First-Order Side-Channel Attacks on the Permutation Tables Countermeasure

–Extended Version–

Emmanuel Prouff and Robert McEvoy

[1] Oberthur Technologies, France
e.prouff@oberthur.com
[2] Claude Shannon Institute for Discrete Mathematics, Coding and Cryptography, University College Cork, Ireland
robertmce@eleceng.ucc.ie

Abstract. The use of random permutation tables as a side-channel attack countermeasure was recently proposed by Coron [6]. The countermeasure operates by ensuring that during the execution of an algorithm, each intermediate variable that is handled is in a permuted form described by the random permutation tables. In this paper, we examine the application of this countermeasure to the AES algorithm as described in [6], and show that certain operations admit first-order side-channel leakage. New side-channel attacks are developed to exploit these flaws, using correlation-based and mutual information-based methods. The attacks have been verified in simulation, and in practice on a smart card.

Keywords: Side-Channel Attacks, Permutation Tables, CPA, MIA, Masking

1 Introduction

When a cryptographic algorithm is implemented in hardware or embedded software, information may be leaked about the intermediate variables being processed by the device. This information may be present in the device's power consumption, electromagnetic emanations or timing characteristics. The class of implementation attacks called Side-Channel Attacks (SCA) aims to exploit these leakages, and recover secret information [13]. Since their introduction in 1996 [13], a wide variety of material has been published relating to newer and increasingly more sophisticated side-channel attacks. Similarly, side-channel attack countermeasures have become an important research topic in both academia and industry.

Masking is one of the most popular SCA countermeasures, used to protect sensitive variables (i.e. variables whose statistical distribution is dependent on the secret key) [5]. In masking, each sensitive variable x

is combined (masked) with a random bit string m (mask). Masking is usually Boolean, where the mask is combined with the sensitive variable using the XOR function; or arithmetic, where a modular addition is used. If the source of random bits is uniformly distributed, then the resultant masked variable x' is also uniformly distributed, and no longer depends on the secret key. Masking has been well studied, and has been shown to be effective against a number of types SCA [3, 5], but remains ineffective in stronger attack models (*e.g.* Higher-Order SCA [15]).

Recently, Coron presented the permutation tables countermeasure, as an alternative to masking [6]. The new proposal can be viewed as a generalization of the classical approach, where masking is no longer performed through a random translation, but through a random permutation. Like classical masking, the permutation tables countermeasure also requires a random bit string, which is used at the start of the cryptographic algorithm to generate a permutation P. In the case of an encryption algorithm, P is then applied to both the message x to be encrypted and the secret key k, producing $P(x)$ and $P(k)$ respectively. It is these permuted variables that are used by the encryption algorithm. At each stage of the algorithm, the cryptographic operations must be modified so that all of the intermediate variables remain in the permuted form described by P. If the countermeasure is applied correctly, the intermediate variables should all have a uniform distribution independent of sensitive variables, thereby precluding side-channel attacks that rely on statistical dependency of the intermediate variables with the secret key.

In this paper, we examine the application of the permutation tables countermeasure to AES, as described by [6]. We show that certain sensitive intermediate variables in this algorithm are, in fact, not uniformly distributed, and therefore leak side-channel information about the secret key. However, because of the nature of the permutation tables countermeasure, it is not possible to exploit these flaws with classical approaches (such as those used in [7, 9, 11]). In fact, the main issue is to exhibit a sound *prediction function* to correlate with the leakages in correlation-based SCA (*e.g.* Correlation Power Analysis (CPA) [4]). After modeling the side-channel leakage, we use the method proposed in [18] to exhibit a new prediction function for the permuted sensitive variables. An analytical expression for the optimal prediction function is derived, which, for the correct key hypothesis, maximises the correlation with leakage measurements from the algorithm.

Furthermore, since the flawed intermediate variables do not have a monotonic dependency with the sensitive variables, we consider SCA

attacks involving distinguishers able to exploit non-monotonic interdependencies. We investigate how Mutual Information Analysis (MIA) [10, 20] can be applied in order to exploit the flaws, and compare it with the correlation-based approach. Both of these new attacks are performed both in simulation and in practice on a smart card, and are successful at breaking the countermeasure described in [6]. This is the first study of the permutation tables countermeasure in the literature.

The rest of this paper is organised as follows. Section 2 recalls the background mathematics and terminology of side-channel attacks, and describes the notations used in the paper. The permutation tables countermeasure, and its application to the AES cipher, is described in Section 3. Section 4 examines the side-channel security of an AES implementation protected using permutation tables as described by [6], and operations are identified that admit first-order side-channel leakage. In Sections 5 and 6, we describe CPA and MIA attacks that exploit these side-channel flaws. The attacks are verified both in simulation and using power measurements from a smart card. Finally, a patch for an AES implementation using the permutation tables countermeasure is given in Section 7.

2 Preliminaries

2.1 Mathematical Background and Notation

We use calligraphic letters, like $\mathcal{X}$, to denote finite sets (*e.g.* $\mathbb{F}_2^n$). The corresponding capital letter X is used to denote a random variable over $\mathcal{X}$, while the lowercase letter x denotes a particular element from $\mathcal{X}$. The probability of the event $(X = x)$ is denoted $\mathrm{p}\left[X = x\right]$. The uniform probability distribution over a set $\mathcal{X}$ is denoted by $\mathcal{U}(\mathcal{X})$, and the Gaussian probability distribution with *mean* μ and *standard deviation* σ is denoted by $\mathcal{N}(\mu, \sigma^2)$. The mean of X is denoted by $\mathrm{E}\left[X\right]$ and its standard deviation by $\sigma[X]$. The correlation coefficient between X and Y is denoted by $\rho\left[X, Y\right]$. It measures the linear interdependence between X and Y, and is defined by:

$$\rho\left[X, Y\right] = \frac{\mathrm{Cov}\left[X, Y\right]}{\sigma\left[X\right]\sigma\left[Y\right]} \ , \tag{1}$$

where $\mathrm{Cov}\left[X, Y\right]$, called *covariance of* X *and* Y, equals $\mathrm{E}\left[XY\right] - \mathrm{E}\left[X\right]\mathrm{E}\left[Y\right]$. It can be checked [18] that for every function f measurable on $\mathcal{X}$, the correlation $\rho\left[f(X), Y\right]$ satisfies:

$$\rho\left[f(X), Y\right] = \rho\left[f(X), \mathrm{E}\left[Y|X\right]\right] \times \rho\left[\mathrm{E}\left[Y|X\right], Y\right] \ . \tag{2}$$

This implies (see Proposition 5 in [18]) the following inequality:

$$\rho\left[f(X), Y\right] \leq \frac{\sigma\left[\mathrm{E}\left[Y|X\right]\right]}{\sigma\left[Y\right]} \quad . \tag{3}$$

