Combined Fault and Side-Channel Attack on Protected Implementations of AES

Thomas Roche, Victor Lomné, and Karim Khalfallah

ANSSI, 51, Bd de la Tour-Maubourg, 75700 Paris 07 SP, France
firstname.lastname@ssi.gouv.fr

Abstract. The contribution of this paper is twofold: (1) a novel fault injection attack against AES, based on a new fault model, is proposed. Compared to state-of-the-art attacks, this fault model advantage is to relax constraints on the fault location, and then reduce the a priori knowledge on the implementation. Moreover, the attack algorithm is very simple and leaves room for optimization with respect to specific cases; (2) the fault attack is combined with side-channel analysis in order to defeat fault injection resistant and masked AES implementations. More precisely, our fault injection attack works well even when the attacker has only access to the faulty ciphertexts through a side-channel. Furthermore, the attacks presented in this paper can be extended to any SP-Network.

Keywords: AES, SCA, DPA, DFA, Side-Channel Analysis, Fault Attacks, masking scheme, combined attack.

1 Introduction

In a hostile environment, devices embedding cryptographic algorithms are susceptible to so-called *physical* attacks, namely Side-Channel Analysis (SCA) and Fault Attacks (FA). To recover the key, SCA makes use of physical leakages emanating from a device (power consumption, electromagnetic radiations, ...) while it performs a cryptographic operation. FA exploit logical error(s) induced by the adversary on a device running a cryptographic operation to retrieve the secret.

These two classes of attacks have been widely studied these last years, and countermeasures have been suggested to thwart them. As a first contribution, we propose to study the security of block ciphers, more precisely SP-Networks, against a new type of FA. We will show that in many implementations, it is possible to relax the state-of-the-art fault models of FA. Our second contribution is to attack an implementation where countermeasures against FA and SCA have been embedded such that the faulty outputs are only accessible through side-channel leakage. To that purpose we combine our FA with a classical 1st-order SCA. This second contribution is the true goal of this paper, the FA was indeed designed to this end. We remarked that in order to make use of a statistical analysis such as SCA on faulty computations, one must be able to repeat the same fault several times. Interestingly enough, this property led to a simple and

E. Prouff (Ed.): CARDIS 2011, LNCS 7079, pp. 65–83, 2011.

generic FA attack that straightforwardly combines itself with SCA. The attacks proposed in this article will be presented on the Advanced Encryption Standard (AES) [2], because it is the most studied SP-Network, but they can be applied to any SP-based cipher.

The paper is organized as follows: Section 2 describes previous works on SCA, FA, and their countermeasures. A brief summary on combined attacks is given. Section 3 defines some notations used in the rest of the document. Section 4 describes our new fault attack against AES. Section 5 explains our combined fault and side-channel attack against FA- and SCA-resistant implementations of AES. Possible countermeasures against this attack are given in Section 6. Finally, we conclude and give future directions of this study in Section 7.

2 Previous Work

In this section, we briefly present the main SCA and FA against block cipher implementations and the sound countermeasures that have been introduced in the literature. The combination of such countermeasures should thwart both SCA and FA. The idea of combining SCA and FA in order to create a more powerful attack is not new, we will list the previous combined attacks and discuss their strength against secure implementations.

2.1 SCA and Masking

The observation of a block cipher implementation through a *side-channel* (*e.g.* power consumption, electromagnetic radiations, timing, etc...) has been shown to give information about intermediate variables manipulated by the block cipher.

SCA attacks make use of leaked information about intermediate variables that are dependent on the unknown secret key and the known plaintext (we will refer to *sensitive variables* in the sequel). SCA attacks have been first introduced by Kocher *et al.* in their seminal paper describing Simple Power Analysis (SPA for short) and Differential Power Analysis (DPA for short) [22]. Since then, SCA has become a real threat to security devices. DPA is especially powerful because of its robustness to noise, and has been applied to many block ciphers. Moreover, several variants were proposed to enhance the attack success. Let us just cite the main improvements of the statistical distinguisher: CPA [10] and MIA [17]. In the following, we use the term Differential Side-Channel Analysis (DSCA for short) for attacks based on DPA, CPA, MIA or other variants.

Principle of DSCA: let us consider the sensitive variable s, depending on both the unknown secret key k and the known plaintext p. When s is computed by the device, some information on s is leaked and may be captured by the adversary. We denote by $L(s)$ this information leakage. The *leakage function* $L()$ is composed of a deterministic part ϕ and some noise $\mathfrak{B}$ (that includes both algorithmic and electronic noise):

$$L(s) = \phi(s) + \mathfrak{B} \tag{1}$$

Note that $\mathfrak{B}$ is generally assumed to be close to a Gaussian noise of mean μ and standard deviation σ: $\mathfrak{B} \sim \mathcal{N}(\mu, \sigma)$. In DSCA, the adversary tries to correlate the leaked information $L(s)$ with a prediction $\widehat{L(s)}$ of the behavior of $L(s)$ for various values of p. The predictions are dependent on the choice of ϕ (typically the Hamming weight function, HW for short) and a hypothesis on k. For a given choice of ϕ, the right hypothesis on k is expected to lead to the highest correlation.

The threat of DSCA-like attacks led to the research of sound and efficient countermeasures. Among them, the so-called *masking schemes* constitute the only family of countermeasures for which formal proofs of security have been given [11]. A masking scheme consists in randomizing the manipulated intermediate variables, such that the side channel observations do not yield useful information about sensitive variables.

