Differential Analysis of Round-Reduced AES Faulty Ciphertexts

Amir-Pasha Mirbaha, Jean-Max Dutertre, Assia Tria

Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013 IEEE International Symposium on
http://dx.doi.org/10.1109/DFT.2013.6653607

Differential Analysis of Round-Reduced AES Faulty Ciphertexts

Amir-Pasha Mirbaha* Jean-Max Dutertre* Assia Tria†

*†Secure Systems and Architectures (SAS) Department
*École Nationale Supérieure des Mines de Saint-Étienne †CEA-Tech
*†13541 Gardanne, France
{mirbaha, dutertre}@emse.fr assia.tria@cea.fr

***Abstract*— This paper describes new *Round Reduction* analysis attacks on an Advanced Encryption Standard (AES) implementation by laser fault injection. The previous round reduction attacks require both of spatial and temporal accuracies in order to execute only one, two or nine rounds. We present new attacks by more flexible fault injection conditions. Our experiments are carried out on an 8-bit microcontroller which embeds a software AES with pre-calculated round keys. Faults are injected either into the round counter itself or into the reference of its total round number. The attacks may result to the use of a faulty round key at the last one or two executed rounds. The cryptanalysis of the obtained round-reduced faulty ciphertexts resorts to the differentiation techniques used by *Differential Fault Analysis*.**

I. Introduction

Fault attacks consist in using hardware malfunction to infer secrets from the target's faulty behavior or outputs. These active attacks can be performed in different physical manners as reported by [1]. Modifying the behavior of a device's software refers to *algorithm modification*. This class of active attacks may consist in replacing instructions executed by a microcontroller [2] to circumvent its security features, or in weakening the strength of an encryption algorithm by reducing the number of its rounds [3], [4], [5] (*i.e. Round Reduction Analysis* or RRA). [6] proposed an extension of the latter analysis to the *Round Modification Analysis* (RMA). The RMA is based on decreasing or increasing the number of iterative rounds or on altering them to retrieve information about the secret key.

In this paper, we focus on any feasible round reduction analysis of an AES implementation. Altering the round-controlling stored values by laser fault injection requires accuracy on spatial and on temporal positions. Suppose the previous round reduction attacks may not be achieved by an opponent, due to the difficulty of injecting the required fault value. Therefore, is there any other threat by round reduction attacks? We examine these issues in our study. Our experiments are done on an 8-bit microcontroller which embeds a software AES with pre-calculated round keys. Round-reduced ciphertexts are obtained by laser fault injection either into the round counter itself or into the reference of its total round number. In our experiments by faulting the round counter, the last 1 or 2 executed rounds (of the shortened encryptions) is/are xored with faulty round keys. We present a few instances (including two generalized attacks) with their corresponding cryptanalysis. Remarkably, the cryptanalysis of the obtained round-reduced faulty ciphertexts resorts to the differentiation techniques used by *Differential Fault Analysis* (DFA).

This article is organized as follows: Some reminders on the AES and a review of the state-of-the-art of RRA attacks are given in section II. The theory of RRA by laser fault injection is described in section III. The practical basics of our attacks and the experimental setup are presented in section IV. Finally, our findings are summarized in the concluding section V with some perspectives.

II. Round Reduction Analysis

Many symmetric cryptographic algorithms are based on the repetition of identical sequences of transformations, called *rounds*. A significant part of these algorithms' strength against cryptanalysis is based on their repeated rounds. Any decrease in the number of rounds is likely to reduce their security level [7]. For instance, suppose an attack that induces a jump to the end of the algorithm after the execution of only a few instructions (or after one or two rounds). As a result, much of the encryption process is skipped and the final ciphertext is the product of few algorithm operations that may easily reveal the key by light cryptanalysis operations and lower computational complexity. In the following, we first remind the AES' basics and introduce our software implementation before going deeper into the state-of-the-art of round reduction analysis.

A. The Advanced Encryption Standard

1) The AES-*128:* AES is a Substitution-Permutation Network (SPN) block cipher [8]. AES processes a 128-bit plaintext and a key of 128, 192 or 256 bits long to produce a 128-bit ciphertext, by executing 10, 12 or 14 rounds, respectively. For the sake of simplicity, we will consider hereafter only the 128-bit AES version: denoted by AES or by AES-128. The algorithm has two separated processes: One for the `KeyExpansion` to derive round keys from the secret key and another one for the `DataEncryption`. AES-128 performs encryption in 10 rounds, after a short initial round. Each round has its own round key which is used in one of the transformation steps. Hereafter, we use the "K" prefix plus the round number to refer to a round key (*e.g.* "K_9" for the 9^{th} round key). Solely use of "K" refers to the secret key (K=K_0).

