From AES-128 to AES-192 and AES-256, How to Adapt Differential Fault Analysis Attacks

Noémie Floissac and Yann L'Hyver

SERMA TECHNOLOGIES ITSEF
30, avenue Gustave Eiffel, 33608 Pessac, France
Email: {*n.floissac;y.lhyver*}*@serma.com*

Abstract

Since its announcement, AES has been subject to different DFA attacks. Most of these attacks target the AES with 128-bit key. However, the two other variants are nowadays deployed in various applications and are also submitted to the same attack path. In this paper, we adapt the DFA techniques originally used on AES-128 in order to obtain the keys of AES-192 and AES-256. To illustrate this method, we propose efficient attacks on AES-192 and AES-256 based on a known DFA on KeyExpansion.

Keywords

DFA, fault injection, AES-192, AES-256, adaptation, extension, reproduction

I. Introduction

Initially introduced by D. Boneh and al. in [1], **D**ifferential **F**ault **A**nalysis is an efficient attack path allowing to discover a secret key handled by an embedded cryptographic algorithm. This attack consists in corrupting intermediate states in order to produce faulty ciphers. The faults are obtained by physical perturbations of the targeted device. The DFA attack is based on the analysis of the differences between the faulty ciphers and the expected ones to obtain information on the secret key.

Attacks on **A**dvanced **E**ncryption **S**tandard can be split into two categories depending of the fault location: the DFA on the State and the DFA on the KeyExpansion. In both cases, the AES-128 variant has been widely covered with the DFA proposed in [2], [3], [4], [5], [6], [7] and [8] for the first category, and [4], [9], [10], [11] and [12] for the second one.

Recent papers focus on DFA on the state for AES-192 and AES-256, see [13] and [14]. However, from the best of our knowledge, no DFA attack on KeyExpansion has been explicitly published on these two variants.

The aim of this paper is to propose a general method to perform a DFA on the AES-192 and AES-256 by exploitation of the same techniques used on the AES-128. In several cases and after some changes, the same DFA attack can be applied for all key sizes. We distinguish hereafter the DFAs with a fault injected on the state and those with a fault injected on the KeyExpansion algorithm. In the first case, we formalize a generic adaptation of DFA attacks on the state. The second case is more complicated to treat due to the differences in the KeyExpansion algorithm according to the key size. A specific analysis has to be considered for each variant. We propose a DFA adaptation based on the attack of C. H. Kim and J.-J. Quisquater (see [10]).

This paper is organized as follows: the notations and vocabulary are defined in section II. In section III, we present some problems encountered to adapt DFA attacks from AES-128 and the different techniques to solve them. To illustrate this section, we propose an adaptation of attack [10] in section IV. Finally, we present some results concerning these attack adaptations and we give our conclusion in section V.

II. Notations, vocabulary and background

The AES is a symmetric block cipher standard based on iterations of four transformations (AddRoundKey, SubBytes, ShiftRows and MixColumns). The number of iterations also called rounds depends on the selected variant and so on the key length. Each transformation is performed once by round except for the last round where the MixColumns tranformation is not included.

In the remainder of the paper, we use the following notations and vocabulary:

- $\mathbf{N_R}$: Number of rounds. Its value depends on the size of initial key, see Table I.
- $\mathbf{N_k}$: Number of 32-bit words used during the key schedule process. Its value depends on the size of the initial key, see the following table:

Variant	N_R	N_k
AES-128	10	4
AES-192	12	6
AES-256	14	8

TABLE I

- **t**: one of the following transformations:
  - **SB**: SubBytes
  - **SR**: ShiftRows
  - **MC**: MixColumns
  - **I_MC**: inverse MixColumns
  - **I_SR**: inverse ShiftRows
  - **I_SB**: inverse SubBytes
  - **ARK**: AddRoundKey
  - **SW**: SubWord
  - **RW**: RotWord
- $\mathbf{S_{t,r}}$: State issued from the transformation **t** on round **r**
- $\mathbf{S_{t,r}\{i,j\}}$: Byte {i,j} issued from state $\mathbf{S_{t,r}}$. A state is represented by an array of four lines and four columns (32-bit words) where bytes are disposed as in Table II.
- $\mathbf{S^*_{t,r}}$: Faulty state issued from the transformation **t** on round **r**

$\mathrm{S_{t,r}}\{0,0\}$	$\mathrm{S_{t,r}}\{0,1\}$	$\mathrm{S_{t,r}}\{0,2\}$	$\mathrm{S_{t,r}}\{0,3\}$
$\mathrm{S_{t,r}}\{1,0\}$	$\mathrm{S_{t,r}}\{1,1\}$	$\mathrm{S_{t,r}}\{1,2\}$	$\mathrm{S_{t,r}}\{1,3\}$
$\mathrm{S_{t,r}}\{2,0\}$	$\mathrm{S_{t,r}}\{2,1\}$	$\mathrm{S_{t,r}}\{2,2\}$	$\mathrm{S_{t,r}}\{2,3\}$
$\mathrm{S_{t,r}}\{3,0\}$	$\mathrm{S_{t,r}}\{3,1\}$	$\mathrm{S_{t,r}}\{3,2\}$	$\mathrm{S_{t,r}}\{3,3\}$

TABLE II

- $\mathbf{K_r}$: Key of round **r**
- $\mathbf{K_r\{i,j\}}$: The byte {i,j} issued from key $\mathbf{K_r}$. A round key is represented by an array of four lines and four columns where bytes are disposed as a state.
- $\mathbf{K^*_r}$: Faulty key of round **r**
- **Sb**: Substitution table used by SubBytes and SubWord transformations
- **I_Sb**: Inverse substitution table used by inverse SubBytes transformations
- **RCon**: 32-bit constant word such as:
  $\mathbf{RCon[i] = (2^{i-1}, 0, 0, 0)}$
- **xor** ($\oplus$): Exclusive-OR operation
- **Couple**: A set composed with the correct and faulty ciphers
- **Differential**: Exclusive-OR between two states or round keys
- **X [Y]** or **X mod Y**: Modular reduction of X by Y

The AES cipher algorithm uses the round keys obtained from the KeyExpansion algorithm.

The key schedule consists in diversifying a previous round key in order to obtain a new round key. This transformation can be performed before AES cipher algorithm, see the pseudo-code given in **Algorithm 1**, or *on the fly* during the AES cipher computation.

