = f(X, k^\star) \oplus R \ , \tag{9}$$

where $R$ is a random bit.

If $R$ is uniformly distributed over $\mathbb{F}_2$, then no successful DPA attack is possible. Indeed, in that case $b$ equals 0 (*resp.* 1) with probability $\frac{1}{2}$ independently of $k^\star$. Conversely, when the distribution of $R$ is biased compared to the uniform distribution, then the distribution of $b$ depends on $f(X, k^\star)$, which renders DPA possible. In the following, we denote by $\varepsilon \neq 0$ the bias such that $P[R = 0] = \frac{1}{2} + \varepsilon$.

The DPA works in the same way as in the unmasked case. The sequence of guesses is still defined as $b_i = f(x_i, k)$ (since $R$ is not predictable) and the differential $\Delta_k$ is computed according to (6). The randomization provided by $R$ implies that the bit effectively processed equals $f(x_i, k^\star)$ with probability $\frac{1}{2} + \varepsilon$. One deduces that, for the correct key hypothesis, a portion $\frac{1}{2} + \varepsilon$ of the $b_i$'s is correctly predicted while a portion $\frac{1}{2} - \varepsilon$ is wrongly predicted in average. This implies that the expectation of the differential for the correct key hypothesis satisfies:

$$E[\Delta_{k^\star}] = \left(\frac{1}{2} + \varepsilon\right) \big(E[L(1)] - E[L(0)]\big) + \left(\frac{1}{2} - \varepsilon\right) \big(E[L(0)] - E[L(1)]\big) \ ,$$

that is:

$$E[\Delta_{k^\star}] = 2\varepsilon \times \big(E[L(1)] - E[L(0)]\big) \ .$$

Hence the expectation of $\Delta_{k^\star}$ is divided by a factor $\frac{1}{2\varepsilon}$ compared to an unprotected implementation (this also holds for the differentials $\Delta_k$ obtained for wrong key hypotheses – see Appendix A – ). This implies, according to the analysis in [3], that the number of required leakage measurements is roughly multiplied by $(\frac{1}{2\varepsilon})^2$. A more detailed analysis is conducted in Appendix A where we give the exact distribution of $\Delta_k$, assuming that the leakage noise has a Gaussian distribution.

As a result, Lemma 1 implies that a DPA on Algorithm 1 exploiting the flaw exhibited in Section 4.1 is expected to require about $2^{2n}$ times more leakage measurements than a DPA when no masking is used. Since Algorithm 1 is only interesting for a small value of $n$ (*e.g.* $n = 4$), this factor is not prohibitive.

### 4.3 DPA Attack on the Flaw

In this section, we apply the DPA attack described in Section 4.2 in order to exploit the flaw exhibited in Section 4.1. More precisely, our attack targets a bit

$b$ which is a scalar product $a \cdot Z$ masked with a biased mask $R = R_1 \cdot (\tilde{Z} \oplus R_2)$, that is

$$b = a \cdot Z \oplus R \ . \tag{10}$$

We recall that $a$ refers to a loop index in Algorithm 1 and that its value can be chosen by the attacker among $\{0, \cdots, 2^n - 1\}$. The sensitive variable $Z$ is the sensitive S-box input and it can be written as a function of a public variable $X$ and a piece of secret data $k^\star$. The way our attack is performed depends on this function which can take several forms. In the sequel we consider two usual cases.

The first one is referred as the *linear case* and assumes:

$$Z = X \oplus k^\star \ .$$

This occurs for instance in AES and in FOX algorithms for the first round S-box calculation.

The second case, referred as the *non-linear case*, assumes the existence of a non-linear transformation $\phi$ such that:

$$Z = \phi(X \oplus k^\star) \ .$$

This occurs for instance in the AES algorithm implemented using the *composite field method* [10,11] (see [11, §4.1] for details). In that case, $\phi$ is the non-linear $(8, 4)$-function which from $a \in \mathbb{F}_{256}$ processes $d \in \mathbb{F}_{16}$ according to the notations of [10,11].