A sample of a finite number of values taken by X over $\mathcal{X}$ is denoted by $(x_i)_i$ or by (x_i) if there is no ambiguity on the index, and the mean of such a sample is denoted by $\overline{x} = \frac{1}{\#(x_i)} \sum_i x_i$. Given two sample sets (x_i) and (y_i), the empirical version of the correlation coefficient is the *Pearson coefficient*:

$$\widehat{\rho}\left((x_i), (y_i)\right) = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2} \sqrt{\sum_i (y_i - \overline{y})^2}} \quad , \tag{4}$$

The correlation and Pearson coefficients relate to affine statistical dependencies, and two dependent variables X and Y can be such that $\rho(X, Y) = 0$. To quantify the amount of information that Y reveals about X (whatever the kind of dependency is), the notion of *mutual information* is usually involved. It is the value $\mathrm{I}(X;Y)$ defined by $\mathrm{I}(X;Y) = \mathrm{H}(X) - \mathrm{H}(X|Y)$, where $\mathrm{H}(X)$ is the entropy of X and where $\mathrm{H}(X|Y)$ is the *conditional entropy of X knowing Y* (see [14] for more details).

2.2 Side-Channel Attack Terminology

We shall view an implementation of a cryptographic algorithm as the processing of a sequence of *intermediate variables*, as defined in [3]. We shall say that an intermediate variable is *sensitive* if its distribution is a function of some known data (for example, the plaintext) and the secret key, and is not constant with respect to the secret key. Consequently, the statistical distribution of a sensitive variable depends on both the key and on the distribution of the known data. If a sensitive intermediate variable appears during the execution of a cryptographic algorithm, then that implementation is said to contain a *first-order flaw*. Information arising from a first-order flaw, that can be monitored via a side-channel (such as timing information or power consumption), is termed *first-order leakage*. A *first-order side-channel attack (SCA)* against an implementation is a SCA that exploits a first-order leakage, in order to recover information about the secret key. Similarly, an rth-*order SCA (Higher-Order SCA* or *HO-SCA)* against an implementation is a SCA that exploits leakages at r different times, which are respectively associated with r different intermediate variables.

Remark 1. In [22], an alternative definition for HO-SCA is used, where an rth-order SCA is defined with respect to r different *algorithmic* variables (which may be manipulated simultaneously, or which may correspond to a single intermediate variable). In this paper, we focus on the countermeasure of [6]; therefore, we adhere to the HO-SCA definition in [6] (which is widely accepted in the community [3, 12, 15, 16]). We further discuss the notion of the order of a side-channel attack in Appendix B.

In order to prevent side-channel attacks on cryptographic implementations, many countermeasures (such as masking and the permutation tables countermeasure) aim to randomise the leakage caused by each intermediate variable. An implementation of a cryptographic algorithm can be said to possess *first-order SCA security* if no intermediate variable in the implementation is sensitive. Similarly, rth-*order SCA security* requires an implementation to be such that no r-tuple of its intermediate variables is sensitive.

3 The Permutation Tables Countermeasure

3.1 Generation of Permutation Tables

In order to use permutation tables as a SCA countermeasure, a new permutation table P must be generated at the beginning of each execution of the cryptographic algorithm. Here, P is described in the context of the AES algorithm, where the intermediate variables are 8-bit words. P comprises two 4-bit permutations p_1 and p_2, and operates on an 8-bit variable x according to:

$$P(x) = p_2(x_h)||p_1(x_l) \ , \tag{5}$$

where x_h and x_l respectively denote the high and low nibbles of x, and $||$ denotes concatenation. Upon each invocation of the algorithm, permutations p_1 and p_2 are randomly chosen from a set of permutations $\mathcal{P}$, defined over $\mathbb{F}_2^4$. For efficiency reasons, the set $\mathcal{P}$ is not defined as the set of all the permutations over $\mathbb{F}_2^4$. Indeed, in such a case the random generation of an element of $\mathcal{P}$ would be costly. In [6], Coron defines an algorithm to generate elements of the set $\mathcal{P}$ from a 16-bit random value. Here, we will assume that the random variable P_1 (respectively P_2) associated with the random generation of p_1 (resp. p_2) satisfies $\mathrm{p}\left[P_1 = p_1\right] = 1/\#\mathcal{P}$ (resp. $\mathrm{p}\left[P_2 = p_2\right] = 1/\#\mathcal{P}$) for every $p_1 \in \mathcal{P}$ (resp. $p_2 \in \mathcal{P}$).

3.2 Protecting AES Using Permutation Tables

The Advanced Encryption Standard (AES) is a well-known block cipher, and details of the algorithm can be found in [6]. Essentially, the AES round function for encryption operates on a 16-byte state (with each element labelled a_i, $0 \leq i \leq 15$), and consists of four transformations: `AddRoundKey`, `SubBytes`, `ShiftRows` and `MixColumns`.

In [6], Coron described how to protect the AES encryption algorithm against side-channel attacks, by using the permutation tables countermeasure. We will refer to this encryption algorithm as *randomised AES*. Firstly, after the random permutation P has been generated (as described in [6]), it is applied to each byte of both the message and the key. For every byte x, we will refer to $u = P(x)$ as the *P-representation* of x. Each permuted value is passed to the AES round function, where it is operated upon by the AES transformations listed above. As noticed by Coron in [6], each of these AES transformations must be carefully implemented, such that: (i) sensitive variables always appear in their P-representation, and (ii) the output of each transformation is in P-representation form. Coron described the following implementations of `AddRoundKey` and `MixColumns` (for details of the other transformations, see [6]):

- Randomised `AddRoundKey` takes two bytes $u = P(x)$ and $v = P(y)$ as inputs, and outputs $P(x \oplus y)$. In order to achieve this, two 8-bit to 4-bit tables are defined (for (u_l, v_l) – resp. (u_h, v_h) – in $(\mathbb{F}_2^4)^2$):

$$\mathtt{XT}_4^1(u_l||v_l) = p_1(p_1^{-1}(u_l) \oplus p_1^{-1}(v_l)) \ , \tag{6}$$

$$\mathtt{XT}_4^2(u_h||v_h) = p_2(p_2^{-1}(u_h) \oplus p_2^{-1}(v_h)) \ . \tag{7}$$

  Tables $\mathtt{XT}_4^1$ and $\mathtt{XT}_4^2$ are calculated at the same time as P, and stored in memory. An 8-bit XOR function, denoted by $\mathtt{XT}_8$, is then computed using those table look-ups (for $u, v \in \mathbb{F}_2^8$):

$$\mathtt{XT}_8(u, v) = \mathtt{XT}_4^2(u_h||v_h)||\mathtt{XT}_4^1(u_l||v_l) \ . \tag{8}$$

- Randomised `MixColumns` is computed as a combination of doubling and XOR operations. To calculate randomised `MixColumns` from the $P(a_i)$'s (the P-representations of the bytes of the AES state), the XOR operations are computed using the $\mathtt{XT}_8$ function in Eq. (8). For the doubling operations, Coron defined a function D_2, such that when applied to $u = P(x)$, we get $D_2(P(x)) = P(\{\mathtt{02}\} \bullet x)$ (where $\{\cdot\}$ denotes hexadecimal notation, and $\bullet$ denotes multiplication modulo

$x^8 + x^4 + x^3 + x + 1$). The P-representation of the first byte of the `MixColumns` output is then calculated using:

$$P(a_0^{new}) = \texttt{XT}_8(D_2(a'_0), \texttt{XT}_8(D_2(a'_1), \texttt{XT}_8(a'_1, \texttt{XT}_8(a'_2, a'_3)))) \ , \qquad (9)$$

where a'_i denotes the P-representation of a_i. The other bytes in the randomised `MixColumns` output can be similarly calculated.