To illustrate the basic idea, we consider a Boolean masking scheme of order d applied on a function f. Let s be a sensitive variable depending on p and k, let $m_1, \cdots, m_d$ be the d random input masks, and let $m'_1, \cdots, m'_d$ be the d random output masks. The variable s is then randomized in d shares, as follows: $s \oplus m_1 \oplus \cdots \oplus m_d, m_1, \cdots, m_d$. So the function f has to be transformed into a new function f' such that:

$$f'(s \oplus m_1 \oplus \cdots \oplus m_d, m_1, \cdots, m_d) = f(s) \oplus m'_1 \oplus ... \oplus m'_d \tag{2}$$

At the end of a cipher encryption, one has just to unmask the result with the d masks to get the ciphertext. Several methods have been proposed in the literature to implement such a masking scheme, for different values of d, for instance [3, 20, 22, 26, 28].

Nevertheless, a d^{th}-order masking scheme is susceptible to a $(d+1)^{\text{th}}$-order differential side-channel attack, initially introduced in [23], which consists in combining $(d+1)$ side-channel leakages, *e.g.* the one where the masked sensitive variable is handled and those where the different masks are manipulated. It has to be noted that high-order side-channel attacks become impracticable as the order increases, mainly for two reasons:

- when combining the different side-channel leakages, the resulting noise increases exponentially with d [11];
- the adversary has to guess the instants where the different side-channel leakages occur, which, in the absence of truly efficient detection algorithms, implies an exponential increase of the computation time of the attack with d.

2.2 DFA and Redundancy

The first use of logical faults to break a cipher appeared in a paper of the Bellcore team in 1997 [9]. They showed how a unique fault could be used to break a RSA implementation. The next year, Biham and Shamir introduced the concept of Differential Fault Analysis (DFA) [7] on the Data Encryption Standard (DES) [1]. This concept has been applied later on SP-Networks (mainly on AES), through different attacks [8, 12, 16, 18, 21, 25, 27].

Among them, the attack of Piret *et al.* [27] is probably the most powerful. It allows to retrieve an AES key with only one pair of correct/faulty ciphertexts and a computation time of about 2^{40} (note that improvements have been proposed to reduce this offline phase [19, 32]), when the adversary can induce a fault on a single byte of the state before the penultimate `MixColumn`.

The most popular countermeasure against DFA consists in spatial or temporal redundancy, as explained in [6]. The idea is to compute at least two times the cryptographic operation, and to compare the obtained results. The device provides the result of the cryptographic operation only if the different results are equal. Several variants exist:

- one can (temporally or spatially) duplicate only the last rounds of the block cipher, as most of the DFA focus on inducing faults on the last rounds of block ciphers;
- one can encrypt the plaintext, keep the obtained ciphertext in the device, decrypt the ciphertext and compare the new plaintext with the original one; thus the ciphertext is output only if both are equal.

2.3 Combined Attacks and Combined Countermeasures

Recently, different works proposed to exploit SCA and FA together to develop more powerful attacks. In [30], the author uses a laser beam to increase the power consumption of a micro-controller logic cell and exploits this phenomenon via power analysis. In [29], the authors introduce the Differential Behavioral Analysis (DBA), which combines Safe Error Attack (SEA) and DPA. The DBA requires the adversary to be able to induce a fault during several encryptions. Moreover the fault has to be repeatable, must affect a small number of bits (less than 8) and is also expected to induce a fixed (possibly unknown) *stuck-at* value. The authors show in simulation that the DBA can break an AES hardware implementation, but one has to note that the DBA is ineffective on a masked implementation.

The same year, [4] introduced a Passive and Active Combined Attack (PACA) on Public-Key Cryptography. Later, the same idea was applied on AES in [14]. The fault also assumes a *stuck-at* model. It is shown how a PACA can defeat a masked implementation of AES. Nevertheless, it has to be noticed that this attack can only break a first order masking scheme of AES. In the case of an AES implementation masked at order equal or greater than 2, this PACA just reduces the DSCA of one order.

Thus, to our knowledge, no successful practical side-channel attack has been reported in the literature targeting a masking scheme of AES at order equal or greater than 2. One can then consider that an AES implementation with a masking scheme at order at least 2, with one of the DFA countermeasures cited in Section 2.2, is protected against both state-of-the-art SCA and FA. In Section 5, we show that this is not true if the redundancy check of the fault injection countermeasure manipulates unmasked ciphertexts (or plaintexts).

3 Notations

In the following, we will denote respectively by P, C, K, Z and E the plaintext, the ciphertext, the secret key, the targeted intermediate value and the error. All these values are over 16 bytes. We will frequently use their vector representation in $(\mathrm{GF}(2^8))^{16}$ (*e.g.* $P = (P_0, \cdots, P_{15})$).

We will consider the AES cipher as an example to describe the attacks, but all of them can be straightforwardly extended to any SP-Network. As a consequence, we will use the common AES sub-function names (*e.g.* see [2]) to denote either the non-linear layer (`SubByte`) or the linear layer (`ShiftRow` and `MixColumn`) of the cipher. Moreover the different size parameters will correspond to the AES-128's, involving 10 rounds with 128 bits master key, and 11 round keys (denoted $K^0, \cdots, K^{10}$).

The attacks presented here will mainly involve fault injection in the last round of AES, where only the `SubByte`, `ShiftRow` and the last `AddRoundKey` operation appear. Without loss of generality we omit the last `ShiftRow` operation, as its only effect would be to render indices unreadable.

We will use the so called *correlation* side-channel analysis, based on the Pearson correlation coefficient. Let $\rho(X, Y)$ denote the correlation between the two random variables X and Y, we recall that it is defined by:

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma(X).\sigma(Y)}$$

4 A New DFA

In this section we present a new perturbation attack on AES. We first consider a FA-unprotected AES implementation.