Algorithm 1 *KeyExpansion pseudo-code*

Input: K, *the initial key of length* N_k

Output: K_i *with* $i = 0, ..., N_R$, *the round keys*

for $i = 0$ **to** $N_k - 1$ **by** 1 **do**
  $W[i] \leftarrow [K\{0,i\}, K\{1,i\}, K\{2,i\}, K\{3,i\}]$
end for
for $i = N_k$ **to** $((4 * (N_R + 1)) - 1)$ **by** 1 **do**
  $T \leftarrow W[i-1]$
  if $(i \ mod \ N_k) == 0$ **then**
    $T \leftarrow SW(RW(T)) \oplus RCon[\frac{i}{Nk}]$
  else if $(N_k > 6)$ **and** $(i \ mod \ N_k) == 4$ **then**
    $T \leftarrow SW(T)$
  end if
  $W[i] \leftarrow W[i - N_k] \oplus T$
end for
for $i = 0$ **to** N_R **by** 1 **do**
  $K_i \leftarrow [W[i*4], W[i*4+1], W[i*4+2], W[i*4+3]]$
end for

The interested reader can obtain more information on the AES in document [15].

III. Adaptations of DFA to AES-192 and AES-256

Commonly, the AES-128 is faulted on the last rounds to retrieve the complete last round key or a subset of it. With this knowledge, the initial 128-bit key is calculated thanks to the computation of the inverse KeyExpansion algorithm.

We define by *adaptation* the generalization of such DFA attack to AES-192 or AES-256. The *adaptation* can be decomposed into two phases described below: an *extension* and a *reproduction*.

A. Extension and reproduction

As all the AES variants are based on the same structure, a first idea consists in applying known DFA attacks on AES-128 to the two other variants with the aim to retrieve the last round key $\mathbf{K_{N_R}}$. The identified strategy is to inject fault(s) on rounds having the same position from the end of AES as those targeted by the DFA attack on AES-128 and to exploit the differential faults with the same techniques. We call this phase an *extension*.

For AES-192 and AES-256, the last round key is not sufficient to obtain the initial key: it is required to know respectively the last 8 bytes and 16 bytes of penultimate round key $\mathbf{K_{N_R-1}}$. The *extension* of the DFA attack is not enough to determine all the missing bytes.

The second phase of the *adaptation*, called *reproduction*, consists in reiterating the DFA attack in order to retrieve the penultimate key $\mathbf{K_{N_R-1}}$. To do that, a method is to exploit the differential fault at the end of the penultimate round. The goal is to reduce the AES algorithm by neutralizing its last round. Hence, the DFA attack could be applied on a shorter AES cipher algorithm, the penultimate round becoming the last one.

The AES reduction is performed by reversing a cipher back up to the output of the round $\mathbf{N_R - 1}$ with the knowledge of the last round key. In order for the penultimate round to be equivalent to the last round, the MixColumns transformation must be missing. However, this transformation operates on the state S_{SR,N_R-1} and, at the end of the round $\mathbf{N_R - 1}$, we have the equation below:

$$S_{ARK,N_R-1} = MC(S_{SR,N_R-1}) \oplus K_{N_R-1} \quad (1)$$

For cancelling the effect of MixColumns transformation, we apply its inverse on the state S_{ARK,N_R-1} as presented in paper [14]. Due to the linear property of this transformation, we obtain a cipher

$\mathbf{C'}$ as follows:

$$C' = I\_MC(S_{ARK,N_R-1}) = S_{SR,N_R-1} \oplus I\_MC(K_{N_R-1}) \tag{2}$$

The key used by the new AddRoundKey transformation becomes $\mathbf{I\_MC(K_{N_R-1})}$ instead of $\mathbf{K_{N_R-1}}$. Thus, the *reproduction phase* on the new couples actually returns the inverse MixColumns transformation applied to the key $\mathbf{K_{N_R-1}}$.

In conclusion, the *adaptation* consists in applying twice the original attack: once during the *extension phase* (attack on the last round), then during the *reproduction phase* (attack on the penultimate round). In the remainder of this paper, we discuss whether this general method can be applied more or less easily to different kinds of published DFA attacks.

B. Adaptation of DFA on state

For each key length, the AES cipher algorithms have the same sucessive transformations on the last rounds, only the number of rounds changes. Thereby, the fault diffusion will be the same for each AES variant and the *adaptation* of a known DFA attack is obvious to perform.

This technique has been previously used by A. Barenghi and al. [14] to adapt the attack of G. Piret, and J-J. Quisquater [2] on AES-192 and AES-256 variants. It can be generalized to the other published DFAs targeting the AES state.

In terms of time and number of couples, the cost of the *adaptation* of DFA attack to AES-192 and AES-256, including both *extension* and *reproduction phases*, is twice that needed to perform the DFA attack on the AES-128 algorithm returning the last round key.

For example, the DFA [5], an attack with faults injected on round 8, allows finding the round key $\mathbf{K_{10}}$ of AES-128. The same kind of fault respectivelly applied on round 10 and 12 of AES-192 and AES-256 during *extension phase* returns the last round key $\mathbf{K_{12}}$ and $\mathbf{K_{14}}$. Then the fault injection respectivelly targets the round 9 and 11 of AES-192 and AES-256 and the *reproduction* of attack reveals the round keys $\mathbf{K_{11}}$ and $\mathbf{K_{13}}$. The *adaptation* allows retrieving the whole AES key with only 4 couples in the most efficient case.

In some particular cases, the DFA attack leads to obtaining a subset of solution for the last round key, for example the attack proposed in paper [7]. Thereby, the *reproduction phase* of the DFA attack must take into account each hypothesis on the last round key. The number of faults does not increase but the time required to exploit the differential faults is the square of the time taken to perform the DFA attack on AES-128 variant. Indeed, the *reproduction phase* is based on the knowledge of the last round key and, therefore, the DFA performed in order to reveal the penultimate round key $\mathbf{K_{N_R-1}}$ should be repeated for each element of the subset of solutions. The retrieval of the initial key requires to compute the inverse KeyExpansion algorithm for each element of the subset containing the penultimate round key.

C. Adaptation of DFA on KeyExpansion

In the case of the KeyExpansion, the algorithm differs for each variant. The main differences between AES-192 and AES-128 KeyExpansion algorithms are the following:

- The RotWord and SubWord transformations are not applied on the last column of round key $\mathbf{K_{N_R-2}}$.
- The two first columns of the last round key depend on the two last columns of round key $\mathbf{K_{N_R-2}}$.
- The two last columns of the round key $\mathbf{K_{N_R-1}}$ do not impact the two last columns of round key $\mathbf{K_{N_R}}$.

Concerning the AES-256 variant, the differences with AES-128 are:

- Only the SubWord transformation is applied on the last column of round key $\mathbf{K_{N_R-2}}$.
- All the columns of the last round key depend on the four columns of round key $\mathbf{K_{N_R-2}}$.
- The columns of the round key $\mathbf{K_{N_R-1}}$ do not impact the columns of round key $\mathbf{K_{N_R}}$ except the last one, on which the RotWord and SubWord transformations are applied.

Due to these differences, the fault diffusion will not be the same for each variant. We distinguish three main problems that have to be solved to lead the *adaptation* successfully.

First problem

The aim of the *extension phase* is to reveal the last round key $\mathbf{K_{N_R}}$ by applying the methodology used on AES-128. However, the fault diffusion does not follow the same paths for each AES variant due to the differences in KeyExpansion algorithms. In order to achieve the *extension phase*, the faults for the three variants must be sufficiently similar.