**The Linear Case.** We consider here the case where the targeted bit can be expressed as $b = a \cdot (X \oplus k^\star) \oplus R$ that is:

$$b = a \cdot X \oplus a \cdot k^\star \oplus R \ . \tag{11}$$

The bit $b$ in (11) only depends on one secret binary value $a \cdot k^\star$. Therefore, a DPA on $b$ will provide at most one bit of information on $k^\star$. Hence, recovering the whole secret $k^\star$ requires to perform a DPA attack on $b$ for $t$ different loop indices $a_0$, ..., $a_{t-1}$.

When mounting a DPA attack on $b$ for a particular loop index $a$, the sequence of guesses can only take one of the two following forms: $(a \cdot x_i)_i$ or $(a \cdot x_i \oplus 1)_i$. According to (6), these two sequences result in two differentials that are opposite one to each other. The attacker does not know *a priori* which of these differentials correspond to the correct key hypothesis. Indeed, depending on the device, the polarity $(-1)^s$ of the good differential $\Delta_{a \cdot k^\star}$ may be positive or negative. In other terms, the DPA allows the attacker to recover the value of $a \cdot k^\star \oplus s$, where $k^\star$ and $s$ are unknown.

Since the polarity $s$ is the same for all the loop indices $a$, then performing $t$ DPA attacks for $t$ different loop indices $a_0$, ..., $a_{t-1}$ provides the attacker with a system of $t$ equations and $n + 1$ variables (the polarity bit $s$ and the $n$ bits of $k^\star$). Solving this system requires to have at least $t = n + 1$ equations. After choosing $n$ indices $a_i$ having linearly independent vectorial representations in $\mathbb{F}_2^n$ and after defining $a_n = a_0 \oplus a_1$, it can be checked that solving the system allows the attacker to unambiguously determine the value of $k^\star$.

**The Non-linear Case.** We now consider the case where $b$ satisfies:

$$b = a \cdot \phi(X \oplus k^\star) \oplus R \ . \tag{12}$$

For a non-linear $\phi$, the attack is analogous to a classical DPA on some output bit of *e.g.* a DES or AES S-box [9]. The non-linearity of $\phi$ ensures that for the correct key hypotheses a peak of maximal amplitude will appear while for most other key hypothesis no peak will appear. This enables to fully recover $k^\star$.

In this section, we have described how to exploit the leakage on a sensitive bit which is masked with a biased random bit. In the linear case, the attack requires to perform $n+1$ DPAs while only one DPA is needed in the non-linear case. In the following section, we present experimental results for these two attacks.