We will consider several types of AES implementations, masked or not, and propose a DFA attack based on more or less restrictive fault models. Many types of faults can be investigated; in practice, the apparition of one kind of fault or another is strongly dependent on the means of the fault injection (laser beam, glitches, etc...) and to the hardware target (micro-controller, ASIC or FPGA implementations). We believe that the fault model considered in some of these scenario is much less constraining than any other proposed by now. In the next section, those attacks will be combined to SCA in order to defeat a FA resistant and masked implementation.

4.1 Best Case Scenario: Key Schedule Pre-computation

For sake of clarity, let us first consider the target implementation where our DFA attack accepts the widest fault model. The (hardware or software) implementation is as follows:

- the implementation possesses no countermeasure against DFA;
- the secret key is stored in (*e.g.* non-volatile) memory;

– at each reset of the device (that an attacker has control upon, this can correspond to simply powering off the device), the key schedule is run and the round keys are stored in volatile memory and will be used for any later ciphering/deciphering execution.

Remark that the implementation may include SCA countermeasures.

Fault Model. The attack will target the key-scheduling algorithm. The fault shall be transient (*i.e.* future computations shall not be affected) and affect the key schedule in its last but one round. The fault could be of any kind, short to destroy or stop the device. The wanted effect is to have the last but one round key incorrect (and, as a side effect, also the last round key). Furthermore the fault can affect any part of the last two round keys in any possible way, for each affected byte of the next to last round key the attack will recover one of the last round key byte (see Remark 2 below).

When the fault injection is successful, the affected last two round keys will be written as the XOR of the valid round keys and an error:

$$\begin{aligned} \widetilde{K}^9 &= K^9 \oplus E^9 \\ \widetilde{K}^{10} &= K^{10} \oplus E^{10} \end{aligned} \tag{3}$$

Attack Description

1. Encrypt N messages $P^1, \cdots, P^N$.
2. Reset the device and inject an error in the key-schedule.
3. Encrypt $P^1, \cdots, P^N$ once more[1].
4. It produces N pairs of valid-faulty ciphertexts $(C^1, \widetilde{C}^1), \cdots, (C^N, \widetilde{C}^N)$.
5. For each byte index j of the ciphertexts, process separately:
6. For each hypothetic value $(e_9, e_{10}, k) \in \left(\mathbb{F}_{2^8}\right)^3$ of the error bytes $E_j^9 = e_9$ and $E_j^{10} = e_{10}$ respectively on the 9^{th} and 10^{th} round key, and the subkey byte $K_j^{10} = k$ (of the last round key) create a counter, denoted by $T_{e_9,e_{10},k}$. For each pair of valid-faulty ciphertext $(C^i, \widetilde{C}^i)$, do the following:
7. Increment the counter $T_{e_9,e_{10},k}$ if

$$\texttt{SubByte}\left(\texttt{SubByte}^{-1}(C_j^i \oplus k) \oplus e_9\right) \oplus k \oplus e_{10} = \widetilde{C}_j^i$$

8. Find the triplet of error and subkey bytes that led to the highest counter. If the counter is N, then mark the triplet as right. Otherwise the fault injection was not successful.
9. The last round key is retrieved from the different marked subkey bytes when the corresponding error byte e_9 is non-zero.

The complexity of the attack is in $O(N(2^8)^3)$ for N faulty ciphertexts.

[1] In all the sequel, the only constraint on the plaintexts is that they must be used twice, they do not have to be known.

Why the Attack Works. The main idea in the attack above is the ability to inject an error such that any future cipher execution will be impacted by the same error. More precisely, we are here artificially building an unknown but fixed difference in the last round of the cipher. As a matter of fact, for any $0 \leq j \leq 15$, the difference between $\texttt{SubByte}^{-1}(C_j^i \oplus K_j^{10})$ and $\texttt{SubByte}^{-1}(\widetilde{C}_j^i \oplus \widetilde{K}_j^{10})$ is fixed over all $i \leq N$ and equal to E_j^9.

Let us first assume that the fault injection was successful and that only the last two round keys are affected with respectively the error E^9 and E^{10}. After the execution of the attack algorithm, for any $0 \leq j \leq 15$, the counter $T_{E_j^9, E_j^{10}, K_j^{10}}$ is trivially equal to N. On the other hand the probability for a triplet $(e_9, e_{10}, k) \neq (E_j^9, E_j^{10}, K_j^{10})$ to have $T_{e_9,e_{10},k} = N$ in step 8 of the attack algorithm is about $\left(\frac{1}{256}\right)^N$. This result is only a rough estimation considering the AES cipher. The probability is actually dependent on the S-box uniform differentiability, the better the S-box is (in term of resistance against differential attacks), the better the DFA will work[2]. Our simulations confirm that 3 pairs of valid-faulty ciphertexts are enough to get a success rate superior to 90%.

If we consider now that the fault injection impacted more than the last two round keys (*i.e.* the fault model is violated). Then the difference between $\texttt{SubByte}^{-1}(C_j^i \oplus K_j^{10})$ and $\texttt{SubByte}^{-1}(\widetilde{C}_j^i \oplus \widetilde{K}_j^{10})$ (denoted Δ_j^i) is not fixed anymore. This does not change anything for the wrong triplets $(e_9, e_{10}, k) \neq (E_j^9, E_j^{10}, K_j^{10})$, their corresponding counter will have the same probability to reach N. By opposition, the right counter $T_{E_j^9, E_j^{10}, K_j^{10}}$ will have a lower probability to reach N. This probability is in fact dependent on the distribution of the differences Δ_j^i for a fixed j (*e.g.* when the distribution is uniform, the probability to reach N is equal to the other counters).