In some cases, the faults propagation differs too much to solve this problem and the *adaptation* cannot be performed. For example, applying [11] on AES-128 leads to retrieve one by one the bytes of the last round key, all the bytes being linked. The attack is similar to a resolution of chained equations. In the case of AES-192 and AES-256, the faults layout implies that the bytes searched do not correspond to the ones of AES-128 any more. Thus, the chained equations cannot be solved and the *extension phase* cannot be performed.

In favourable cases, the problem could be solved with few changes from the original attack without modifying its main strategy.

Second problem

The objective of the *reproduction phase* is to reduce the AES algorithm to cancel its last round. The first step of this phase consists in operating the inverse transformations on the cipher until the end of round $\mathbf{N_R - 1}$. With the help of the last round key, the inverse transformations are directly performed on the correct cipher.

Concerning the faulty result, the constraint comes from the diffusion of the injected fault. The effect of diffused faults on the last round key has to be cancelled during the inverse transformation process. Thus, the second problem consists in obtaining the faulty round key $\mathbf{K^*_{N_R}}$.

In some cases, the propagated faults on the last round key are hidden by those obtained on the internal states of the AES cipher algorithm. Indeed, during the last AddRoundKey transformation, the faulty result can include a mix of faults on the key and on the state. In the worst case, the faults occur on the same bytes and the xor operation hides them in the output cipher.

This problem could sometimes be solved because the concealed fault of the last round key can be expressed from a differential analysis of internal state or be deduced from other corrupted bytes of the faulty cipher.

When the round key $\mathbf{K^*_{N_R}}$ is determined, all the inverse transformations can be applied on the faulty computation back up to the expected transformation output.

Third problem

The final step of *reproduction phase* consists in applying the trick of the inverse MixColumns transformation on $\mathbf{S_{ARK,N_R-1}}$ and $\mathbf{S^*_{ARK,N_R-1}}$ in order to reproduce the attack on the reduced form of AES algorithm. The keys used by the new AddRoundKey transformation become $\mathbf{K'} = \mathbf{I\_MC}(\mathbf{K_{N_R-1}})$ and $\mathbf{K'^*} = \mathbf{I\_MC}(\mathbf{K^*_{N_R-1}})$.

The third problem is linked to the inverse MixColumns transformation properties:

- The linearity implies that the key $\mathbf{K'^*}$ is faulted by $\mathbf{I\_MC}(\mathbf{K_{N_R-1}} \oplus \mathbf{K^*_{N_R-1}})$.
- The diffusion properties increase the number of faulted bytes in $\mathbf{K'^*}$ in comparison with $\mathbf{K^*_{N_R-1}}$.
- At the inverse MixColumns output, a byte depends on the four bytes in the same column of the input. Thus the faults in $\mathbf{K'^*}$ are linked in a more complex manner than in $\mathbf{K^*_{N_R-1}}$.

Depending on cases, these properties could or not defeat the *reproduction phase*.

Whenever this problem is solved, the DFA technique previously used during the *extension phase* is applied to the reduced AES to reveal the round key $\mathbf{K_{N_R-1}}$.

In conclusion, the attack *adaptation* on KeyExpansion algorithm from AES-128 to AES-192 and AES-256 is more complex than the *adaptation* of DFA on state. All problems must be successively solved to obtain respectively the 24 and 32 bytes of the last round keys and to finally compute the initial key.

IV. Example : Adaptation of C .H. Kim J.-J. Quisquater DFA attack

In this section, we detail the *adaptation* of the DFA attack on KeyExpansion algorithm proposed by C. H. Kim and J.-J. Quisquater in [10]. This is the most recent paper targeting the key diversification and using only four couples to reveal the last round key in the most efficient version.

The attack *adaptation* is subject to the three mentioned problems. We answer to the first one by the *extension phase* on both AES-192 and AES-256. The analysis of the fault propagation has led to study the adaptation on AES-256 first and then the adaptation on AES-192. As we will see, the AES-256 adaptation has to overcome the second problem whereas the AES-192 adaptation is subject to the third problem. To illustrate each problem, we succesively treat the cases of AES-256 and AES-192.

A. The original attack

The basic attack exploits a fault injection corrupting one byte during the computation of the 9^{th} round key. The full attack requires eight couples to retrieve the 16 bytes of the last round key. Whenever the fault injection impacts several bytes, the number of required couples is reduced to four. For simplicity reasons, only the *adaptation* of the one-byte perturbation is described because fault repercussion is easier to treat.

The fault injection targets the first column of the round key $\mathbf{K_{N_R-1}}$. We denote by $\mathbf{a}$ the random fault value and by $\mathbf{i}$ the line index where the fault is injected. Hence, the faulty round key can be written:

$$\mathbf{K^*_{N_R-1}\{i,0\} = K_{N_R-1}\{i,0\} \oplus a} \tag{3}$$

During the KeyExpansion computation, each word of key comes from a linear transformation depending on a part of the previous word. Thereby, the fault is propagated on each column of the key $\mathbf{K_{N_R-1}}$ such that all bytes of line $\mathbf{i}$ are impacted by the same fault value $\mathbf{a}$. The equation (3) applies to all columns $\mathbf{j}$ as below:

$$\mathbf{K^*_{N_R-1}\{i,j\} = K_{N_R-1}\{i,j\} \oplus a}, \tag{4}$$

$$\text{where j} \in [0..3]$$

The faults present on the round key $\mathbf{K^*_{N_R-1}}$ contaminate the state following the AddRoundKey transformation of round $\mathbf{N_R-1}$. As the AddRoundKey transformation is linear (xor operation between round key and state bytes), the fault value $\mathbf{a}$ is transferred onto each byte of line $\mathbf{i}$ of state $\mathbf{S_{ARK,N_R-1}}$. We obtain:

$$\mathbf{S^*_{ARK,N_R-1}\{i,j\} = S_{ARK,N_R-1}\{i,j\} \oplus a}, \tag{5}$$

$$\text{where j} \in [0..3]$$

The non-linear SubBytes transformation is applied to $\mathbf{S^*_{ARK,N_R-1}}$. This transforms the fault value $\mathbf{a}$ to a new value that is unpredictable without the prior knowledge of $\mathbf{S^*_{ARK,N_R-1}}$.

Finally, the last AddRoundKey transformation induces a new fault value $\mathbf{b}$ on all the bytes of line $\mathbf{(i-1)\ mod\ 4}$ due to SubWord and RotWord transformations. The fault value $\mathbf{b}$ can be expressed from $\mathbf{a}$ as follows:

$$\mathbf{b = Sb(K_{N_R-1}\{i,3\} \oplus a) \oplus Sb(K_{N_R-1}\{i,3\})} \tag{6}$$

The result $\mathbf{C^*}$ of the faulted computation corresponds to the expected result $\mathbf{C}$ with exactly two lines entirely faulted.