To encrypt a plaintext, denoted by M, the encryption process considers its 16 bytes as a matrix of 4×4 bytes.

Each round of the algorithm, except the initial and the final ones, includes 4 transformations: First, the value of each matrix element, *i.e.* one byte value, is exchanged with the corresponding value in a substitution table (`SubBytes` or `SB`). Second, a rotational operation on the matrix rows is executed (`ShiftRows` or `SR`). Third, a linear transformation is applied to each element (`MixColumns` or `MC`). `MC` operates column-by-column and may be written as a matrix multiplication (which coefficients are 01, 02, or 03) in GF(2^8). Forth, a bitwise xor operation is performed between the value of each element and the corresponding byte of the round key (`AddRoundKey` or `ARK`). Before the 1^{st} round, an `ARK` is applied to M and K (*i.e.* `Round 0` or initial round). The `MC` transformation is omitted in the final round. If the algorithm is implemented with iterative rounds, a round counter, hereafter RC, counts the rounds. We use also R_{max} as the reference for total rounds. In a correct typical execution of AES-128, R_{max} is equal 10 and RC is counted from 0 or 1, to 10.

Notation: In the following, we use the "R" prefix plus the round number to refer to the transformations involved in an AES encryption. Hence, R_0-R_1-R_2-R_3-R_4-R_5-R_6-R_7-R_8-R_9-R_{10}, or shortly $R_0 \ldots R_{10}$, represents the rounds of a complete (*i.e.* unmodified) AES. R_1, R_2, ... and R_9 are used to represent unchanged middle rounds, including all the four transformations. R_0 and R_{10} represent the unchanged initial and final rounds of an AES-128, including only `ARK` and `ARK`∘`SR`∘`SB` transformations, respectively. Besides, we use $R_{m=i}$ to express that, due to a fault, a round composed of the `ARK`∘`MC`∘`SR`∘`SB` transformations (where "$_m$" stands for *middle* round) is using an incorrect round key of index i. $R_{f=j}$ has the same meaning for a round without the `MC` transformation ("$_f$" stands for *final* round). In addition, "M_i" represents the AES intermediate state at the end of round i.

2) Software implementations of the AES-*128:* In this work we consider a software implementation of the AES-128, shown in algorithm 1. This implementation is embedded in an 8-bit microcontroller (see section IV). Algorithm 1 is similar to the proposed implementation in the official NIST publication for AES[8]. The only difference lies in the last `ARK` transformation which uses RC value, as the final round key index, instead of the total round number reference (R_{max}).

Algorithm 1 Software implementation of the AES algorithm with pre-calculated round keys.

$C \leftarrow M$
$C \leftarrow C \oplus K_0$
$RC = 1$
while $(RC < R_{max})$ **do**
  $C \leftarrow \texttt{SB}(C)$
  $C \leftarrow \texttt{SR}(C)$
  $C \leftarrow \texttt{MC}(C)$
  $C \leftarrow C \oplus K_{RC}$
  $RC \leftarrow RC + 1$
end while
$C \leftarrow \texttt{SB}(C)$
$C \leftarrow \texttt{SR}(C)$
$C \leftarrow C \oplus K_{RC}$

Where C is an intermediate variable used to memorize the AES state throughout the encryption process. The round counter, RC, is used as an index to select the round key processed during every `ARK` transformation (except the initial one). Moreover, RC is compared to the total round number reference, R_{max}, to end the iterative loop preceding the final round. R_{max} is not a constant number to permit the use of different values (10, 12, or 14 rounds).

Note that the initial and final rounds (R_0 and R_{10}) are implemented outside the iterative loop. Hence, even with complete removal of the middle rounds, the initial and the final rounds will still be executed. Moreover, if for any reason RC takes a value greater than 10, the algorithm will use the 16 bytes stored in the memory that correspond to an address calculated by the same formula for K_{RC}. Consequently, a block of unknown and invalid values will be mapped in the memory and processed by the next `ARK` transformation.