## 5 Experimental Results

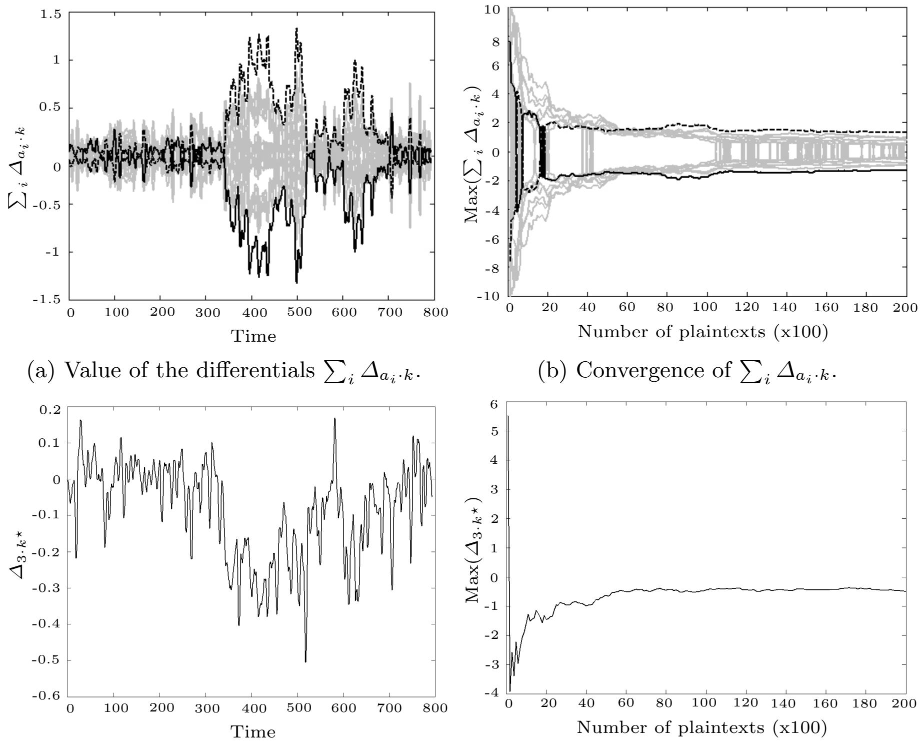
We put into practice the attacks described in Section 4.2 for two S-box implementations on an 8-bit smart card. Both attacks exploited the power consumption resulting from several S-box calculations.

Regarding the linear case, we performed the attack on the S-box calculation of FOX algorithm during the first round protected by the method described in [11]. In this case, the sensitive bits we targeted are of the form $a \cdot (X \oplus k^\star) \oplus R$, where $a, X, k^\star \in \mathbb{F}_2^4$. Following the outlines of the attack described in Section 4.3 for the linear case, we have applied $4+1$ DPAs on five different loop iterations of Algorithm 1, namely one DPA for every $a \in \{1, 2, 4, 8, 3\}$.

Figure 1.a represents the value of $\sum_{i=0}^{3} \Delta_{a_i \cdot k}$, where $a_i = 2^i$, obtained after 20 000 executions of the algorithm. The full black curve corresponds to the correct subkey value $k^\star$ and the dotted black curve corresponds to the complementary of this value. As expected, these two candidates are such that the highest peaks of the differential vectors $\Delta_{a_i \cdot k}$ are either all positive or all negative, hence leading to the highest amplitudes for $\sum_{i=0}^{3} \Delta_{a_i \cdot k}$. As explained in Section 4.3, we then computed the differential $\Delta_{a \cdot k^\star}$ for $a = a_0 \oplus a_1 = 3$. Figure 1.c illustrates this computation. The polarity of the highest peak of $\Delta_{3 \cdot k^\star}$ being negative, one deduces that the correct subkey value $k^\star$ corresponds to the full black curve in Figure 1.a.

Figures 1.b and 1.d represent respectively the convergence of the peak of maximal amplitude for $\sum_{i=0}^{3} \Delta_{a_i \cdot k}$ and for $\Delta_{3 \cdot k^\star}$ according to the number of power consumption measurements. By analyzing these curves, we deduce that the value of the 4-bit subkey $k^\star$ is recovered by using about 8 000 executions of the algorithm.

Regarding the non-linear case, we attacked the AES S-box calculation using the composite field method in order to perform the inversion in $\mathbb{F}_2^4$ instead of $\mathbb{F}_2^8$ and the method of [11] to protect this inversion (see [11, § 4.1] for more details). In that case, the targeted bit is of the form $a \cdot \phi(X \oplus k^\star) \oplus R$ where $X, k^\star \in \mathbb{F}_2^8$, $a \in \mathbb{F}_2^4$ and $\phi : \mathbb{F}_2^8 \to \mathbb{F}_2^4$. Figure 2.a represents the value of the differentials $\Delta_k$'s for $k \in \mathbb{F}_2^8$ and $a = 1$, when 200 000 executions of the algorithm are used. It can be seen that the correct subkey $k^\star$ (plotted in black) is easily distinguishable.