The DFA attack on KeyExpansion exploits several couples ($\mathbf{C}$, $\mathbf{C^*}$) to reveal the last round key $\mathbf{K_{N_R}}$. All intermediate states can be expressed from the cipher. It is the same for the states $\mathbf{S_{ARK,N_R-1}}$ and $\mathbf{S^*_{ARK,N_R-1}}$ that can be respectively written from the obtained ciphers $\mathbf{C}$ and $\mathbf{C^*}$ as follows:

$$\mathbf{S_{ARK,N_R-1}} = \mathbf{LSB}(\mathbf{LSR}(\mathbf{C} \oplus \mathbf{K_{N_R}})) \tag{7}$$

$$\mathbf{S^*_{ARK,N_R-1}} = \mathbf{LSB}(\mathbf{LSR}(\mathbf{C^*} \oplus \mathbf{K^*_{N_R}})) \tag{8}$$

For the bytes on line $\mathbf{i}$, the previous equations become:

$$\mathbf{S_{ARK,N_R-1}}\{\mathbf{i},\mathbf{j}\} = \mathbf{LSb}(\mathbf{C}\{\mathbf{i},(\mathbf{j}-\mathbf{i})[\mathbf{4}]\} \oplus \mathbf{K_{N_R}}\{\mathbf{i},(\mathbf{j}-\mathbf{i})\ [\mathbf{4}]\}) \tag{9}$$

$$\mathbf{S^*_{ARK,N_R-1}}\{\mathbf{i},\mathbf{j}\} = \mathbf{LSb}(\mathbf{C^*}\{\mathbf{i},(\mathbf{j}-\mathbf{i})[\mathbf{4}]\} \oplus \mathbf{K^*_{N_R}}\{\mathbf{i},(\mathbf{j}-\mathbf{i})\ [\mathbf{4}]\}) \tag{10}$$

The equations (5), (9) and (10) give a new equation where unknown values are $\mathbf{a}$ and $\mathbf{K_{N_R}}\{\mathbf{i},\mathbf{j}\}$:

$$\mathbf{a} = \mathbf{LSb}(\mathbf{C}\{\mathbf{i},\mathbf{j}-\mathbf{i}[\mathbf{4}]\} \oplus \mathbf{K_{N_R}}\{\mathbf{i},\mathbf{j}-\mathbf{i}[\mathbf{4}]\}) \oplus \mathbf{LSb}(\mathbf{C^*}\{\mathbf{i},\mathbf{j}-\mathbf{i}[\mathbf{4}]\} \oplus \mathbf{K^*_{N_R}}\{\mathbf{i},\mathbf{j}-\mathbf{i}[\mathbf{4}]\}), \quad \text{where } \mathrm{j} \in [0;3] \tag{11}$$

The equation (11) can be simplified, as:

- $\mathbf{K^*_{N_R}}\{\mathbf{i},\mathbf{j}\} = \mathbf{K_{N_R}}\{\mathbf{i},\mathbf{j}\}$, whenever the column $\mathbf{j}$ equals to 1 or 3;
- $\mathbf{K^*_{N_R}}\{\mathbf{i},\mathbf{j}\} = \mathbf{K_{N_R}}\{\mathbf{i},\mathbf{j}\} \oplus \mathbf{a}$ in both other cases.

Thus, to solve the equation, it is necessary to know the fault value $\mathbf{a}$ when $\mathbf{j}$ is equal to 0 or 2. The equation is solved by an exhaustive search on each $\mathbf{K_{N_R}}\{\mathbf{i},\mathbf{j}\}$ value, beginning with the cases where $\mathbf{j}$ is equal to 1 or 3.

At this step, the attack returns a subset of possible quadruplets for the four bytes of the last round key. To retrieve the expected quadruplet of key bytes, two couples ($\mathbf{C}$, $\mathbf{C^*}$) and ($\mathbf{D}$, $\mathbf{D^*}$) coming from a random fault on the same localization are sufficient. Indeed, only the correct values of key bytes verify the equations obtained from different couples.

The fault injection is reiterated for the three other lines $\mathbf{i}$ of the first column to reveal the whole key $\mathbf{K_{N_R}}$ with eight couples.

B. First problem

The first step of the adaptation is the *extension phase*. Concerning the C. H. Kim and J.-J. Quisquater's attack, this is trivially processed. Small differences on the fault diffusion appear whenever the original attack is applied on both AES-192 and AES-256 variants. All the faults are identically propagated like for AES-128 algorithm except that:

- In the case of AES-192, the byte $\mathbf{K^*_{N_R}}\{\mathbf{i},\mathbf{0}\}$ is not faulted,
- In the case of AES-256, the bytes $\mathbf{K^*_{N_R}}\{\mathbf{i},\mathbf{0}\}$ and $\mathbf{K^*_{N_R}}\{\mathbf{i},\mathbf{2}\}$ are not faulted.

The equation (11) needs to be adapted to take into account the fault value $\mathbf{a}$ on the identified bytes of $\mathbf{K^*_{N_R}}$. Nevertheless, it does not have a large impact because the C. H. Kim and J.-J. Quisquater attack does not exploit the faults in the last round key. The **Algorithm 2** details the different steps of this *extension phase*.

Algorithm 2 *Extension phase on AES-192 and AES-256*

Input: i, the index of the line corrupted by the fault on $\mathbf{K_{N_R-1}}$, 2 couples ($\mathbf{C}$, $\mathbf{C^*}$) and ($\mathbf{D}$, $\mathbf{D^*}$) with fault injected on line i

Output: $\mathbf{K_{N_R}\{i,j\}}$, with j = 0,..,3

Let k_0,k_1,k_2 and k_3 be the respective values of $K_{N_R}\{i,0\}$,$K_{N_R}\{i,1\}$,$K_{N_R}\{i,2\}$ and $K_{N_R}\{i,3\}$

```
for k1 = 0 to 255 by 1 do
  for k3 = 0 to 255 by 1 do
    e1 = I_Sb(C{i,1} ⊕ k1) ⊕ I_Sb(C*{i,1} ⊕ k1)
    e3 = I_Sb(C{i,3} ⊕ k3) ⊕ I_Sb(C*{i,3} ⊕ k3)
    if e1 == e3 then
      for k0 = 0 to 255 by 1 do
        e0 = I_Sb(C{i,0} ⊕ k0) ⊕ I_Sb(C*{i,0} ⊕ k0)
        if e0 == e1 then
          for k2 = 0 to 255 by 1 do
            if AES-256 then
              e2 = I_Sb(C{i,2} ⊕ k2) ⊕ I_Sb(C*{i,2} ⊕ k2)
            else
              e2 = I_Sb(C{i,2} ⊕ k2) ⊕ I_Sb(C*{i,2} ⊕ k2 ⊕ e1)
            end if
            if e2 == e1 then
              f1 = I_Sb(D{i,1} ⊕ k1) ⊕ I_Sb(D*{i,1} ⊕ k1)
              f3 = I_Sb(D{i,3} ⊕ k3) ⊕ I_Sb(D*{i,3} ⊕ k3)
              f0 = I_Sb(D{i,0} ⊕ k0) ⊕ I_Sb(D*{i,0} ⊕ k0)
              if AES-256 then
                f2 = I_Sb(D{i,2} ⊕ k2) ⊕ I_Sb(D*{i,2} ⊕ k2)
              else
                f2 = I_Sb(D{i,2} ⊕ k2) ⊕ I_Sb(D*{i,2} ⊕ k2 ⊕ f1)
              end if
              if f0 == f1 == f2 == f3 then
                return [k0, k1, k2, k3]
              end if
            end if
          end for
        end if
      end for
    end if
  end for
end for
```

With this algorithm and faults injected twice on each line $\mathbf{i}$, the round key $\mathbf{K_{N_R}}$ is fully discovered.