At the completion of the last encryption round, the inverse permutation P^{-1} is applied to each byte of the AES state, revealing the ciphertext.

4 Security of Randomized AES against First-Order SCA

4.1 Examining the Proof of Security

In [6], the author proposes the following Lemma to argue that the randomised AES implementation is resistant against first-order SCA:

Lemma 1. *For a fixed key and input message, every intermediate byte that is computed in the course of the randomised AES algorithm has the uniform distribution in* $\{0,1\}^8$.

In [6], the proof of Lemma 1 is based on the fact that any intermediate AES data W is assumed to be represented as $P(W) = P_2(W_h)||P_1(W_l)$. However, this assumption is incorrect for the implementation described in [6] and recalled in Section 3.2. Indeed, when $\texttt{XT}_8(P(X), P(Y))$ is computed (Eq. (8)), the two functions $\texttt{XT}_4^1$ and $\texttt{XT}_4^2$ are parameterized with the intermediate variables $P_1(X_l)||P_1(Y_l)$ and $P_2(X_h)||P_2(Y_h)$ respectively. Namely, the same permutation P_1 (resp. P_2) is applied to the lowest and the highest nibbles of the intermediate data $W = X_l||Y_l$ (resp. $W = X_h||Y_h$). In this case W is not of the form $P(W)$; therefore, the statement made in [6] to prove Lemma 1 is incorrect. Actually, not only the proof but the Lemma itself is incorrect. If two nibbles are equal, e.g. $X_l = Y_l$, then their P_1-representations will also be equal, i.e. $P_1(X_l) = P_1(Y_l)$, irrespective of P_1. Otherwise, if $X_l \neq Y_l$, then $P_1(X_l)$ and $P_1(Y_l)$ behave like two independent random variables, except that they cannot be equal. This implies that the variable $P_1(X_l)||P_1(Y_l)$ will have two different non-uniform distributions depending on whether X_l equals Y_l or not. This gives rise to first-order leakage.

4.2 First-Order Leakage Points

In the randomised AES, the function $\texttt{XT}_8$ is employed to securely implement every bitwise addition between 8-bit words. To compute the P-representation of $X \oplus Y$ from the P-representations $X' = P(X)$ and $Y' = P(Y)$, the following successive operations are processed:

$$\begin{array}{lll} 1. & R_1 & \leftarrow \texttt{XT}_4^1\ (X'_l||Y'_l) \\ 2. & R_2 & \leftarrow \texttt{XT}_4^2\ (X'_h||Y'_h) \\ 3. & \mathit{output} & \leftarrow \quad R_2||R_1 \end{array}$$

Register *output* contains $P(X \oplus Y)$ at the end of the processing above. Let us focus on the intermediate result R_1 (the same analysis also holds for R_2). It is computed by accessing the table $\texttt{XT}_4^1$ at address $Z = X'_l||Y'_l$ which, by construction, satisfies:

$$Z = P_1(X_l)||P_1(Y_l)\ . \tag{10}$$

As discussed in Section 4.1, the manipulation of Z therefore induces a first-order leakage in the AES implementation, whenever (X_l, Y_l) statistically depends on a secret information and a known data. This condition is satisfied when $\texttt{XT}_8$ is used to process the randomised `AddRoundKey` and randomised `MixColumns` operations during the first round of AES:

- [**Randomised** `AddRoundKey`] During this step, $\texttt{XT}_8$ takes the pair $(P(A), P(K))$ as operand, where K is a round key byte and A is a known byte of the AES state. In this case, (10) becomes:

$$Z = P_1(A_l)||P_1(K_l)\ . \tag{11}$$

- [**Randomised** `MixColumns`] During this step, $\texttt{XT}_8$ takes the pair $(A'_1, A'_2) = (P(S[A_1 \oplus K_1]), P(S[A_2 \oplus K_2]))$ as operand, with S denoting the AES S-box, with A_1 and A_2 being two known bytes of the AES state and with K_1 and K_2 being two round-key bytes.
  In this case, (10) becomes:

$$Z = P_1(S[A_1 \oplus K_1]_l)||P_1(S[A_2 \oplus K_2]_l)\ , \tag{12}$$

  where $S[\cdot]_l$ denotes the lowest nibble of $S[\cdot]$.

Both of the leakage points described above are first-order flaws, since they depend on a single intermediate variable Z. In the next sections, we will develop first-order side-channel attacks, that exploit these first order leakages. In both attacks, we will use the notation $Z(k_l)$ (resp. $Z(k_1, k_2)$)

for the random variable $Z|(K_l = k_l)$ (resp. $Z|(K_1 = k_1, K_2 = k_2)$), each time we need to specify which key(s) Z is related to. The random variable corresponding to the leakage on Z shall be denoted by L. They are related through the following relationship:

$$L = \varphi(Z) + B \ , \tag{13}$$

where φ denotes a deterministic function called the *leakage function* and B denotes independent noise. We shall use the notation $L(k_l)$ (resp. $L(k_1, k_2)$) when we need to specify the key(s) involved in the leakage measurements.

5 Attacking the Randomised `AddRoundKey` Operation

There are currently two main ways to perform an attack on the manipulation of a random variable Z. The first method relies on affine statistical dependencies (for example CPA), whereas the second method relies on any kind of statistical dependency (for example MIA). Here, we describe a CPA attack on the first use of randomised `AddRoundKey` (performing an MIA attack on randomised `AddRoundKey` is less pertinent, as will be discussed in Section 6).

5.1 CPA Preliminaries

In a CPA [4], the attacker must know a good affine approximation $\hat{\varphi}$ of φ. It is common to choose the Hamming Weight (HW) function for $\hat{\varphi}$, as this is known to be a good leakage model for devices such as 8-bit microcontrollers. The attacker must also know a good affine approximation $\hat{Z}$ of Z. Based on these assumptions, key candidates $k_l^\star$ are discriminated by testing the correlation between $\hat{\varphi}(\hat{Z}(k_l^\star))$ and $L(k_l)$, for a sample of leakage measurements from the target device, and the corresponding known plaintexts.