(a) Value of the differentials $\sum_i \Delta_{a_i \cdot k}$.

(b) Convergence of $\sum_i \Delta_{a_i \cdot k}$.

(c) Value of the differential $\Delta_{3 \cdot k^\star}$.

(d) Convergence of $\Delta_{3 \cdot k^\star}$.

**Fig. 1.** Practical DPA attack – the linear case

Figure 2.b represents the convergence of the maximum peak amplitude for the differentials according to the number of power consumption measurements. The analysis of these curves shows us that the value of the 8-bit subkey $k^\star$ is recovered after about 100 000 executions of the algorithm.

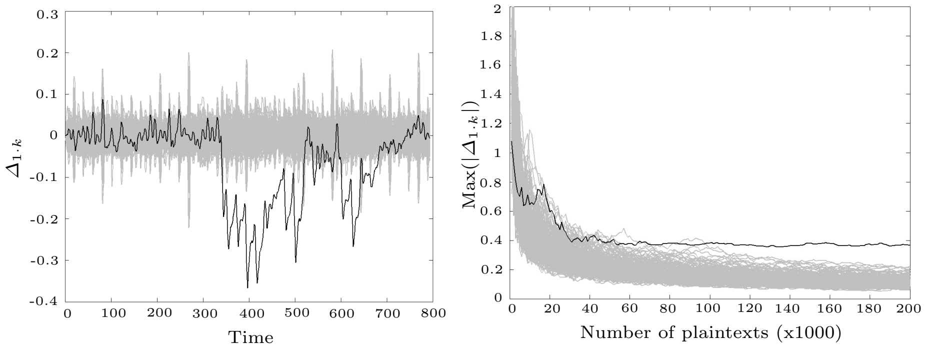
## 6 An Improved Version of a Secure S-Box Calculation

In the following we propose an improvement of Algorithm 1 that allows to circumvent the flaw depicted in Section 4.1 and also leads to a more efficient implementation.

The new algorithm is still a secure calculation of a Fourier Transform but it is based on a slightly modified version of (3) which we rewrite in the following form:

$$\begin{aligned}(-1)^{R_2}F(Z) + R_3 \bmod 2^n & \\ &= \left\lfloor \frac{1}{2^n}\left(R' + \sum_{a \in \mathbb{F}_2^n} \widehat{F}(a)(-1)^{R_2 \oplus a \cdot \widetilde{Z} \oplus a \cdot R_1} \bmod 2^{2n}\right)\right\rfloor \quad , \qquad (13)\end{aligned}$$

where $\widetilde{Z} = Z \oplus R_1$, $R_2 \in \mathbb{F}_2$, $(R_1, R_3, R_4) \in (\mathbb{F}_2^n)^3$ and $R' = 2^n R_3 + R_4$.



(a) Value of the vectors $\Delta_{1\cdot k}$.

(b) Convergence of $|\Delta_{1\cdot k}|$.

**Fig. 2.** Practical DPA attack – the non-linear case

After a brief look at (13) (and before the deeper analysis conducted later on in this section), we can notice that the sensitive variable $a \cdot Z$ is now masked with the uniformly distributed random bit $R_2$. Furthermore, it may be noticed that the exponent in the summation in (13) involves less operations than in (3).

Let us denote by SP the function $X, Y \mapsto X \cdot Y$ and by SFT the function $X, T \mapsto \widehat{F}(X)(-1)^T$. As we prove in this section, Algorithm 2 implements (13) securely.