B. Round Reduction Analysis: State-of-the-Art

1) H. Choukri and M. Tunstall's attack: The first practical round reduction analysis experiment was introduced by H. Choukri and M. Tunstall in 2005. [3] shows that a transient glitch on the power supply of a microcontroller may change the round counter (RC) value of an iterative AES cipher. They succeeded in changing the round counter of an AES program at the beginning of algorithm execution to its final value. Hence, the ciphertext was the product of a single round (plus the initial round): R_0-R_m (according to the notation introduced in II-A). They also introduced a cryptanalysis technique that makes it possible to retrieve the secret key: eq. 1 is obtained by xoring two faulty ciphertexts outputs: D^a and D^b (M^a and M^b, being the corresponding plaintexts):

$$\texttt{MC}^{-1}(D^a \oplus D^b) = \texttt{SB}(M^a \oplus K) \oplus \texttt{SB}(M^b \oplus K) \quad (1)$$

For every key byte, eq. 1 yields two different hypotheses. Finally, an exhaustive search over the 2^{16} possible keys is made to retrieve the secret key. Note that this cryptanalysis does not require the knowledge of the correct encryptions of M^a and M^b.

2) Y. Monnet et al.'s attack: Y. Monnet et al. reported in [9] another round reduction attack on two asynchronous cryptoprocessors running the *Data Encryption Standard* (DES) algorithm. The attack was done by laser fault injection. A pool of over 50 reproducible round-corrupted results was obtained. Then, the key was retrieved by differential analysis of faulty encryptions pairs whose round execution sequences have a difference of one round.

3) J.H. Park et al.'s attack: This attack, reported in [4], is a laser fault injection into an ATmega128 8-bit microcontroller with an embedded AES. The AES implementation is compliant with the algorithm structure proposed in [8]. They reported a successful attack that consists in jumping from R_1 to R_{10}. The faulty execution path is R_0-R_1-R_{10}. Therefore, an additional round is executed in comparison to [3] that included only R_0-R_m. The associated cryptanalysis requires data from ten different reduced encryptions. Calculations involved four steps of exhaustive search of 2^{40}, 2^{32}, 2^{24}, and 2^{32} steps respectively. This has taken approximately ten hours on a PC.

4) K.S. Bae et al.'s attack: K.S. Bae et al. presented in [5] a successful attack by eliminating the AES penultimate round. The encryption included $R_0 \ldots R_8$-R_{10}. The attack was performed on the experimental setup introduced in [4]. The key was revealed using two pairs of corresponding faulty and correct ciphertexts and an exhaustive search between the two

candidates for each key byte. This technique is similar to the reported one by H. Choukri and M. Tunstall in [3]. The difference is in using ciphertexts instead of plaintexts. Thus, the cryptanalysis needs finding a key between 2^{16} candidates.

III. Theory of our Round Reduction Analysis

Previous round reduction attacks on AES are based on the analysis of a round-reduced encryption in one, two or nine rounds. Their exploitation requires a differential analysis referring either to the plaintexts or to the correct ciphertexts. Round-reduced ciphertext in the previous attacks did not contain any faulty round key value in their sequences.

In this paper, we study all the feasible attacks by reducing the number of executed rounds. We examine if an AES implementation can be considered safe while the opponent cannot reduce the rounds to one, two or nine. Indeed, is there still a threat if the opponent may reduce the total rounds to between 3 and 8 rounds? Our aim is to identify all the round reduction threats via round controlling operations on an AES implementation.

A. RRA Scenarios

For reducing the number of total rounds in algorithm 1 by fault injection, two scenarios are conceivable: changing the R_{max} or altering the RC value. We assume in this paper a bit-flip fault model. We show the injected fault by "e". Consequently, e is always single-byte and xored either with R_{max} or with RC. Few more potential scenarios exist by finding some external targets; for instance, the chip's `program counter`. In this article, we cannot include all the potential targets for algorithm 1. Thus, we focus only on the fault attacks into R_{max} and into RC values which are more common for an AES implementation similar to the NIST proposed version [8].

1) Scenario I: Attacks on the round number reference: Suppose a fault injection into R_{max} during AES execution. R_{max} is accessed only once per round, at the beginning of the `while` loop in algorithm 1. Depending on the resulting $R_{max} \oplus e$ value and on the RC value during the attack, an increase or a decrease may be induced in the total round number:

Case a: If $R_{max} \oplus e < R_{max}$ and $RC < R_{max} - 1 \Rightarrow$ *Round reduction* or *suppression of one or several rounds* to $\texttt{Max}\{R_{max} \oplus e,\ RC{+}1\}$ rounds. For instance: if e=8 is injected, when RC = 5, then $R_{max} \oplus e$=2 and the AES execution will be: $R_0 \ldots R_5$-$R_{f=6}$. Therefore, as $R_{max} \oplus e < RC$, the following round after the attack will be executed as the final round and any remaining round will be suppressed.