Here, our attack targets the use of $\texttt{XT}_8$ when the first randomised `AddRoundKey` operation is performed. We assume that a sample of N leakages (ℓ_i) has been measured for N known lowest nibbles (a_i) of the AES state. Due to (11) and (13), the ℓ_i's and the a_i's satisfy the following relation:

$$\ell_i = \varphi\left(p_{1,i}(a_i)||p_{1,i}(k_l)\right) + b_i \ , \tag{14}$$

for $1 \leq i \leq N$, where b_i denotes the value of the noise for the ith leakage measurement and where $p_{1,i}$ denotes the permutation used at the time of the ith measurement.

To test a hypothesis $k_l^\star$ on k_l, the following Pearson's coefficient $\widehat{\rho}_{k_l^\star}$ is computed for an appropriate prediction function f:

$$\widehat{\rho}_{k_l^\star} = \widehat{\rho}((\ell_i)_i, (f(a_i, k_l^\star))_i) \ . \tag{15}$$

If f has been well chosen, the expected key will satisfy $k_l = \text{argmax}_{k_l^\star} |\widehat{\rho}_{k_l^\star}|$. This is the case for leakage functions φ in (14) where $\mathrm{E}\left[\varphi[Z(k_l)]\right]$ is not constant on $\mathcal{K}_l$ (recall that $Z(k_l)$ equals $P_1(A)||P_1(k_l)$). Almost all functions φ satisfy this condition. However, this is not the case for functions φ where $\varphi(X||Y) = \varphi(X) + \varphi(Y)$ (*e.g.* $\varphi = \mathrm{HW}$). For those leakage functions, Pearson's coefficient (15) is not a sound key-distinguisher when applied directly to the leakages ℓ_i's. Indeed, in this case, (3) and $\sigma\left[\varphi[Z(K_l)] \mid K_l\right] = 0$ imply that $\rho\left[L(k_l), f(A, k_l)\right]$ is null, regardless of the prediction function f. For such functions φ, it makes sense (see for instance [21]) to focus on higher order moments, and to compute the following Pearson's coefficient for an appropriate function f, which may differ from the case when $o = 1$:

$$\widehat{\rho}_{k_l^\star} = \widehat{\rho}(((\ell_i - \overline{\ell})^o)_i, (f(a_i, k_l^\star))_i) \ . \tag{16}$$

For instance, if $\varphi = \mathrm{HW}$, then the second order centered moments of the $\varphi[Z(k_l)]$'s are different, so (16) must be computed for $o = 2$.

Remark 2. For $o = 1$ (*i.e.* when the CPA focuses on the means), there is no need to center the leakage measurements and the term $\overline{\ell}$ can be omitted. In the other cases, centering the leakage measurements (and thus the predictions) improves the CPA efficiency (see [18]).

When a good approximation $\hat{\varphi}$ of φ is assumed to be known, the efficiency of the CPA relies on the prediction function f that is chosen. This is especially true in our case where data is not simply masked by the addition of a random value, but by a random permutation, so that removing the effect of the masking (even biased) is difficult. Designing a prediction function f, such that a CPA involving this function in (16) succeeds, is not straightforward. Therefore, to exploit the flaw in (11) using a CPA attack, we need to exhibit a sound prediction function f.

5.2 Designing f_{opt}

The target intermediate variable Z in (11) takes the general form $P_1(X)||P_1(Y)$, where P_1, X and Y are random variables and where

Y depends on k_l. In [18], Prouff *et al.* showed that for every function $\mathcal{C} : L \mapsto \mathcal{C}(L)$, the optimal prediction function f_{opt} is the function $x, y \mapsto \mathrm{E}\left[\mathcal{C}(L(k_l))|X = x, Y = y\right]$. In our case, $\mathcal{C}(L(k_l))$ equals $(L(k_l) - \mathrm{E}\left[L(k_l)\right])^o$ for a given order o. To mount the attack , we need an analytical expression for the function f_{opt} so that it can be estimated even when no information on the noise parameters is known (non-profiling attacks). Therefore, we conducted an analytical study of the function f_{opt} defined by:

$$f_{opt}(x, y) = \mathrm{E}\left[(L(k_l) - \mathrm{E}\left[L(k_l)\right])^o \mid X = x, Y = y\right] \ , \tag{17}$$

for $o \in \mathbb{N}$, for L equal to $\hat{\varphi}(Z) + B$, for $B \sim \mathcal{N}(\varepsilon, \sigma^2)$ and for Z equal to $P_1(X)||P_1(Y)$ with $X, Y \sim \mathcal{U}(\mathbb{F}_2^n)$ and P_1 is a random variable over $\mathcal{P}$.

Below we state the results of our analysis (the derivations of the formulas are given in Appendix A):

- To compute (16), we suggest using the prediction function f defined for every $(a_i, k_l^\star) \in (\mathbb{F}_2^4)^2$ by:

$$f(a_i, k_l^\star) = \sum_{p_1 \in \mathcal{P}} (\hat{\varphi}\left(p_1(a_i)||p_1(k_l^\star)\right) - \mathrm{E}\left[\hat{\varphi}_{k_l^\star}\right])^o \mathrm{p}\left[P_1 = p_1\right] \ , \tag{18}$$

  where:

$$\mathrm{E}\left[\hat{\varphi}_{k_l^\star}\right] = 2^{-4} \sum_{a \in \mathbb{F}_2^4} \sum_{p_1 \in \mathcal{P}} \hat{\varphi}\left(p_1(a)||p_1(k_l^\star)\right) \mathrm{p}\left[P_1 = p_1\right] \ . \tag{19}$$

  For $o \in \{1, 2\}$, it is argued in Appendix A that the functions f above are affine equivalent to f_{opt}.
- Let $\delta_x(y)$ be the function defined by $\delta_x(y) = 1$ if $x = y$, and $\delta_x(y) = 0$ otherwise. If we assume that $o = 2$, $\hat{\varphi} = \mathrm{HW}$, and that P_1 has a uniform distribution over $\mathcal{P}$, we suggest using the following function:

$$f_{opt}(a_i, k_l^\star) = \delta_{a_i}(k_l^\star) \ , \tag{20}$$

  which is affine equivalent to f_{opt}.