Furthermore, the four bytes of last column of $\mathbf{K_{N_R-1}}$ can be deduced from the resolution of equation (6). The fault values $\mathbf{a}$ and $\mathbf{b}$ are known and so each byte value of $\mathbf{K_{N_R-1}\{i,3\}}$ is retrieved by an exhaustive search.

Thus, the *extension phase* of the DFA attack allows finding the whole key $\mathbf{K_{N_R}}$ and the four last bytes of the key $\mathbf{K_{N_R-1}}$.

C. Second problem: case of AES-256

At this step, a part of the final key is henceforth known. Twelve more bytes need to be revealed with the *reproduction* phase.

Suppose that a random fault value denoted $\mathbf{a}$ occurs on line $\mathbf{i}$ on the first column of $\mathbf{K_{N_R-2}}$. We illustrate in figure 1 the propagation of a fault injection on line 1.

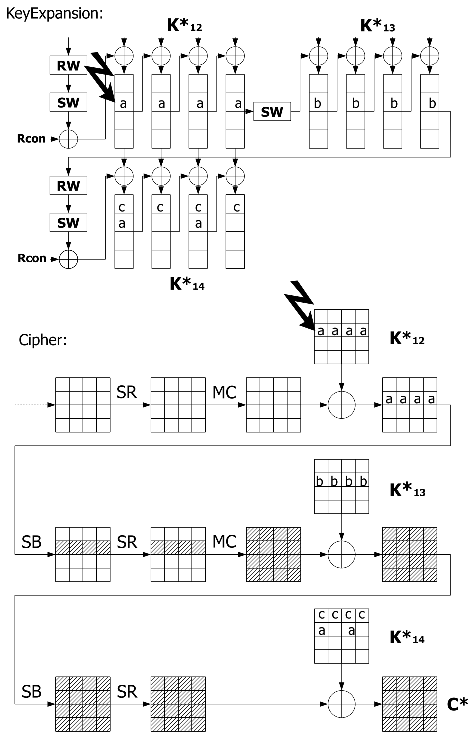

Fig. 1. *Reproduction phase* on AES-256

The computation of the three following columns results in the diffusion of this fault on the entire line. We obtain the equation below:

$$\mathbf{K^*_{N_R-2}\{i,j\} = K_{N_R-2}\{i,j\} \oplus a}, \qquad (12)$$
$$\text{where } \mathrm{j} \in [0..3]$$

As the round key $\mathbf{K_{N_R-1}}$ is computed from the previous key $\mathbf{K_{N_R-2}}$, the injected fault also corrupts the key of round $\mathbf{N_R-1}$. The obtained faulty key $\mathbf{K^*_{N_R-1}}$ presents the following characteristics: its first column is the result of the xor operation between columns 3 and 0 of round keys $\mathbf{K^*_{N_R-2}}$ and $\mathbf{K_{N_R-3}}$ respectivelly.

The corresponding equation can be written as below:

$$\mathbf{K^*_{N_R-1}\{i,0\} = Sb(K_{N_R-2}\{i,3\} \oplus a) \oplus K_{N_R-3}\{i,0\}} \qquad (13)$$

Let $\mathbf{b}$ be the fault value issued from the SubWord transformation on $\mathbf{K^*_{N_R-2}}$. The relation between the two faults can be expressed as follows:

$$\mathbf{b = Sb(K_{N_R-2}\{i,3\} \oplus a) \oplus Sb(K_{N_R-2}\{i,3\})} \qquad (14)$$

The whole line $\mathbf{i}$ of $\mathbf{K^*_{N_R-1}}$ is corrupted by $\mathbf{b}$.

Finally, the key $\mathbf{K^*_{N_R}}$ is computed from $\mathbf{K^*_{N_R-2}}$ and $\mathbf{K^*_{N_R-1}}$. The KeyExpansion algorithm of AES-256 implies that the last column of $\mathbf{K^*_{N_R-1}}$ is affected by the SubWord and the RotWord transformations. Thus, the fault on line $\mathbf{i}$ induces a new fault value on line $\mathbf{(i-1) \bmod 4}$. We denote by $\mathbf{c}$ the fault issued from $\mathbf{b}$ before the SubWord transformation and giving the following equation:

$$\mathbf{c = Sb(K_{N_R-1}\{i,3\} \oplus b) \oplus Sb(K_{N_R-1}\{i,3\})} \qquad (15)$$

In addition to fault $\mathbf{c}$, the round key $\mathbf{K^*_{N_R}}$ is impacted by the result of xor operation between fault $\mathbf{a}$ and $\mathbf{K^*_{N_R-2}}$ on all the bytes of line $\mathbf{i}$. As each column of the round key is the result of a xor operation depending on the previous column, fault $\mathbf{a}$ is present in columns 0 and 2 and is absent from columns 1 and 3.

Fault $\mathbf{a}$ is introduced in the state following the AddRoundKey transformation of round $\mathbf{N_R-2}$. The following SubBytes transformation changes the fault value $\mathbf{a}$ to four new values that are unpredictable without the prior knowledge of $\mathbf{S_{ARK,N_R-2}}$. They correspond to the shaded cases of figure 1.

Next, all the bytes of the state issued from the MixColumns transformation are modified by the diffusion of the previous faults.

The faults continue their propagation until the cipher output. The resulting cipher $\mathbf{C}^*$ is integrally faulted.

In order to perform the *reproduction phase*, and particularly to reduce the AES, the knowledge of the round key $\mathbf{K}^*_{\mathbf{N_R}}$ is required. However, the faults occurring in $\mathbf{K}^*_{\mathbf{N_R}}$ cannot be directly determined from the faulty cipher. They are xored during the last AddRoundKey transformation with the unpredictable faults issued from the last ShiftRows transformation. Thus, the values $\mathbf{c}$ and $\mathbf{a}$ cannot be directly extracted from the cipher and we are confronted to the second problem.