**Algorithm 2.** First order Secure S-box calculation

Inputs: A masked value $\widetilde{Z} = Z \oplus R_1$ and the mask $R_1$

Output: The 3-tuple $((-1)^{R_2}F(Z) + R_3 \mod 2^n, R_3, R_2)$, where $R_2$ and $R_3$ are random values.

```
Generate a random bit $R_2$
Generate two $n$-bit random $R_3$ and $R_4$
$result \leftarrow 2^n R_3 + R_4$
for $a$ from 0 to $2^n - 1$ do
    $T_1 \leftarrow \mathrm{SP}(a, \widetilde{Z})$        $[T_1 = a \cdot \widetilde{Z}]$
    $T_1 \leftarrow T_1 \oplus R_2$        $[T_1 = R_2 \oplus a \cdot \widetilde{Z}]$
    $T_2 \leftarrow \mathrm{SP}(a, R_1)$        $[T_2 = a \cdot R_1]$
    $T \leftarrow T \oplus T$        $[T = R \oplus a \cdot Z]$
    $T_1 \leftarrow \mathrm{SFT}(a, T_1)$        $[T_1 = \widehat{F}(a)(-1)^{R_2 \oplus a \cdot Z}]$
    $result \leftarrow result \boxplus T_1$        $[result = (2^n R_3 + R_4) \boxplus \sum_{i \in \{0,a\}} \widehat{F}(i)(-1)^{R_2 \oplus i \cdot Z}]$
end
$result \leftarrow result \gg n$        $[result = (-1)^{R_2} F(Z) + R_3 \bmod 2^n]$
return $(result, R_3, R_2)$
```

**Efficiency Analysis.** Although Algorithm 2 is more secure than Algorithm 1, it is also faster. For each loop, Algorithm 2 requires two XORs, two calls to the function SP and one call to the lookup table SFT. Therefore, for each loop Algorithm 1 performs 2 extra multiplications compared to Algorithm 2. Combining this result with the fact that function SP is slightly faster than function SSP, we deduce that our method is faster than the one proposed in [11].

**Security Analysis.** In Table 1, we list the intermediate variables of Algorithm 1 that involve a sensitive variable. The values which only depend on the loop counter or on a random value are obviously omitted.

**Table 1.** The different sensitive values manipulated during Algorithm 2

| Step | Instruction | Masked Value | $Mask(s)$ |
|---|---|---|---|
| 5.1 | register $\leftarrow$ $\widetilde{Z}$ | $\widetilde{Z}$ | $R_1$ |
| 5.2 | $T_1 \leftarrow \mathrm{SP}(a, \widetilde{Z})$ | $a \cdot \widetilde{Z}$ | $a \cdot R_1$ |
| 6 | $T_1 \leftarrow T_1 \oplus R_2$ | $R_2 \oplus a \cdot \widetilde{Z}$ | $R_2 \oplus a \cdot R_1$ |
| 8 | $T_1 \leftarrow T_1 \oplus T_2$ | $R_2 \oplus a \cdot Z$ | $R_2$ |
| 9 | $T_1 \leftarrow \mathrm{SFT}(a, T_1)$ | $\widehat{F}(a)(-1)^{R_2 \oplus a \cdot Z}$ | $R_2$ |
| 10 | $result \leftarrow result \boxplus T_1$ | $(2^n R_3 + R_4) \boxplus \sum_i \widehat{F}(i)(-1)^{R_2 \oplus i \cdot Z}$ | $(R_2, R_3, R_4)$ |
| 11 | $result \leftarrow result \gg n$ | $(-1)^{R_2} F(Z) + R_3 \bmod 2^n$ | $R_3$ |

As it can be checked in Table 1, the intermediate variables manipulated at Steps 5.1, 6, 8, 9, 10 and 11 are additively masked with a uniformly distributed random variable (resp. $R_1$, $R_2 \oplus a \cdot R_1$, $R_2$, $R_2$, $R_3 || R_4$ and $R_3$) which is independent of the sensitive variable. Those intermediate variables are therefore independent of the sensitive variable $Z$.

The intermediate variable at Step 5.2 can be rewritten $a \cdot Z \oplus a \cdot R_1$. When $a$ equals 0, this variable equals 0 whatever $Z$ and $R_1$. Otherwise, for every $a \neq 0$ the variable $a \cdot R_1$ is uniformly distributed and independent of $Z$. We deduce that $a \cdot Z \oplus a \cdot R_1$ (and hence $a \cdot \widetilde{Z}$) is independent of $Z$ whatever $a$.