5.3 Attack Results

In the attack simulations presented in Table 1, we give an estimation of the minimum number of measurements required to achieve the success rate 0.9 for $P_1 \sim \mathcal{U}(\mathcal{P})$ (where $\mathcal{P}$ is designed as proposed in [6]) and

Table 1. Num. of measurements required in simulated CPA attack on randomised `AddRoundKey`

Noise standard deviation	Number of measurements
0	100
0.5	1,000
1	1,500
2	4,500
5	60,000
7	230,000
10	900,000

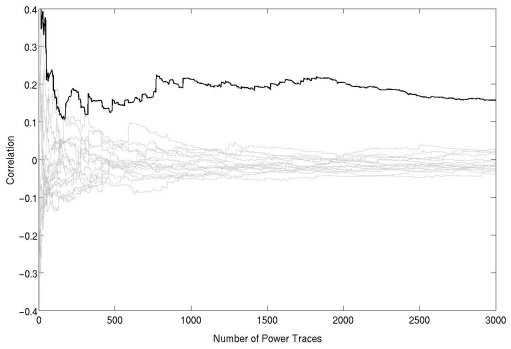

Fig. 1. CPA attack on smart card implementation of randomised `AddRoundKey`

$\varphi = \hat{\varphi} = \mathrm{HW}$. In this case, Pearson coefficients have been computed between $((\ell_i - \overline{\ell})^2)_i$ and $(f_{opt}(a_i, k_l^\star))_i$ for the function f_{opt} defined in (18) for $o = 2$. This success rate is defined as the ratio of successful attacks involving N measurements to the number of attacks involving N measurements. We assumed that an attack is successful if the highest correlation is attained for the correct key. The simulations show that for noiseless measurements, key nibbles can be successfully recovered from the randomised `AddRoundKey` operation using only 100 power traces. We also carried out the CPA attack on a practical smart card implementation of the randomised AES, as described by [6]. We used a Silvercard, which contains a programmable 8-bit PIC16F877 microprocessor, and verified that the power consumption of the card leaks information in the HW model. For each plaintext sent to the card, the encryption operation was performed ten times (with the same random values used to generate the permutation tables) and an average trace of the power consumption was recorded, in order to reduce the effects of acquisition noise. For the attack, we calculated the correlations between $((\ell_i - \overline{\ell})^2)_i$ and $(f_{opt}(a_i, k_l^\star))_i$ for the simplified function f_{opt} defined in (20). The results of the attack are shown in Fig. 1 for various numbers of power traces. The correlation for the correct key nibble is highlighted, showing that the correct key nibble can be recovered using fewer than 1,000 plaintext/power trace pairs.

6 Attacking the Randomised `MixColumns` Operation

In this section, we describe CPA and MIA attacks that target the use of $\mathtt{XT}_8$ when the first `MixColumns` operation is performed. These attacks are

of interest, because they allow recovery of two key bytes (*cf.* Eq. (12)), as opposed to a single key nibble when the `AddRoundKey` operation is targeted. We assume that a sample of N leakages $(\ell_i)_i$ has been measured for N pairs of known AES state values $((a_{1,i}, a_{2,i}))_i$ (where $a_{j,i}$ denotes the known value of byte a_j at the time of the ith measurement ℓ_i). Due to (12) and (13), the ℓ_i's and the $a_{j,i}$'s satisfy the following relation:

$$\ell_i = \varphi(p_{1,i}(S[a_{1,i} \oplus k_1]_l)||p_{1,i}(S[a_{2,i} \oplus k_2]_l) + b_i \ , \tag{21}$$

where b_i and $p_{1,i}$ are as defined for Eq. (14).

6.1 MIA Preliminaries

In MIA attacks [10], key candidates $k^\star$ are discriminated by estimating the mutual information $\mathrm{I}(\hat{\varphi}(\hat{Z}(k^\star)); L(k))$. In an MIA, the attacker is potentially allowed to make weaker assumptions on φ and on Z than in the CPA. Indeed, rather than a good affine approximation of φ and of Z, we only require a pair $(\hat{\varphi}, \hat{Z})$ such that $\mathrm{I}(\hat{\varphi}(\hat{Z}(k)); L(k))$ is non-negligible when the good key k is tested (which may happen even if $\rho(\hat{\varphi}(\hat{Z}(k)), L(k)) = 0$) (see [2, 20] for more details). Therefore, we do not require a lengthy derivation for a prediction function f_{opt}, as was required in Section 5.2 for the CPA. After assuming that a good approximation $\hat{\varphi}$ of the leakage function φ is known, an MIA attack can be performed by estimating the mutual information between the random variable L associated with the leakage measurements ℓ_i in (21) and the prediction function $\hat{\varphi}(S[A_1 \oplus k_1^\star]_l||S[A_2 \oplus k_2^\star]_l)$, for various hypotheses $(k_1^\star, k_2^\star)$ on key bytes (k_1, k_2). The mutual information will attain its maximum value for the correct set of key hypotheses.

Remark 3. As noted in Sec. 5, it was less pertinent to use mutual information as a distinguisher when attacking the randomised `AddRoundKey` operation. The main reason for this is that when φ is the Hamming weight function, the conditional random variable $\varphi(Z(k))$ has the same entropy for each k. As discussed in [10, 20], a way to deal with this issue is to focus on the mutual information between $\varphi(Z(k))$ and predictions in the form $\hat{\varphi} \circ \psi(\hat{Z}(k^\star))$, where ψ is any non-injective function. Even if this approach enables recovery of the key, we checked that for various functions ψ (in particular for functions ψ selecting less than 8 bits in $\hat{Z}(k)$), MIA attacks were much less efficient than CPA.

6.2 Attack Results

In simulation, we tested both an MIA attack and a CPA attack, targeting the first call to $\texttt{XT}_8$ in the first `MixColumns` operation. For the MIA, we used the Kernel Density and Parametric estimation methods described in [20] to estimate the mutual information. For the same reasons as given in Sec. 5.2 (and Appendix A), the CPA simulations used the pre-processing described in Eq. (16), and the following prediction function:

$$f_{opt}(a_{1,i}, k_1^\star, a_{2,i}, k_2^\star) = \delta_{(S[a_{1,i} \oplus k_1^\star]_l)}(S[a_{2,i} \oplus k_2^\star]_l) \tag{22}$$

In the attack simulations presented in Table 2, we give a rough estimation of the minimum number of measurements required to achieve the success rate 0.9 for the different distinguishers. In these experiments, one key byte was fixed at the correct value, and the distinguishers were calculated for the 2^8 values of the second key byte. Fewer measurements are required for a successful attack using CPA than are required when using MIA, for low-noise measurements. This is to be expected, since in the simulations, the HW of the attack variable leaks perfectly, so there is a linear relation between the deterministic part of the leakage and the prediction. MIA is more useful when the relationship between the leakage and the prediction is non-linear. It is interesting to note that when the measurements are noisy, the parametric MIA attack is more efficient than the CPA attack (even in this simulation context that is favourable to CPA).

These attacks were also verified using measurements from the smart card implementation, as shown in Figures 2 and 3 (where the distinguisher for the correct key byte is highlighted). Since the noise in these acquisitions has been reduced due to averaging, the CPA succeeds in fewer measurements ($\sim 2,000$ power traces) than an MIA attack ($\sim 23,000$ traces, using the histogram method to estimate the mutual information [10]).