The trick

The $\mathbf{K}^*_{\mathbf{N_R}}$ key is faulted by $\mathbf{c}$ on the entire line $(\mathbf{i-1})$ **mod 4** and by $\mathbf{a}$ on columns 0 and 2 of line $\mathbf{i}$. It means that the second problem is reduced to finding the values $\mathbf{c}$ and $\mathbf{a}$ and the index $\mathbf{i}$ of the impacted line.

Fortunately, the faults obtained on the KeyExpansion algorithm are linked together due to reiteration of a linear transformation. Thus, fault $\mathbf{c}$ can be expressed from $\mathbf{a}$ with the help of equations (14) and (15) as follows:

$$\mathbf{c} = \mathbf{Sb}(\mathbf{K_{N_R-1}}\{\mathbf{i},\mathbf{3}\} \oplus \mathbf{Sb}(\mathbf{K_{N_R-2}}\{\mathbf{i},\mathbf{3}\} \oplus \mathbf{a}) \oplus \mathbf{Sb}(\mathbf{K_{N_R-2}}\{\mathbf{i},\mathbf{3}\})) \oplus \mathbf{Sb}(\mathbf{K_{N_R-1}}\{\mathbf{i},\mathbf{3}\}) \quad (16)$$

Moreover, column 3 of $\mathbf{K_{N_R-2}}$ is obtained by xor operation between columns 2 and 3 of $\mathbf{K_{N_R}}$ and column 3 of $\mathbf{K_{N_R-1}}$ previously found during the *extension phase*. In conclusion, only values $\mathbf{a}$ and $\mathbf{i}$ need to be found to solve the second problem. This can be performed by a quick exhaustive search because only 255 values of $\mathbf{a}$ and four values of $\mathbf{i}$ have to be guessed.

In order to validate or not a hypothesis, we consider the following assumption:

The entire line $\mathbf{i}$ *on the state following the AddRoundKey transformation of round 12 is faulted by* $\mathbf{a}$.

Only a correct hypothesis verifies this assumption.

The assumption on the entire line $\mathbf{i}$ cannot be checked because the couple $(\mathbf{C}, \mathbf{C}^*)$ cannot be deciphered until the state $\mathbf{S_{ARK,N_R-2}}$. Only the following data are known from the inverse transformation and the knowledge of the last column of $\mathbf{K_{N_R-1}}$:

- the whole state $\mathbf{S_{MC,N_R-1}}$,
- the last column of inverse MixColumns transformation applied on $\mathbf{S_{MC,N_R-1}}$,
- the bytes {3,0}, {2,1}, {1,2} and {0,3} after the inverse ShiftRows and inverse SubBytes transformations on $\mathbf{S_{MC,N_R-1}}$.

The assumption is reduced to the four known values of $\mathbf{S_{SR,N_R-1}}$:

The byte on line $\mathbf{i}$ *is faulted by* $\mathbf{a}$ *and the three others are not faulted.*

The **Algorithm 3** details this trick.

Algorithm 3 *Attack adaptation on AES-256*

Input: ($\mathbf{C}$, $\mathbf{C}^*$) with fault injected on line i of the first column of $\mathbf{K_{N_R-2}}$, the round key $\mathbf{K_{N_R}}$, the last column of $\mathbf{K_{N_R-1}}$

Output: $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{i}$

for $i = 0$ **to** 3 **by** 1 **do**
  for $a = 1$ **to** 255 **by** 1 **do**
    $b = Sb(K_{N_R-2}\{i,3\} \oplus a) \oplus Sb(K_{N_R-2}\{i,3\})$
    $c = Sb(K_{N_R-1}\{i,j\} \oplus b) \oplus Sb(K_{N_R-1}\{i,3\})$
    // Compute $K^*_{N_R}$:
    for $j = 0$ **to** 3 **by** 1 **do**
      $K^*_{N_R}\{(i-1)[4], j\} = K_{N_R}\{(i-1)[4], j\} \oplus c$
    end for
    $K^*_{N_R}\{i,0\} = K_{N_R}\{i,0\} \oplus a$
    $K^*_{N_R}\{i,2\} = K_{N_R}\{i,2\} \oplus a$
    // Compute $K^*_{N_R-1}$:
    $K^*_{N_R-1}\{i,3\} = K_{N_R-1}\{i,3\} \oplus b$
    $s_0 = S_{ARK,N_R-2}\{0,3\} \oplus S^*_{ARK,N_R-2}\{0,3\}$
    $s_1 = S_{ARK,N_R-2}\{1,2\} \oplus S^*_{ARK,N_R-2}\{1,2\}$
    $s_2 = S_{ARK,N_R-2}\{2,1\} \oplus S^*_{ARK,N_R-2}\{2,1\}$
    $s_3 = S_{ARK,N_R-2}\{3,0\} \oplus S^*_{ARK,N_R-2}\{3,0\}$
    // Test the assumption:
    if $s_i == a$ *and* $s_j == 0$ *with* $j\ != i$ **then**
      return $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{i}$
    end if
  end for
end for

Only one value $\mathbf{a}$ and one line index $\mathbf{i}$ answer to the previous assumption. Thus, by applying once the **Algorithm 3**, the faulty key $\mathbf{K^*_{N_R}}$ is easily obtained and the second problem is solved. The adaptation on AES-256 does not involve the third problem. The key $\mathbf{K'^*}$ issued from the inverse MixColumns transformation on $\mathbf{K^*_{N_R-1}}$ is integrally faulted. However, as we know the fault value $\mathbf{b}$, we can remove the faults in $\mathbf{K'^*}$ by applying the xor operation with $I\_MC(K_{N_R-1} \oplus K^*_{N_R-1})$.

End of the adaptation

Once the keys $\mathbf{K_{N_R}}$ and $\mathbf{K^*_{N_R}}$ are known, the ciphers $\mathbf{C}$ and $\mathbf{C}^*$ are reversed back up to the output of the round $\mathbf{N_R - 1}$. The inverse MixColumns transformation is applied on both outputs. Thus, we have a new couple ($\mathbf{C'}$, $\mathbf{C'^*}$) corresponding to the couple ($\mathbf{C}$, $\mathbf{C}^*$) without their last round. From that, we are able to discover $\mathbf{I\_MC(K_{N_R-1})}$ instead of $\mathbf{K_{N_R-1}}$.

The faults repercussion on the keys and so on the states is quite different, but the equation (11) is still verified by changing the targeted key and the used couple. We obtain:

$$\mathbf{a} = \mathbf{I\_Sb}(\mathbf{C'}\{\mathbf{i}, (\mathbf{j}-\mathbf{i})[\mathbf{4}]\} \oplus \mathbf{K'}\{\mathbf{i}, (\mathbf{j}-\mathbf{i})[\mathbf{4}]\}) \oplus \mathbf{I\_Sb}(\mathbf{C'^*}\{\mathbf{i}, (\mathbf{j}-\mathbf{i})[\mathbf{4}]\} \oplus \mathbf{K'}\{\mathbf{i}, (\mathbf{j}-\mathbf{i})[\mathbf{4}]\} \oplus \mathbf{b}), \tag{17}$$
$$\text{where j} \in [0;3]$$

The resolution of these equations is easier than for the original attack because the faults values $\mathbf{a}$ and $\mathbf{b}$ are known in this context. The equation presents only one unknown value found by an exhaustive search. The solution to the previous equation is not unique but, once again, the use of two couples reduces the set of solutions to one element.