Therefore, we have proved that all the intermediate variables manipulated during the execution of Algorithm 1 are independent of $Z$, which implies that our method is secure against first order DPA.

## 7 Conclusion

In this paper, we have shown that a provably secure DPA countermeasure published at CHES 2006 has a flaw. We have explained how this flaw can be exploited to mount an efficient attack on S-box implementations protected by this countermeasure. Our attack is not only theoretical since we have successfully put it into practice on two different S-box implementations: the AES S-box using the composite field method and the FOX S-box.

Finally, we have proposed an improvement of the CHES 2006 countermeasure for which we prove the resistance against first order DPA. Moreover we showed that our improvement is not only more secure but can also be implemented more efficiently than the original countermeasure.

## References

1. Akkar, M.-L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)
2. Blömer, J., Guajardo, J., Krummel, V.: Provably Secure Masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004)
3. Clavier, C., Coron, J.-S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
4. Coron, J.-S., Prouff, E., Rivain, M.: Side Channel Cryptanalysis of a Higher Order Masking Scheme. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 28–44. Springer, Heidelberg (2007)
5. Goubin, L.: A Sound Method for Switching between Boolean and Arithmetic Masking. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 3–15. Springer, Heidelberg (2001)
6. Goubin, L., Patarin, J.: DES and Differential Power Analysis – The Duplication Method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
7. Gueron, S., Parzanchevsky, O., Zuk, O.: Masked Inversion in GF($2^n$) Using Mixed Field Representations and its Efficient Implementation for AES. In: Nedjah, N., Mourelle, L.M. (eds.) Embedded Cryptographic Hardware: Methodologies and Architectures, pp. 213–228. Nova Science Publishers (2004)
8. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
9. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
10. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A Side-Channel Analysis Resistant Description of the AES S-box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
11. Prouff, E., Giraud, C., Aumonier, S.: Provably Secure S-Box Implementation Based on Fourier Transform. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249. Springer, Heidelberg (2006)
12. Rivain, M., Dottax, E., Prouff, E.: Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. Cryptology ePrint Archive, Report 2008/021 (2008), http://eprint.iacr.org/

## A Distribution of the Differentials

In this section, we investigate the distribution of the differential $\Delta_k$ when the attack targets a masked bit $b = f(X, k^\star) \oplus R$ where $R$ is a random bit satisfying $P[R = 0] = \frac{1}{2} + \varepsilon$. Our analysis includes the unmasked case by setting $\varepsilon$ to $\frac{1}{2}$.

We make the usual assumption that the leakage has a Gaussian distribution:

$$L(b) \sim \mathcal{N}\left(\mu - \frac{\delta}{2}(-1)^b, \sigma^2\right) , \tag{14}$$

where $\mu$, $\delta$ and $\sigma$ are constants and $\delta$ equals $E[L(1)] - E[L(0)]$.

The leakage measurement $l_i$ obtained for the $i^{\text{th}}$ encryption can thus be expressed as:

$$l_i = \mu - \frac{\delta}{2}(-1)^{b_i^\star + r_i} + \eta_i \ , \tag{15}$$

where, for the $i^{\text{th}}$ encryption, $b_i^\star$ is the unmasked value of $b$ (*i.e.* $b_i^\star = f(x_i, k^\star)$), $r_i$ is the mask value and $\eta_i$ is the noise in the leakage measurement.