Table 2. Num. measurements required in MIA and CPA attacks on randomised `MixColumns` (where '$-$' implies no successful result with up to 1 million measurements)

Noise standard deviation	0	0.5	1	2	5	7	10	15	20
Nb of measurements [MIA with Kernel]	2,500	20,000	60,000	290,000	–	–	–	–	–
Nb of measurements [Parametric MIA]	na	3,000	4,000	25,000	250,000	500,000	800,000	–	–
Nb of measurements [CPA with f_{opt}]	1,000	1,000	1,500	6,500	120,000	550,000	–	–	–

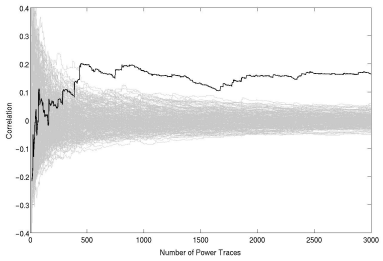

Fig. 2. CPA attack on smart card implementation of randomised `MixColumns`

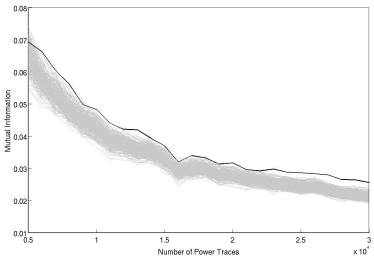

Fig. 3. MIA attack on smart card implementation of randomised `MixColumns`

7 Patching the Permutation Tables Countermeasure

In this section we propose a patch to repair the AES implementation proposed in [6], that prevents the attacks presented in the previous sections. Essentially, it consists of a new, secure implementation of the $\mathtt{XT}_8$ function. Like in the implementation proposed in [6] and recalled in Sec. 3.2, two 8-bit to 4-bit tables $\mathtt{XT}_4^1$ and $\mathtt{XT}_4^2$ are used. However, prior to their applications to the nibbles of the data u $(= P(x))$ and v $(= P(y))$ to be XORed, we add a preliminary step which aims to change the P-representation of y into a new one $P^\star(y)$. This is done by accessing the table of the function $P^\star \circ P^{-1}$. Like P, permutation $P^\star$ is defined with respect to two permutations $p_1^\star$ and $p_2^\star$ by $P^\star(x) = p_2^\star(x_h)||p_1^\star(x_l)$. For security reasons $p_1^\star$ and $p_2^\star$ must be independent of p_1 and p_2 respectively. The two new tables $\mathtt{XT}_4^1$ and $\mathtt{XT}_4^2$ are then defined by:

$$\mathtt{XT}_4^1(u_l, v_l) = p_1(p_1^{-1}(u_l) \oplus p_1^{\star -1}(v_l)) \tag{23}$$

$$\mathtt{XT}_4^2(u_h, v_h) = p_2(p_2^{-1}(u_h) \oplus p_2^{\star -1}(v_h)) \tag{24}$$

Then, the new 8-bit XOR function $\mathtt{XT}_8$, is defined w.r.t. these tables such that (for $u, v \in \mathbb{F}_2^8$):

$$\mathtt{XT}_8(u, v) = \mathtt{XT}_4^2(u_h, v_h)||\mathtt{XT}_4^1(u_l, v_l) \tag{25}$$

Consequently, we have the following secure implementation of $P(x \oplus y)$ from the P-representations $u = P(x)$ and $v = P(y)$:

1. Compute $v^\star = P^\star \circ P^{-1}[v]$,
2. Process $q_l = \mathtt{XT}_4^1(u_l, v_l^\star)$
3. Process $q_h = \mathtt{XT}_4^2(u_h, v_h^\star)$

4. Output $\mathtt{XT}_8(u, v) = q_h||q_l$.

Since p_1 and p_2 are independent permutations, we can choose $P^\star = p_1||p_2$ without losing security against first-order SCA. Namely, $P^\star$ can be simply defined from $P = p_2||p_1$ by reversing the roles of p_1 and p_2. With such a construction, our patch does not require further random permutations, other than those already involved in the definition of P, to be designed. Moreover, computing $P^\star \circ P^{-1}[v]$ amounts in this case to processing $p_1 \circ p_2^{-1}(v_h)$ and $p_2 \circ p_1^{-1}(v_l)$. When compared to the $\mathtt{XT}_8$ implementation proposed in [6], our secure implementation requires a single additional access to a look-up table and two additional 16-byte tables to store the functions $p_2 \circ p_1^{-1}$ and $p_1 \circ p_2^{-1}$. In fact, $p_2 \circ p_1^{-1}$ and $p_1 \circ p_2^{-1}$ correspond exactly to the tables p_{21} and p_{12} that are proposed in [6] as part of a time-memory trade-off in the randomised AES implementation. Therefore, if such a trade-off is already being used, additional memory is not required by our patch.

8 Conclusion

In this paper, we have shown that first-order flaws exist in the permutation tables countermeasure proposed in [6]. In order to exploit this leakage, two attacks have been developed. The first attack applies the recent work of [18] to develop an optimal prediction function for use in a correlation-based attack. The second attack is based on mutual information analysis, and uses estimation methods proposed by [20]. The new attacks were verified in both simulation and practice. In the extended version of this paper [17], we suggest a patch for the permutation tables countermeasure, thereby removing the first-order leakage. It is interesting to note that even if the permutation tables countermeasure is flawed, exploiting this flaw requires more traces than, for instance, an attack on a flawed masking scheme. Therefore, an avenue for further research is to examine the HO-SCA resistance of the (patched) permutation tables countermeasure, as it may also be more HO-SCA resistant than masking.

Acknowledgements

The authors would like to thank Jean-Sébastien Coron of the University of Luxembourg, and the anonymous reviewers of CHES 2009 for their helpful comments. This work was sponsored in part by Enterprise Ireland grant number EI PC/2008/13.

References

1. Mehdi-Laurent Akkar and L. Goubin. A Generic Protection against High-order Differential Power Analysis. In T. Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 192–205. Springer, 2003.
2. S. Aumonier. Generalized Correlation Power Analysis. Published in the Proceedings of the Ecrypt Workshop Tools For Cryptanalysis 2007, 2007.
3. J. Blömer, J. G. Merchan, and V. Krummel. Provably Secure Masking of AES. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.
4. 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.
5. S. Chari, C.S. Jutla, J.R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In M.J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
6. J.-S. Coron. A New DPA Countermeasure Based on Permutation Tables. In R. Ostrovsky, R. De Prisco, and I. Visconti, editors, *Security and Cryptography for Networks, 6th International Conference, SCN 2008*, volume 5229 of *Lecture Notes in Computer Science*, pages 278–292. Springer, 2008.
7. J.-S. Coron, C. Giraud, E. Prouff, and M. Rivain. Attack and Improvement of a Secure S-Box Calculation Based on the Fourier Transform. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems — CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2008.
8. J.-S. Coron, E. Prouff, and M. Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 2007.
9. G. Fumaroli, E. Mayer, and R. Dubois. First-Order Differential Power Analysis on the Duplication method. In K. Srinathan, C. Pandu Rangan, and M. Yung, editors, *INDOCRYPT 2007*, volume 4859 of *Lecture Notes in Computer Science*, pages 210–223. Springer, 2007.
10. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual information analysis. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems — CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
11. J. Golić and C. Tymen. Multiplicative Masking and Power Analysis of AES. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002.
12. M. Joye, P. Paillier, and B. Schoenmakers. On Second-order Differential Power Analysis. In J.R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