The whole key $\mathbf{K'}$ is obtained by reiteration of the attack with three other pairs of couples. Each pair comes from a random fault injected on each index of line.

Finally, the $\mathbf{K_{N_R-1}}$ is found with the MixColumns transformation and the initial key can be computed from the inverse KeyExpansion algorithm.

D. Third problem: case of AES-192

The *reproduction phase* of adaptation on AES-192 involves the third problem. In section IV-B, we determined the whole key $\mathbf{K_{N_R}}$ and the last column of $\mathbf{K_{N_R-1}}$. Only four bytes (third column of $\mathbf{K_{N_R-1}}$) miss to finalize the DFA attack. Our adaptation is based on eight new couples, but an exhaustive search is also reasonable.

To be compliant with the *reproduction phase*, a random fault $\mathbf{a}$ is injected on a line noted $\mathbf{i}$ on the first column of $\mathbf{K_{N_R-2}}$. Figure 2 illustrates the propagation whenever the fault occurs on line 1.

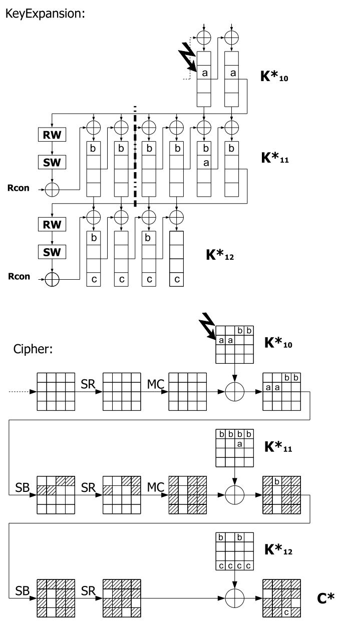

Fig. 2. *Reproduction phase* of AES-192

The KeyExpansion algorithm of AES-192 implies that the RotWord and SubWord transformations are applied on the second column of $\mathbf{K_{N_R-2}}$. So, the fault value $\mathbf{a}$ only impacts the first and second columns of $\mathbf{K_{N_R-2}}$ instead of the entire line $\mathbf{i}$.

Due to the RotWord and SubWord transformations, the fault value $\mathbf{a}$ becomes a new fault named $\mathbf{b}$ such that:

$$\mathbf{b} = \mathbf{Sb}(\mathbf{K_{N_R-2}}\{\mathbf{i},\mathbf{1}\} \oplus \mathbf{a}) \oplus \mathbf{Sb}(\mathbf{K_{N_R-2}}\{\mathbf{i},\mathbf{1}\}) \tag{18}$$

The RotWord and SubWord transformations are not applied on the last word of $\mathbf{K_{N_R-2}}$ during the computation of $\mathbf{K_{N_R-1}}$. Due to the xor operations, the fault value $\mathbf{b}$ is present on the two last columns of $\mathbf{K_{N_R-2}}$ and on the whole line $(\mathbf{i}-\mathbf{1})$ $\mathbf{mod\ 4}$ of $\mathbf{K_{N_R-1}}$. Furthermore, the fault $\mathbf{a}$ of the first column of $\mathbf{K_{N_R-2}}$ is propagated on line $\mathbf{i}$ of the third column of $\mathbf{K_{N_R-1}}$. During the last round key generation, the fault value $\mathbf{b}$ induces a new fault value $\mathbf{c}$ issued from SubWord and RotWord transformations on the last column of $\mathbf{K_{N_R-1}}$. The relation between these two faults can be expressed as:

$$\mathbf{c} = \mathbf{Sb}(\mathbf{K_{N_R-1}}\{(\mathbf{i}-\mathbf{1})[\mathbf{4}],\mathbf{3}\} \oplus \mathbf{b}) \oplus \mathbf{Sb}(\mathbf{K_{N_R-1}}\{(\mathbf{i}-\mathbf{1})[\mathbf{4}],\mathbf{3}\}) \tag{19}$$

Furthermore, line $(\mathbf{i}-\mathbf{1})$ $\mathbf{mod\ 4}$ of round key $\mathbf{K^*_{N_R}}$ is corrupted by fault $\mathbf{b}$ due to the xor operation with $\mathbf{K^*_{N_R-1}}$. As each column of the round key results from a xor operation with the previous column, fault $\mathbf{b}$ only appears on the first and the third columns.

The faults of round key $\mathbf{K^*_{N_R-2}}$ are propagated onto the different internal states during the cipher computation. These faults are transformed and are diffused all over the states. The obtained cipher is not wholly faulted. Depending on the line index $\mathbf{i}$ of the injected fault, the differential of cipher presents the following features:

- $C^*\{1,1\} = 0$ and $C^*\{2,0\} =$ c, when i = 0;
- $C^*\{1,0\} = C^*\{2,3\} = 0$ and $C^*\{3,2\} =$ c, when i = 1;
- $C^*\{3,1\} = C^*\{2,2\} = 0$ and $C^*\{0,0\} =$ c, when i = 2;
- $C^*\{3,0\} = C^*\{0,3\} = 0$ and $C^*\{1,2\} =$ c, when i = 3.

The knowledge of fault value $\mathbf{c}$ present on the differential (C, C^*) is very helpful. Indeed, the round key $\mathbf{K^*_{N_R}}$ is faulted by values $\mathbf{c}$ and $\mathbf{b}$. From the equation (19) and the values of $\mathbf{K_{N_R-1}}$, we determine the fault value $\mathbf{b}$. Furthermore, the line index $\mathbf{i}$ is trivially determined due to the safe bytes localization (see above). It follows that the positions of the different propagated faults are known. In conclusion, the adaptation on AES-192 of C. H. Kim and J.-J. Quisquater's DFA attack is not affected by the second problem.

The knowledge of $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{i}$ is not sufficient to determine the fault value $\mathbf{a}$ present on $\mathbf{K^*_{N_R-1}}$. The equation (18) contains too many unknown values which prevent its resolution.

Furthermore, an entire line of $\mathbf{K^*_{N_R-1}}$ is faulted by $\mathbf{b}$. Thus, $\mathbf{K'}$ issued from the inverse MixColumns of $\mathbf{K^*_{N_R-1}}$ is integrally faulted and some faults are unknown. We cannot remove the faults in $\mathbf{K'^*}$ by applying the xor operation of $I\_MC(K_{N_R-1} \oplus K^*_{N_R-1})$. In conclusion, we are confrontated to the third problem.