Remark 1. In the case where the fault model is not respected but still the difference Δ_j^i is fixed with a biased probability, it is possible to adapt the attack algorithm in order to retrieve the key. Indeed, a non-uniform distribution in the Δ_j^i will eventually be distinguished (when N grows) by looking at the $T_{e_9,e_{10},k}$ distribution.

Remark 2. Let us note that if the byte K_j^9 of K^9 is not affected by the fault injection, then the corresponding byte K_j^{10} will not be retrieved by our attack. Hence, the wider the fault, the better the attack. This is in opposition with classical DFA where the fault model restricts the fault size.

The attack described here assumed that the key schedule was pre-computed, let us now consider cipher implementations where the key schedule is re-executed for each cipher run.

4.2 Key Schedule Re-executed at Each Cipher Execution

In the setup where the key schedule is re-computed at each cipher execution, the attack algorithm will have to be modified, now N fault injections being necessary

[2] This is a classical observation in DFA [27].

to get N faulty ciphertexts. The fault model will also have to be modified, two cases can be distinguished whether the key schedule implementation uses masking countermeasures or not. Eventually, in a fully masked implementation, the fault model will be severely restricted.

Unmasked Key Schedule Implementation. In the case of an unmasked key schedule implementation, our attack still targets the key schedule algorithm. The main difference with the attack proposed in previous Section is the fact that a unique error will not be enough anymore to compute N faulty ciphertexts. For each faulty ciphertext computation, the fault injection must be re-applied. As a consequence, the fault injection must possess a new property: a good *repeatability* (*i.e.* two injected faults have a good chance to induce the same error). Indeed, if we consider that, with an identical fault injection setup on the key schedule computation of two cipher executions on the same key, the injected fault is always the same, then the attack here will have the same success rate and same complexity than the attack described in Section 4.1. Let us insist that since the fault is injected on the key schedule, the same values are modified (at each execution the key schedule is manipulating the same intermediate variables), thus, again here, any kind of fault that affected the last two round keys can be written as the XOR of the valid round keys and an error (as in equation 3)

The attack steps are then:

1. Encrypt N messages $P^1, \cdots, P^N$.
2. Encrypt $P^1, \cdots, P^N$ once again, this time with a fault injection during the last but one round of the key schedule.
3. It produces N pairs of valid-faulty ciphertexts $(C^1, \widetilde{C}^1), \cdots, (C^N, \widetilde{C}^N)$.
4. For each byte index j of the ciphertexts, process separately:

    5. For each hypothetic value $(e_9, e_{10}, k) \in (\mathbb{F}_{2^8})^3$ of the errors byte $E_j^9 = e_9$ and $E_j^{10} = e_{10}$ respectively on the 9^{th} and 10^{th} round key, and the subkey byte $K_j^{10} = k$ (of the last round key) create a counter, denoted by $T_{e_9,e_{10},k}$. For each pair of valid-faulty ciphertext $(C^i, \widetilde{C}^i)$, do the following:

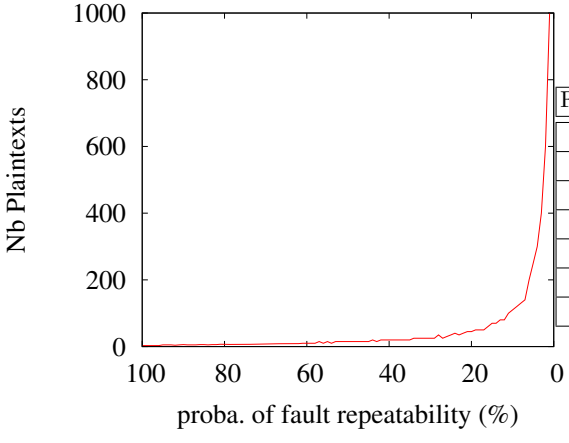
        6. Increment the counter $T_{e_9,e_{10},k}$ if

$$\texttt{SubByte}\left(\texttt{SubByte}^{-1}(C_j^i \oplus k) \oplus e_9\right) \oplus k \oplus e_{10} = \widetilde{C}_j^i$$

    7. Find the triplet of error and subkey bytes that led to the highest counter. If the counter is "high enough" compared to the others, then mark the triplet as right. Otherwise, if no triplet can be significantly distinguished from the others, the fault injection was not successful (it might either mean that the fault is not injected at the right location or that the fault repeatability is not high enough).
8. The last round key is retrieved from the different marked subkey bytes when the corresponding error byte e_9 is non-zero.

We believe that the hypothesis about fault injection repeatability is realistic, although of course a 100% repeatability seems a too strong assumption. Hence we simulated the attack (at each cipher execution, the injected fault is either fixed or random with a given probability). Figure 1 shows the exponential cost in the number of plaintexts (N) to get a 90% success rate attack as a function of the repeatability probability (the experiment has been done 100 times for each probability of fault repeatability).

Furthermore, we implemented the AES-128 on a FPGA platform and performed the described attack. We induced the faults by clock glitches during the next to last round of the key schedule and verified that the fault repeatability was good enough: the attack required 15 faulty ciphertexts to recover the 11 round key bytes that have been affected by the fault. According to the table given besides Figure 1, this means that the fault repeatability of our attack is about 50%. The setup and attack are detailed in Annex A.