Case b: If $R_{max} \oplus e > R_{max}$ and $RC < R_{max} \Rightarrow$ *Round addition* or *execution of additional rounds*. For instance: if e=4 is injected, when $RC < 10$, then $R_{max} \oplus e$=14 and the AES execution will be: $R_0 \ldots R_9$-$R_{m=10}$-$R_{m=11}$-$R_{m=12}$-$R_{m=13}$-$R_{f=14}$. The 10^{th} round and the additional rounds 11 to 13 will be executed similar to middle rounds (*i.e.* including `MC` step). Then, the final round is executed as R_{14}. With pre-calculated key option, and in absence of countermeasures, the rounds above R_{10} will use invalid round key values $K_{11} \ldots K_{14}$ in their `ARK` transformation. The total number of executed rounds will be incremented to 14.

Faulting R_{max} when $RC < R_{max} - 1$ results always in round reduction or addition. A fault in R_{max} cannot cause a redundant execution of any round.

2) Scenario II: Attacks on the round counter value: The second scenario is like the attacks reported in [3], [4] and [5], by targeting the round counter, RC, during the AES execution. The attack alters the index of the current executing round. In an algorithm similar to algorithm 1, RC is accessed two times during each middle and final round execution. Thus, depending on the instant of fault injection, various changes can occur to the encryption process: An RC change before `ARK` alters the index of the executing round and thus, the index of the mapped round key. Besides, RC modification during `ARK` has not any immediate effect until the next RC incrementation in the `while` loop; *i.e.* beginning of the following round.

An RC variation often leads to a change in the number of total executing rounds, by adding, by suppressing or even by repetitively executing of several rounds:

Case a: If $RC \oplus e < RC \Rightarrow$ *Round addition* or *repetitive execution of several rounds*. For instance: if $RC = 7$ and $e = 2$ then $RC \oplus e = 5$. If the fault is injected before the `ARK`, AES execution will be: $R_0 \ldots R_5$-R_6-R_5-R_6-$R_7 \ldots R_{10}$. Thus, R_5 and R_6 will be executed twice and the total number of executed rounds will be incremented to 12. An attack during the `ARK` affects the encryption on the following round. In this example, R_6 and R_7 (instead of R_5 and R_6) will be repeated and AES execution will become: $R_0 \ldots R_5$-R_6-R_7-R_6-$R_7 \ldots R_{10}$.

Case b: If $RC \oplus e > RC$ when $RC < R_{max} - 1 \Rightarrow$ *Round reduction*. For example: if $RC = 4$ and $e = 2$ (before `ARK`) then $RC \oplus e = 6$ and the faulty AES execution will be :$R_0 \ldots R_3$-$R_6 \ldots R_{10}$. Therefore, R_4-R_5 will be skipped and the total number of executed rounds will be reduced to 8.

Case c: If $RC \oplus e > RC$ when $RC = R_{max} - 1 \Rightarrow$ *Round alteration*: the total number of rounds remains unchanged, but the attack has effects on the following `ARK` transformations. For instance: if $RC = 9$ and $e = 2$ is injected before `ARK`, then $RC \oplus e = 11$ and AES execution will be: $R_0 \ldots R_8$-$R_{m=11}$-$R_{f=12}$. Consequently, the total number of executed rounds will remain 10, but the penultimate round and the final round will use invalid round keys values (K'_{11} and K'_{12}) during their `ARK` transformations. When the round keys are pre-calculated, K'_{11} and K'_{12} are mapped to the memory contents which exclude any information about K.

As it is shown, contrary to the attacks on R_{max}, faulting RC may cause redundant execution of one or several rounds. In this paper, we focus only on the round reduction attacks.

B. Round Reduction Experiments

Now, we examine all the round reduction attack possibilities by the two scenarios on our AES implementation, using the experimental setup described in section IV-B.

1) Round Reduction to the Initial Round: A round reduction attack which reduces AES encryption to only the initial round transformation, is theoretically, the most optimal attack. Because, the round-reduced ciphertext (shown as D)

is produced with the minimal transformation on the plaintext (shown as M) containing all the key information:

$$D = M \oplus K \tag{2}$$

Thus, the attack analysis requires the knowledge of only one pair of a round-reduced ciphertext and its corresponding plaintext. The key is revealed by an xor between them. This attack seems to be infeasible on an AES implementation similar to algorithm 1.

2) Round Reduction to One Round: Round reduction to only one round (after the initial round) is maybe the optimal realistic attack on several AES implementations. Because, the encryption involves only one round transformations. Thus, the results can be exploited very fast. The analysis requires two pairs of corresponding round-reduced ciphertext and plaintext.