The trick

Fortunately, the unknown fault value $\mathbf{a}$ impacts a single byte of $\mathbf{K^*_{N_R-1}}$. As all the elements of a column influence each resulting element of the inverse MixColumns transformation, we do not perform this transformation on the column where the fault $\mathbf{a}$ is present. The inverse MixColumns transformation is only applied on the three other columns (0, 1 and 3) of states $\mathbf{S_{ARK,N_R-1}}$ and $\mathbf{S^*_{ARK,N_R-1}}$.

Furthermore, we are able to determine the bytes of first and second columns of $\mathbf{K_{N_R-1}}$ with the properties of KeyExpansion algorithm and the known bytes of the key. These values are provided respectively by the xor operation between:

- the second and the third columns of $\mathbf{K_{N_R}}$,
- the third and the fourth columns of $\mathbf{K_{N_R}}$.

The columns 0, 1 and 3 of $\mathbf{S_{ARK,N_R-1}}$ and $\mathbf{S^*_{ARK,N_R-1}}$ are deciphered until states $\mathbf{S_{ARK,N_R-2}}$ and $\mathbf{S^*_{ARK,N_R-2}}$. The differential between these two resulting states returns the fault value $\mathbf{a}$. Thereby, the differential $\mathbf{K_{N_R-1} \oplus K^*_{N_R-1}}$ is known and the third problem is solved.

End of the adaptation

In order to finalize the *reproduction phase*, the third column of $\mathbf{K_{N_R-1}}$ needs to be discovered. To do that, the DFA attack could be reproduced. However, in this specific case, we use an easier alternative.

The column 2 of $\mathbf{K_{N_R-1}}$ is expressed from the AES-192 KeyExpansion algorithm as:

$$\mathbf{K_{N_R-1}\{m,2\} = K_{N_R-2}\{m,1\} \oplus K_{N_R-1}\{m,3\}}, \tag{20}$$

$$\text{where m} \in [0;3]$$

The column 3 of $\mathbf{K_{N_R-1}}$ has already been found. When $\mathbf{m}$ is equal to $\mathbf{i}$, we resolve the equation (18) to determine $\mathbf{K_{N_R-2}\{i,1\}}$. The resolution is performed by exhaustive search on $\mathbf{K_{N_R-2}\{i,1\}}$. As the solution is not unique, the equation is solved with a second couple to reduce the set of solutions to one element.

Finally, we are able to solve the equation (20) when $\mathbf{m}$ is equal to $\mathbf{i}$ using two couples impacted by a fault on line index $\mathbf{i}$. Exactly one byte of the searched third column of $\mathbf{K_{N_R-1}}$ is found with two couples, the faults corrupting the same line $\mathbf{i}$. In conclusion, at least eight couples are sufficient to retrieve the entire missing column and finally the initial key.

V. Results and conclusion

Through this paper, we generalize a technique to adapt the DFA attack on AES-128 to the variants AES-192 and AES-256. This technique makes many of the articles published on AES-128 adaptable to these variants. However, the adaptation does not always lead to a solution.

We distinguish two main parts in the adaptation: the first one consists in extending the original attack and the second one in reproducing this attack on an anterior round.

The adaptation of the DFA on KeyExpansion with this methodology is more complex than the DFA on state and each attack has to be considered as a specific case. For this kind of DFA, we evidence three main problems to be solved to obtain enough round key bytes in order to reveal the initial key.

In the case of the C. H. Kim and J.-J. Quisquater's attack, we succeed in adapting the original attack by using two specific tricks. Our adaptation requires 16 couples for both 192-bit and 256-bit keys variants corresponding to the double of AES-128 number of needed couples.

Acknowledgment

This article is the result of work accomplished during a training period at SERMA TECHNOLOGIES ITSEF in 2009. We would like to especially thank all the persons who helped us during our research and all the reviewers without whom this article could not be published.

References

[1] D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Eliminating Errors in Cryptographic Computations," in *EUROCRYPT*, vol. LNCS, vol. 1233. Berlin, Heidelberg: Springer-Verlag, 1997, pp. 37–51.

[2] G. Piret and J.-J. Quisquater, "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD," in *CHES 2003*, vol. 2779. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 77–88.

[3] P. Dusart, G. Letourneux, and O. Vivolo, "Differential Fault Analysis on AES," in *Applied Cryptography and Network Security*, vol. 2846. Berlin, Heidelberg: Springer, 2003, pp. 293–306.

[4] C. Giraud, "DFA on AES," in *Advanced Encryption Standard - AES, 4th International Conference, AES 2004*, vol. 3373. Berlin, Heidelberg: Springer, 2005, pp. 27–41.

[5] D. Mukhopadhyay, "An Improved Fault Based Attack of the Advanced Encryption Standard," in *AFRICACRYPT '09: Proceedings of the 2nd International Conference on Cryptology in Africa*, vol. 5580. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 421–434.

[6] D. Saha, D. Mukhopadhyay, and D. RoyChowdhury, "A Diagonal Fault Attack on the Advanced Encryption Standard," Cryptology ePrint Archive, Report 2009/581, 2005, http://eprint.iarc.org/.

[7] M. Tunstall and D. Mukhopadhyay, "Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault," Cryptology ePrint Archive, Report 2009/575, 2009, http://eprint.iarc.org/.

[8] A. Moradi, M. T. M. Shalmani, and M. Salmasizadeh, "A Generalized Method of Differential Fault Attack against AES Cryptosystem," in *CHES 2006*, vol. 4249. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 91–100.

[9] C.-N. Chen and S.-M. Yen, "Differential Fault Analysis on AES Key Schedule and some Countermeasures," in *ACISP 2003*, vol. 2727. Berlin, Heidelberg: Springer-Verlag, 2008, p. 217.

[10] C. H. Kim and J.-J. Quisquater, "New Differential Fault Analysis on AES Key Schedule: Two Faults Are Enough," in *CARDIS '08: Proceedings of the 8th IFIP WG 8.8/11.2 international conference on Smart Card Research and Advanced Applications*, vol. 5189. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 48–60.

[11] D. Peacham and B. Thomas, "A DFA attack against the AES key schedule," SiVenture White Paper 001, 2006.

[12] J. Takahashi, T. Fukunaga, and K. Yamakoshi, "DFA Mechanism on the AES Key Schedule," in *FDTC '07: Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 62–74.

[13] J. Takahashi and T. Fukunaga, "Differential Fault Analysis on AES with 192 and 256-bit Keys," Cryptology ePrint Archive, Report 2010/023, 2010, http://eprint.iarc.org/.

[14] A. Barenghi, G. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi, "Low Voltage Fault Attacks to AES and RSA on General Purpose Processors," Cryptology ePrint Archive, Report 2010/130, 2010, http://eprint.iarc.org/.

[15] *FIPS 197 : Announcing the Advanced Encryption Standard*, National Institute of Standards and Technology, Novembre 2001.