We make the additional assumption that for every key hypothesis $k$, the sequence of guesses satisfies: $\#\{i; b_i = 0\} = \#\{i; b_i = 1\} = N/2$. This assumption is realistic since the functions $f(\cdot, k)$ are usually *balanced* (*i.e.* $\#\{x; f(x,k) = 1\} = \#\{x; f(x,k) = 0\}$) and since the $x_i$'s are usually uniformly distributed. It allows us to rewrite (6) as:

$$\Delta_k = -\frac{2}{N}\left(\sum_{i=1}^{N}(-1)^{b_i} l_i\right) \ . \tag{16}$$

This relation together with (15) leads to:

$$\begin{aligned}\Delta_k &= \frac{\delta}{N}\sum_{i=1}^{N}(-1)^{b_i + b_i^\star + r_i} - \frac{2}{N}\sum_{i=1}^{N}(-1)^{b_i}\eta_i \\ &= \frac{\delta}{N}\left(\sum_{\substack{i=1\\ b_i = b_i^\star}}^{N}(-1)^{r_i} - \sum_{\substack{i=1\\ b_i \neq b_i^\star}}^{N}(-1)^{r_i}\right) - \frac{2}{N}\sum_{i=1}^{N}(-1)^{b_i}\eta_i\end{aligned}$$

Recalling that $\alpha$ is the *ratio* of the $b_i$'s that are wrongly predicted (*i.e.* $\alpha = \#\{i; b_i \neq b_i^\star\}/N$) and after rewriting $(-1)^{r_i}$ as $1 - 2r_i$, we get:

$$\Delta_k = \delta(1 - 2\alpha) + \frac{2\delta}{N}\left(\sum_{\substack{i=1\\ b_i \neq b_i^\star}}^{N} r_i - \sum_{\substack{i=1\\ b_i = b_i^\star}}^{N} r_i\right) - \frac{2}{N}\sum_{i=1}^{N}(-1)^{b_i}\eta_i \ .$$

Since $r_i$ is distributed over $\mathbb{F}_2$ with $P[r_i = 1] = 1/2 - \varepsilon$ then for every $I \subseteq \{1, \cdots, N\}$, the sum $\sum_{i \in I} r_i$ has a binomial distribution with parameter $(\#I, 1/2 - \varepsilon)$. Moreover, since $\eta_i$ has a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, then the sum $\sum_{i=1}^{N}(-1)^{b_i}\eta_i$ has a Gaussian distribution $\mathcal{N}(0, N\sigma^2)$. This way, we obtain:

$$\Delta_k \sim \mathcal{N}\left(\delta(1-2\alpha), \frac{4\sigma^2}{N}\right) + \frac{2\delta}{N}\mathcal{B}\left(\alpha N, \frac{1}{2} - \varepsilon\right) - \frac{2\delta}{N}\mathcal{B}\left((1-\alpha)N, \frac{1}{2} - \varepsilon\right) \ .$$

After approximating $\mathcal{B}(n,p)$ by $\mathcal{N}(np, np(1-p))$ (which is almost exact when $n \geq 30$, $np > 5$ and $n(1-p) > 5$), we finally get:

$$\Delta_k \sim \mathcal{N}\left(2\varepsilon \times \delta(1 - 2\alpha), \frac{4\sigma^2 + \delta^2(1 - 4\varepsilon^2)}{N}\right) \ .$$

This relation shows that the biased masking results in a reduction of the expectation of $\varDelta_k$ and in an increase of its variance. The expectation is divided by a factor $1/2\varepsilon$ while its variance is multiplied by a factor $1 + \delta^2(1 - 4\varepsilon^2)/\sigma^2$. When the leakage signal-to-noise ratio is low, *i.e.* $\sigma \gg \delta$, then the biais has a weak influence on the variance and its main effect is the reduction of the expectation. According to [3] this results in an increase of the number of required leakage measurements by a factor $(1/2\varepsilon)^2$. If the leakage signal-to-noise ratio is not that low, the increase of the variance is significant and the number of required leakage measurements is multiplied by $(1/2\varepsilon)^2\big(1 + \delta^2(1 - 4\varepsilon^2)/\sigma^2\big)$.