13. P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In N. Koblitz, editor, *Advances in Cryptology — CRYPTO '96, 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
14. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
15. T.S. Messerges. Using Second-order Power Analysis to Attack DPA Resistant software. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.
16. G. Piret and F.-X. Standaert. Security Analysis of Higher-Order Boolean Masking Schemes for Block Ciphers (with Conditions of Perfect Masking). *IET Information Security*, 2(1):1–11, March 2008.
17. E. Prouff and R. McEvoy. First-Order Side-Channel Attacks on the Permutation Tables Countermeasure — Extended Version. To appear on the Cryptology ePrint Archive, http://eprint.iacr.org, 2009.
18. E. Prouff, M. Rivain, and R. Bévan. Statistical Analysis of Second Order Differential Power Analysis. *IEEE Transactions on Computers*, 58(6):799–811, June 2009.
19. Emmanuel Prouff and Matthieu Rivain. A Generic Method for Secure SBox Implementation. In Sehun Kim, Moti Yung, and Hyung-Woo Lee, editors, *WISA*, volume 4867 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2008.
20. Emmanuel Prouff and Matthieu Rivain. Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In P.-A. Fouque M. Abdalla, D. Pointcheval and D. Vergnaud, editors, *Applied Cryptography and Network Security – ANCS 2009*, volume 5536 of *Lecture Notes in Computer Science*, pages 499–518. Springer, 2009.
21. T. T. Soong. *Fundamentals of Probability and Statistics for Engineers*. John Wiley & Sons, Ltd., 3rd edition, 2004.
22. J. Waddle and D. Wagner. Toward Efficient Second-order Power Analysis. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.

A Derivation of f_{opt} for CPA attacks

This section aims at deriving analytical expressions for the function f_{opt}. We begin with the expression $f_{opt}(x,y) = \mathrm{E}\left[(L - \mathrm{E}\left[L\right])^o \mid X = x, Y = y\right]$ (Eq. (17)), where for clarity reasons and because there is no ambiguity, we use the notation L in place of $L(k_l)$. We recall that the random variable L is assumed to satisfy $L = \hat{\varphi}(Z) + B$ and that Z equals $P_1(X)||P_1(Y)$ with $P_1 \sim \mathcal{U}(\mathcal{P})$. Since the expectation is linear and the random variables B and (X,Y) are independent, developing $(L - \mathrm{E}\left[L\right])^o$ leads to:

$$f_{opt}(x,y) = \mathrm{E}\left[(\hat{\varphi}(Z) - m)^o \mid (X,Y) = (x,y)\right] + \mu_o + \sum_{i=1}^{o-1} \binom{o}{i} \mu_{o-i} \mathrm{E}\left[(\hat{\varphi}(Z) - m)^i \mid (X,Y) = (x,y)\right] \quad , \quad (26)$$

where m denotes the mean $\mathrm{E}\left[\hat{\varphi}(Z)\right]$ and μ_i denotes the ith order central moment of $B \sim \mathcal{N}(\varepsilon, \sigma^2)$. Let us notice that since μ_1 is zero, the sum in (26) can start from $i = 2$.

Example 1. For o equal to 1 and 2, we respectively have:

$$f_{opt}(x, y) = \mathrm{E}\left[\hat{\varphi}(Z) - m \mid (X, Y) = (x, y)\right]$$

and

$$f_{opt}(x, y) = \mathrm{E}\left[(\hat{\varphi}(Z) - m)^2 \mid (X, Y) = (x, y)\right] + \mu_2 \ .$$

The prediction function given in (18) corresponds to the development of the terms in (26) that do not depend on noise parameters. It must be noticed that in the cases $o = 1$ and $o = 2$, such an estimation of f_{opt} is perfect since the terms that depend on noise parameters are either null or constant.

Henceforth, we assume that $\mathcal{P}$ is the set of all permutations over $\mathbb{F}_2^n$ and that P_1 is a random variable with uniform distribution over $\mathcal{P}$. This assumption is very favorable to the permutation table countermeasure since it implies that the choice of the masking permutation P_1 is not reduced to a sub-class of the set of permutations over $\mathbb{F}_2^n$.

We now focus on the non-noisy term in (26), namely on the mean $\mathrm{E}\left[(\hat{\varphi}(Z) - m)^o \mid (X, Y) = (x, y)\right]$. Moreover, we denote this conditional mean by $g(x, y)$, and define $\delta_x(y)$ s.t. $\delta_x(y) = 1$ if $y = x$ and $\delta_x(y) = 0$ otherwise (resp. $\overline{\delta_x}(y) = 1 - \delta_x(y)$). We have the following Lemma:

Lemma 2. *Let X and Y be two random variables with uniform distributions over $\mathbb{F}_2^n$ and let P_1 be a random variable uniformly distributed over the set of all permutations over $\mathbb{F}_2^n$. Then for every $\hat{\varphi}$ the function g is 2-valued and satisfies:*

$$g(x, y) = \frac{2^n \delta_x(y) - 1}{2^n - 1} E\left[(\hat{\varphi}(I||I) - m)^o\right] + \frac{2^n \overline{\delta_x}(y)}{2^n - 1} E\left[(\hat{\varphi}(I||J) - m)^o\right] \ , \tag{27}$$

where I and J are two independent random variables with uniform distribution over $\mathbb{F}_2^n$.

Proof. For every $(x, y) \in \mathbb{F}_2^{2n}$ we have

$$g(x, y) = \sum_{i,j \in \mathbb{F}_2^n} (\hat{\varphi}(i||j) - m)^o \mathrm{p}\left[P_1(x) = i, P_1(y) = j\right] \ .$$

Since P_1 is assumed to have uniform distribution over the set of permutations over $\mathbb{F}_2^n$, for every $(x, y) \in (\mathbb{F}_2^n)^2$ s.t. $x \neq y$ we have:

$$\mathrm{p}\left[P_1(x) = i, P_1(y) = j\right] = \begin{cases} 1/2^n(2^n - 1) & \text{if } i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

If $x = y$, we have

$$\mathrm{p}\left[P_1(x) = i, P_1(y) = j\right] = \begin{cases} 1/2^n & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Combining (28) and (29) gives (27).