Fault repeatability	Nb plaintexts
100%	3
90%	6
50%	15
10%	110
5%	250
1%	1000
0%	∞

Fig. 1. Number of valid-, faulty-ciphertext pairs for 90% success rate with respect to the probability of fault injection repeatability (in %)

Remark 3. The attack can be seen as an artificially constructed differential cryptanalysis. As a matter of fact, it works the same way and is based on the same assumption: there exists an output differential that occurs more often than the others. Hence, the classical results on differential cryptanalysis success rate can be straightforwardly applied and formalize our results here (see, for instance, [5]). Let us note here that the value of the output differential (before the last round) is unknown in our case, which does not change much, it shows that the knowledge of the existence of a good differential is what matters and the knowledge of the difference value is useless.

Remark 4. We have considered here that the key schedule is unmasked, it is interesting to add that the attack would also work if the key schedule was masked with a fixed mask (*i.e.* at each key schedule execution the same mask is used). Such fixed mask key schedule is very attractive from the performances point-of-view and yet resistant against 1^{st}-order SPA or profiled attacks on key values.

Masked Implementation. In the case where the implementation is fully masked, *i.e.* the intermediate variables of the whole cipher computation is masked with randomly generated values, the fault model will have to be drastically restricted, we still believe it is relevant when considering certain categories of hardware/injection materials (*e.g.* using a laser beam [24]). We consider here that a Boolean masking scheme is used, the masking may be of any fixed order d (see Section 2.1 for definitions and details about masking countermeasures).

Fault Model. As mentioned in the previous attack scenarios, the gist of the attack is to get pairs of valid-faulty ciphertexts with fixed difference in the last round. Let us now consider that an attacker injects the same fault in two different executions during the last but one round of the masked key schedule, for our attack to work, we would like the following equations 4 and 5 to be satisfied with fixed (a priori unknown) values E^9 and E^{10}:

$$\begin{aligned} \widetilde{K}^9 &= K^9 \oplus M^9 \oplus E^9 \\ \widetilde{K}^{10} &= K^{10} \oplus M^{10} \oplus E^{10} \end{aligned} \tag{4}$$

$$\begin{aligned} \widetilde{K}'^9 &= K^9 \oplus M'^9 \oplus E^9 \\ \widetilde{K}'^{10} &= K^{10} \oplus M'^{10} \oplus E^{10} \ , \end{aligned} \tag{5}$$

where K^9 and K^{10} are the last two round keys, M^9 and M^{10} (resp. M'^9 and M'^{10}) are the random masks of round keys K^9 and K^{10} for the first (resp. second) cipher execution. $\widetilde{K}^9$ and $\widetilde{K}^{10}$ (resp. $\widetilde{K}'^9$ and $\widetilde{K}'^{10}$) are faulty round keys for the first (resp. second) cipher execution.

Equations 4 and 5 lead to

$$\begin{aligned} \widetilde{K}'^9 \oplus \widetilde{K}^9 &= M^9 \oplus M'^9 \\ \widetilde{K}'^{10} \oplus \widetilde{K}^{10} &= M^{10} \oplus M'^{10} \end{aligned} \tag{6}$$

As these equations must be satisfied for any values of M^i and M'^i ($i \in \{9; 10\}$), it is easy to see[3] that only the XOR error function (error by bit-flip) is acceptable:

$$F_e : x \mapsto x \oplus e \ ,$$

where F_e is the *fault injection function*: $\forall X, F_e(X) = \widetilde{X}$.

[3] The first derivative of the fault injection function F_e is 1: Equation 6 implies that

$$\forall (K, M, M'), \frac{F_e(K \oplus M) \oplus F_e(K \oplus M')}{M \oplus M'} = 1.$$

In the two previous attack scenarios, we *modeled* the fault by a XOR operation; this was possible since the fault was injected on fixed values; Now the target values are varying (because of masking), to write the fault as a XOR, the fault injection needs really to flip the bits.

Assuming that the fault injection function satisfies the constraints (*i.e. bit-flip*), the attack algorithm is similar to the unmasked case, with same success rate as a function of the fault injection repeatability.

Remark 5. The fault model may seem unrealistic, much more than satisfying very good fault repeatability. In fact a trade-off can be made between those two parameters: many fault models will coincide with the *bit-flip* model for a majority of its input space. A good example is the so-called *stuck-at* fault model. Let us consider that the fault injection function is of the following form[4]:

$$F_e : x \mapsto x \& e \ ,$$

where & is the AND bit wise operator. Even though this corresponds to stuck some bits at 0, it can be seen as a *bit-flip* fault injection function with a certain probability of success (*i.e.* they coincide on a fraction of their inputs). This probability of success will add itself to the fault repeatability. Moreover the probability of success increases with the hamming weight of e. When e is the vector made of all 0 bits, no information is leaked about the value of x through the fault and the proposed attack will not be successful. In such a case of fault model (full stuck-at) we point out that other types of combined DFA-SCA attacks exist [14].

Remark 6. When assuming that the fault injection follows the *bit-flip* model (exactly or by approximation), it is equivalent to consider the fault to be injected in the key schedule or directly in the state. Therefore:

- when the key schedule is computed in parallel of the state computation, the fault can touch both parts altogether.
- if the fault targets the state only, the attacker can apply exactly the same attack algorithm as before without the prediction of the error value on the last round subkeys. This simply decreases by a factor of 2^8 the offline complexity of the attack.

5 Combined Attack Against (HO-)Masked and DFA Resistant Implementation of AES

In this section is introduced a combined attack that bypasses both higher order masking countermeasures and integrity check by the simultaneous use of faults injection and side-channel observation.

The combined attack is based on what we point out to be a weakness in the way the error detection mechanism is usually performed. Such a mechanism implies the manipulation of the unmasked faulty ciphertext (or plaintext), therefore allowing to mount a classical 1st-order DSCA.

[4] We present the case of stuck-at 0 but all the following is also true for stuck-at 1 or a mix of both.

First we come back to the different implementations that have been presented in the previous section, now integrating a DFA countermeasure by computation redundancy. Then, in Section 5.2, we propose a generic attack procedure by adding a 1st-order DSCA to the fault injection attack.

5.1 Fault Injection Resistant Implementations

We have proposed in Section 4 a new kind of DFA based on the ability to reproduce a fixed fault for different cipher executions. We presented the attack for several cipher implementations — considering the pre-computation of the key schedule algorithm and the use of masking schemes to thwart SCA attacks — and, for each of them, different fault models led to successful perturbation attacks. Namely:

- The key-schedule is pre-computed once for several cipher executions.
  - Fault Model: Single injection inside the next to last round of the key schedule computation.
- The key-schedule is unmasked (or masked with a fixed mask) and computed at each cipher execution.
  - Fault Model: Injection inside the last but one round of each key schedule computation, does not touch the state computation. The fault needs a good repeatability.
- The key-schedule is masked and computed for each cipher execution.
  - Fault Model: Injection inside the last but one round of each key schedule computation and/or the state computation. It can be modeled, with good approximation, as a *bit-flip* error and possesses good repeatability.

We will moreover assume that one of the countermeasures described in Section 2.2 against DFA has been implemented.

5.2 Combined Attack Description

The idea follows the fact that the faulty ciphertexts are not received by the adversary anymore. However he is still able to observe the computation through a side-channel, and more precisely the manipulation of the (potentially faulty) ciphertext when its integrity is checked[5].

We propose to transform the round key retrieving part of the DFA algorithms in a standard CPA attack (as presented in a generic way in Section 2.1). The attack is straightforwardly extended to fault injection resistant implementations, assuming the knowledge of good approximation of the leakage model (for purpose of clarity we will use here the Hamming weight model as presented in Section 2.1, but, any kind of leakage model could be used). The attack becomes then:

[5] We can add here that such a location in the cipher algorithm is usually easy to spot out from side-channel leakage traces as it corresponds to the end of encryption.

1. Produce N pairs of valid-faulty ciphertexts $(C^1, \widetilde{C}^1), \cdots, (C^N, \widetilde{C}^N)$ (for any of the implementations and their fault models described in Section 4). The faulty ciphertexts are not returned by the chip, the attacker has to record the side-channel traces during the faulty computations (by monitoring the power consumption, the electromagnetic radiation or any exploitable side-channel). We will consider that for each trace Ω_i, there is a known instant t_i such that $\Omega_i[t_i]$ is the side channel observation of the faulty ciphertext $\widetilde{C}^i$ manipulation. Hence, according to the notations introduced in Section 2.1, we have $\Omega_i[t_i] = \phi(s) + \mathfrak{B}$, where s is the value manipulated at time t_i (*i.e.* $\widetilde{C}^i$ or a subpart of it depending on the data register size), ϕ is a deterministic leakage function (*e.g.* Hamming Weight function) and $\mathfrak{B}$ denotes the noise (*e.g.* a Gaussian noise with mean μ and standard deviation σ).
2. For each byte index j of the ciphertexts (included in s), process separately:

    3. For each hypothetic value (e_9, e_{10}, k) of the errors byte $E_j^9 = e_9$ and $E_j^{10} = e_{10}$ respectively on the 9^{th} and 10^{th} round key, and the subkey byte $K_j^{10} = k$ compute the correlation value $\rho_{e_9,e_{10},k}$:

        4. For each ciphertext C^i, compute the *prediction* values:

$$pred_i = HW(\texttt{SubByte}\left(\texttt{SubByte}^{-1}(C_j^i \oplus k) \oplus e_9\right) \oplus k \oplus e_{10}) \ .$$

        Then compute the Pearson correlation coefficient between the predictions and the observations (the function $\rho(,)$ is recalled in Section 3):

$$\rho_{e_9,e_{10},k} = \rho(\{pred_i\}_{1 \le i \le N}, \{\Omega_i[t_i]\}_{1 \le i \le N})$$

    5. Find the triplet of error and subkey bytes that led to the highest correlation. If the correlation value is "high enough" compared to the others, then mark the triplet as right. Otherwise, if no triplet can be significantly distinguished from the others, the fault injection was not successful (it might either mean that the fault is not injected at the right location or that the fault repeatability is not high enough with respect to the noise and the number of samples).
6. The last round key is retrieved from the different marked subkey bytes when the corresponding error byte e_9 is non-zero.

The success rate of the attack described above is dependent on the noise in the side-channel measure, the repeatability of the fault injection and, of course, the number of plaintexts (N). A heuristic evaluation of the success rate as a function of these different parameters is given in the next section.