When the estimation $\hat{\varphi}$ is the Hamming weight over $\mathbb{F}_2^{2n}$, (27) can be further developed. Indeed, in this case we have:

$$g(x, y) = \frac{2^n\delta_x(y) - 1}{2^n - 1} 2^o \mathrm{E}\left[\left(\mathrm{HW}(I) - \frac{n}{2}\right)^o\right] + \frac{2^n\overline{\delta_x}(y)}{2^n - 1} \mathrm{E}\left[(\mathrm{HW}(I||J) - n)^o\right] \ ,$$

since m equals $\mathrm{E}\left[\mathrm{HW}\right]$, i.e. n when HW is defined over $\mathbb{F}_2^{2n}$.

As $\mathrm{E}\left[\mathrm{HW}(I)\right]$ equals $\frac{n}{2}$ and $\mathrm{E}\left[\mathrm{HW}(I||J)\right]$ equals n, the function g is constant equal to 0 when $o = 1$. For $o = 2$, it satisfies:

$$g(x, y) = \delta_x(y)\frac{n2^{n-1}}{2^n - 1} + \frac{n(2^{n-1} - 1)}{2^n - 1} \ , \tag{30}$$

since we have $\mathrm{E}\left[(\mathrm{HW}(I) - \frac{n}{2})^2\right]$ (resp. $\mathrm{E}\left[(\mathrm{HW}(I||J) - n)^2\right]$) equal to $\mathrm{Var}\left[\mathrm{HW}(I)\right] = \frac{n}{4}$ (resp. $\mathrm{Var}\left[\mathrm{HW}(I||J)\right] = \frac{n}{2}$).

For $o = 2$, (30) implies that f_{opt} is an affine increasing function of $\delta_x(y)$. Since the correlation coefficient is invariant for any affine transformation of one or both of its parameters, the function $x, y \mapsto \delta_x(y)$ itself (and every affine transformation of it) is actually an optimal prediction function for $o = 2$. Hence, in its simplest form the optimal function for $o = 2$ is defined for every $(x, y) \in \mathbb{F}_2^{n\,2}$ as:

$$f_{opt}(x, y) = \delta_x(y) \ . \tag{31}$$

For $o = 2$, the function f_{opt} in (31) can be applied to conduct CPA attacks in the particular case of Coron's construction of $\mathcal{P}$ (which is not the set of all permutations over $\{0, ..., 15\}$ but a subset of it with cardinality 16^4), without losing a significant factor in attack efficiency (in terms of number of leakage measurements).

B On the Notion of Order in SCA

B.1 Brief History and Discussion

Since the original work of Messerges in [15], Higher Order SCA have been widely studied. The notion of HO-SCA itself has been refined so that we nowadays have a better understanding of the theoretical foundations behind them. According to Messerges's definition, a d-th order SCA was an attack that made use of d different instants in the power consumption measurement that correspond to d different intermediate values calculated during the execution of an algorithm. However, this definition is not sufficient alone to determine unambiguously the order of an attack. Indeed, in Messerges's definition the order of a HO-SCA relies not only on the number of times targeted by leakage measurement but also on the number of intermediate results. So, both of those numbers must be equal to d for the attack to be qualified of d-th order. Whereas it is straightforward to determine the number of times targeted by an attack, determining the number of targeted intermediate results imposes, as argued by Blömer *et al* in [3], to formally define the notion of *intermediate result*. Indeed, to convince us about this fact, let us consider the case of the result $Z = (Z_1, \cdots, Z_n)$ of a calculation $\mathcal{C}$ appearing in the theoretical description of an algorithm. If Z is computed by hardware means then each of its coordinates is the result of a different hardware computation, so that one can consider that the time t when Z is computed is associated with n different intermediate results. On the other hand, if $\mathcal{C}$ is implemented in software on a n-bit architecture and by accessing a look-up table, then all the coordinates of Z are processed in the same way and thus the time t when Z is computed can only be associated with one different intermediate result (which is Z itself). Actually, in [3], Blömer *et al* argued that the nature of an intermediate variable depends on the context in which the algorithm is analyzed. When it is analyzed in the context of a software implementation on a r-bit architecture, an intermediate result is an r-bit long result or input of an intermediate calculation; in such a context, the notion of intermediate result refers to every r-bit long variable which is manipulated during the processing of the algorithm. When the algorithm is analyzed in the context of an hardware implementation, then each bit manipulation results in and makes use of an intermediate binary result; in such a context, the notion of intermediate result refers to every bit (or 1-bit long variable) which is manipulated during the processing of the algorithm. In fact, the work of Blömer *et al* showed that the order of an HO-SCA depends on the context in which the attack is performed. Nowadays, most researchers

distinguish the hardware case from the software case when defining the notion of HO-SCA order [1, 3, 8, 12, 19, 16]. However, it must be noted that in all the definitions, the core idea is that a d-th order HO-SCA is an attack that exploits d different times per leakage measurement. The contextual definitions only differ in the way how the intermediate result is defined (according to the context).

B.2 Different notions of SCA Order in the Literature

Hereafter, we list several different definitions of the notion of order for a Side Channel Attack.

Messerges [15] According to Messerges's definition, a d-th order SCA is an attack that make use of d different times in the power consumption measurement that correspond to d different intermediate values calculated during the execution of an algorithm. In some situations, this definition does not allow us to determine unambiguously the order of an attack. Indeed, for this definition, the order of an HO-SCA relies not only on the number of times targeted by leakage measurement but also on the number of intermediate results. So, both of those numbers must be equal to d for the attack to be qualified of d-th order. Whereas it is straightforward to determine the number of times targeted by an attack, determining the number of targeted intermediate results imposes, as argued by Blömer *et al* in [3], to formally define the notion of *intermediate result*.

Blömer et al. [3] According to Blömer *et al.*'s definition, a d-th order SCA is an attack that makes use of d different intermediate values calculated during the execution of an algorithm. In [3], it is argued that the nature of an intermediate variable depends on the context in which the algorithm is analyzed. Depending on the context (algorithmic description, software implementation, hardware implementation, etc.) several definitions of an *intermediate variable* are given.

Waddle and Wagner [22] According to Waddle and Wagner's definition, a d-th order SCA is an attack exploiting d shares (viewed as intermediate variables) simultaneously. This definition is not time-related. Moreover, the notion of share or of intermediate variable is not formally defined. It seems that for Waddle and Wagner's definition, an attack is of order d if it targets a masked variable and $(d-1)$ masks.

Other possible definitions

- **Statistical Definition.** Let $\mathbf{L}$ be a vector of d leakage measurements (not necessarily related to variables manipulated at different times). Let k denote the secret parameter the attack aims to recover from $\mathbf{L}$ and let $\mathcal{K}$ denote its definition set. The order of a SCA exploiting $\mathbf{L}$ is d if the d-th order statistical moments of the distributions of the random variables of the leakages $\mathbf{L}(k) = (\mathbf{L} \mid K = k)$ differ when k ranges over $\mathcal{K}$ whereas the $(d-1)$-th order statistical moments do not.
- **Spatial** definition. The order of a SCA targeting Z is d if it involves d leakage points leaking at different locations in space in the device.