5.3 Evaluation of the Attack Success Rate

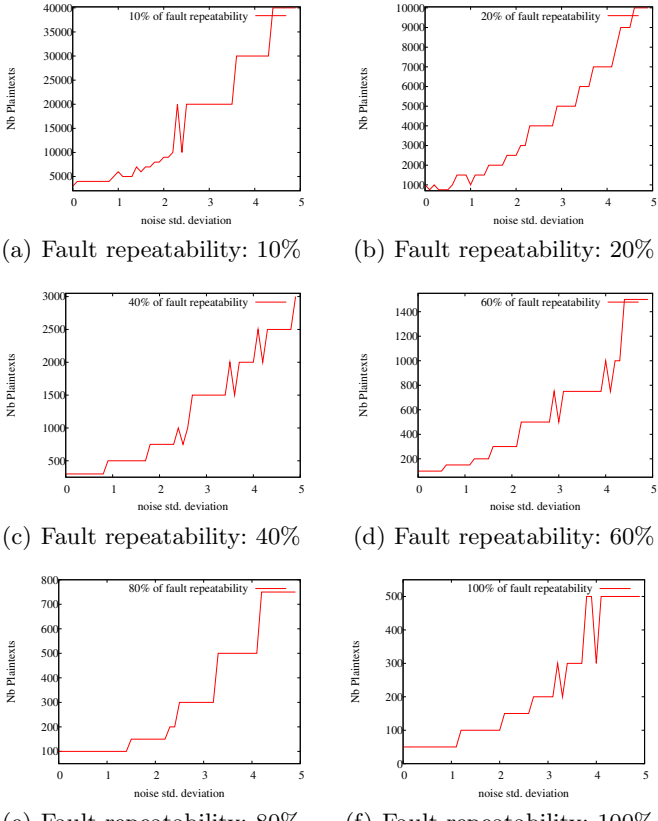
In this section we present the results of the attack simulations to show the rough evolution of complexity of the attack (in terms of number of plaintexts) as a function of fault injection repeatability and noise strength in order to reach a fixed

success rate of 90%. As in previous experiments, for each choice of parameters, we reached 90% of success over 100 attacks.

The simulations assumed a leakage function to be, for each ciphertext byte $\widetilde{C}_j^i$, of the form:

$$L(\widetilde{C}_j^i) = HW(\widetilde{C}_j^i) + \mathcal{N}(\mu, \sigma) \ ,$$

with $\mu = 0$ and σ going from 1 to 5. Figures 2(a) to 2(f) show the results, each of them for a fixed fault injection repeatability. The exponential growth of complexity as a function of the noise standard deviation is clearly visible on the figures, this is also the case with respect to the fault injection repeatability. Figures 2(c) to 2(f) capture practical scenarios where the complexity is limited (< 4000 plaintexts), σ up to 5 and fault repeatability down to 40%. Recall that in the first attack setup (with pre-computed key schedule), the repeatability would be 100%.



(a) Fault repeatability: 10% (b) Fault repeatability: 20%

(c) Fault repeatability: 40% (d) Fault repeatability: 60%

(e) Fault repeatability: 80% (f) Fault repeatability: 100%

Fig. 2. Number of valid-, faulty-ciphertext pairs for 90% success rate with respect to the gaussian noise standard deviation for each probability of fault injection repeatability (from 10% to 100%)

6 Countermeasures

Some countermeasures against classical SCA and FA attacks are still applicable to the combined attack presented in the previous Section. For instance, the insertion of random delays during the cryptographic computation. This countermeasure is not new, it is already used by industrials and several papers provide a study of its efficiency [15, 31]. Nevertheless, this countermeasure does not make the attack infeasible, it just makes it harder. Moreover there exists methods to bypass this countermeasure, which can be applied in our case (for instance [13]).

Another possible countermeasure could be to keep the ciphertext masked before applying the integrity check. For instance, in the case of the redundancy countermeasure, if C^1 and C^2 are the two ciphertexts obtained from the same encryption and M^1 and M^2 the two different masks used in each instance of the encryption, one gets $C^1 \oplus M^1, C^2 \oplus M^2, M^1$ and M^2. Then one has to XOR $C^1 \oplus M^1$ with M^2 and $C^2 \oplus M^2$ with M^1, and compare the two values. If they are equal, one can unmask the ciphertext without risk of vulnerability. A similar method could be used in the case of the DFA countermeasure consisting of encryption/decryption to check the validity of the ciphertext.

7 Conclusion

In this paper we have introduced a new DFA on AES (or any SP-Network). This attack is particularly interesting when the key schedule is unmasked or masked with a fixed mask. In such cases, we believe this FA attack to have the least restrictive fault model in terms of fault pattern and fault location compared to state-of-the-art attacks. A relaxation on the fault model means that the attacker needs less knowledge about the implementation or less precise fault injection material. The price we pay for this relaxation is the constraint of repeatability of the fault (not necessary in the case of a pre-computed key schedule), we think that in an overwhelming majority of cases, this hypothesis will be easily satisfied. However, when the key-schedule is masked with fresh random values at each execution, the types of faults that lead to a successful attack are severely restricted and the attack becomes harder but still possible in some context.

In a second part, the DFA is combined with a 1^{st}-order SCA to defeat block cipher implementations including not only a masking countermeasure but also an error detection mechanism on the cipher result. Simulations of the attacks verify that, even though the complexity (in term of plaintext number) grows exponentially with the probability of error in the fault repeatability, the attack is still practical for low repeatability ($\sim 40\%$) and realistic noise strength. Finally, we propose some simple countermeasures which can thwart our combined attack.

Future directions will be first to mount such combined attack on an off-the-shelf device and secondly to study the injection of *bit-flip* errors with good repeatability (which lead to the attack of fully masked block cipher implementations).

References

1. Data Encryption Standard, FIPS PUB 46-3 (1999)
2. Advanced Encryption Standard, FIPS PUB 197 (2001)
3. 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)
4. Amiel, F., Villegas, K., Feix, B., Marcel, L.: Passive and Active Combined Attacks: Combining Fault Attacks and Side Channel Analysis. In: Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC, pp. 92–102 (2007)
5. Selçuk, A.A., Biçak, A.: On Probability of Success in Linear and Differential Cryptanalysis. In: Cimato, S., Persiano, G., Galdi, C. (eds.) SCN 2002. LNCS, vol. 2576, pp. 174–185. Springer, Heidelberg (2003)
6. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The Sorcerer's Apprentice Guide to Fault Attacks. IACR Eprint archive (2004), `http://eprint.iacr.org/2004/100`
7. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
8. Blömer, J., Seifert, J.-P.: Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
9. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
10. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
11. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener (ed.) [33], pp. 398–412
12. Chen, C.-N., Yen, S.-M.: Differential Fault Analysis on AES Key Schedule and Some Coutnermeasures. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 118–129. Springer, Heidelberg (2003)
13. Clavier, C., Coron, J.-S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
14. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M.: Passive and Active Combined Attacks on AES: Combining Fault Attacks and Side Channel Analysis. In: Breveglieri, L., Joye, M., Koren, I., Naccache, D., Verbauwhede, I. (eds.) FDTC, pp. 10–19. IEEE Computer Society (2010)
15. Coron, J.-S., Kizhvatov, I.: An Efficient Method for Random Delay Generation in Embedded Software. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 156–170. Springer, Heidelberg (2009)
16. Dusart, P., Letourneux, G., Vivolo, O.: Differential Fault Analysis on A.E.S. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer, Heidelberg (2003)
17. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)

18. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 27–41. Springer, Heidelberg (2005)
19. Giraud, C., Thillard, A.: Piret and Quisquater's DFA on AES Revisited. IACR Eprint archive (2010), http://eprint.iacr.org/2010/440
20. 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)
21. Kim, C.H., Quisquater, J.-J.: New Differential Fault Analysis on AES Key Schedule: Two Faults Are Enough. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 48–60. Springer, Heidelberg (2008)
22. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener (ed.) [33], pp. 388–397
23. Messerges, T.S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
24. Mirbaha, A.P., Dutertre, J.-M., Ribotta, A.-L., Agoyan, M., Tria, A., Naccache, D.: Single-Bit DFA Using Multiple-Byte Laser Fault Injection. In: Technologies for Homeland Security (HST), pp. 113–119 (2010)
25. Mukhopadhyay, D.: An Improved Fault Based Attack of the Advanced Encryption Standard. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 421–434. Springer, Heidelberg (2009)
26. 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)
27. Piret, G., Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
28. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
29. Robisson, B., Manet, P.: Differential Behavioral Analysis. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 413–426. Springer, Heidelberg (2007)
30. Skorobogatov, S.P.: Optically Enhanced Position-Locked Power Analysis. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 61–75. Springer, Heidelberg (2006)
31. Tunstall, M., Benoit, O.: Efficient Use of Random Delays in Embedded Software. In: Sauveron, D., Markantonakis, K., Bilas, A., Quisquater, J.-J. (eds.) WISTP 2007. LNCS, vol. 4462, pp. 27–38. Springer, Heidelberg (2007)
32. Tunstall, M., Mukhopadhyay, D., Ali, S.: Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. In: Ardagna, C.A., Zhou, J. (eds.) WISTP 2011. LNCS, vol. 6633, pp. 224–233. Springer, Heidelberg (2011)
33. Wiener, M. (ed.): CRYPTO 1999. LNCS, vol. 1666. Springer, Heidelberg (1999)

A An FPGA Implementation Tampered with Clock Glitches

A.1 Architectural Details

In order to validate the assumption of a good repeatability for the fault injection (both for the key-schedule and the encryption state) a machine prototype was developed which allowed us to induce glitches into the clock signal of an AES hardware accelerator. We used an Altera Cyclone II FPGA[6] board involving an AES hardware engine and a main control unit, we will now describe the architecture thereof.

One should note that even though our design is rather specific, as it was tailored to the purpose of the feasibility study/demonstration, it clearly manages to demonstrate the proof of concept of a good repeatability. Indeed, no more assumptions were made on the implementation other than those described in the present paper and which are specifically required for the attack to succeed.

More precisely:

- the AES engine is a hardware unit performing the ten AES rounds in ten successive cycles in a combinational way, therefore using a sole 128-bit register to store the AES state;
- for each encryption run the key-schedule is computed separately from the encryption process, beforehand, allowing the fault to be injected at the same instant each time, and its effect on the last two subkeys to be appreciated according to what matters: its repeatability.
- a control unit is present in the design aside from the AES engine, which handles I/O serial communications with a remote PC and scheduling of computations;
- the AES engine and the control unit each occupies a private clock-domain; an on-chip PLL allows generating a separated clock for each, which share nonetheless both frequency (250 MHz) and phase alignment;
- the control unit drives a clock-enable signal that is fed to a global clock buffer placed ahead the clock network of the AES engine; this clock-enable signal is asserted one cycle out of eight, thus emulating an approx. 30 MHz clock to the encryption core.
- for each encryption run, the user can specify the plaintext and the master-key, along with the precise cycle in which he desires the AES engine clock to be cut short during the key-schedule computation. In the cycle specified by user, the control unit will assert, for two consecutive periods of its own clock, the clock-enable signal of the AES-engine, thereby emulating a transient clock of 250 MHz for an isolated period.

A.2 Attack Description

The above implementation was used to put the attack method described in Section 4.2 to the test:

[6] This is a 130 nm low-cost family.

- we randomly chose 100 plaintexts and performed their encryption without any fault injection;
- the same plaintexts were then encrypted again, this time with a clock-glitch at the next to last cycle of the key-schedule;
- this provided us with 100 pairs of valid-faulty ciphertexts;
- we ran the attack on an increasing number of such pairs. Eventually, only 15 pairs were sufficient to retrieve 11 round key bytes. The other 5 round key bytes were not affected by the fault injection; they could be obtained by exhaustive search.

Note that, according to Figure 1, one can deduce from this result that the probability of fault repeatability is of 50% in this experiment.