The first realization of this attack is reported by H. Choukri and M. Tunstall in [3]. We refer the readers to their article for more details about their technique. This attack seems to be feasible, only by targeting R_{max} on an AES implementation, similar to algorithm 1.

3) Round Reduction to Two Rounds (Experiment 1): A round reduction to two rounds (after the initial round) is a realistic attack (and maybe the optimal one) on the NIST proposed implementation. This attack is exploitable in a feasible calculation time. J.H. Park et al. reported a successful realization of this attack in [4]. Their technique requires 10 pairs of corresponding round-reduced ciphertext and plaintext. Their analysis takes 10 hours on a PC. Here, we report our round reduction attacks to two rounds as our experiment 1. Our technique has more performance in comparison to [4].

a) Experiment 1.a Using Scenario I: By injection an adequate fault value into R_{max}, the opponent may reduce the encryption sequence to 2 rounds, *i.e.*: R_0-R_1-$R_{f=2}$. [4] can successfully exploit this attack, even if the encryption sequence differs at the final round. We review a more complex case in experiment 1.b for scenario II and then present our technique.

b) Experiment 1.b Using Scenario II: We performed this attack by faulting RC during the execution of R_1. If the fault is injected during `ARK` of R_1, the encryption sequence becomes: R_0-R_1-R_f. [4] may again be used for the exploitation. Besides, if the fault is injected before `ARK` of R_1, the encryption sequence becomes R_0-R_m-R_f. This is a more sophisticated case in comparison to J.H. Park et al.'s attack. We present here a technique exploiting our attack which can be also applied to J.H. Park et al.'s attack.

This experiment is done by faulting the RC during the first round in order to increment it to 9 or a higher value. This was achieved with a single-bit or single-byte fault equal to `0x08` or any other value between `0x0a` and `0xff`. Thus, $RC \oplus e$ became bigger than 8 and the next round was performed as the final round. Therefore, the encryption sequence of rounds was: R_0-R_m-R_f with faulty key values for the second and the final rounds. The cryptanalysis of this attack scheme requires at least three pairs of plaintexts and corresponding faulty ciphertexts (M^a,D^a), (M^b,D^b) and (M^c,D^c). Considering a plaintext M^a, the corresponding faulty ciphertext is given by eq. 3:

$$D^a = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^a \oplus K) \oplus K'_x] \oplus K'_y \tag{3}$$

In eq. 3, K'_x and K'_y are unknown constant values corresponding to invalid round keys, for the rounds R_m and R_f, respectively. By xoring eq. 3 and the similar equation for D^b, we discard K'_y values and we obtain eq. 4. We repeat the xoring for eq. 3 and for D^c equation.

$$\begin{aligned} D^a \oplus D^b = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^a \oplus K) \oplus K'_x] \\ \oplus \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^b \oplus K) \oplus K'_x] \end{aligned} \tag{4}$$

Nevertheless, K'_x remained in eq. 4 and in corresponding equation for $D^a \oplus D^c$. The solution for resolving them consists in creating hypotheses on both of K and K'_x where M^a, M^b, M^c, D^a, D^b, and D^c are known values. Therefore, a brute-force search of $(2^8)^4 \times 2^8 \times 4 = 2^{42}$ values is necessary for each column of $\texttt{SR}(K)$. The search results to only 1, 2 or 3 hypotheses for each corresponding column of $\texttt{SR}(K)$ and K'_x. At the next step, each combination of key column hypotheses can be examined by using one of the pairs of corresponding plaintext and correct ciphertext; *e.g.* M^a and C^a. The entire key is discovered after a brute-force search of $2^{42} \times 4 = 2^{44}$ values. With a PC running with an Intel Core i5-2410M microprocessor at 2.30GHz (hereafter our PC), the full attack exploitation takes about 3 hours and 50 minutes. In our experiment, except the initial round key (*i.e.* K), any round key (*i.e.* K_m or K_1 and also K_f or K_{10}) may be inexistent or fully faulty. However, our method is about 2.5 times faster and needs only 3 encryptions, instead of 10 required encryptions in Park et al.'s technique.

4) Round Reduction to Three Rounds: Reducing AES execution to three rounds (after the initial round) is realistic in both of our implementations. However, we could not find any differential technique in a relatively calculation time ($<\sim 10$ hours).

5) Differential One-Round Analysis of the Round-Reduced Encryptions (Experiment 2): Suppose an attack that changes the number of executed rounds during two consecutive encryption of a unique plaintext. If the two obtained round-reduced encryptions differ in only one round, the corresponding plaintext or correct ciphertext are no longer required for the exploitation. A differential analysis between the two round-reduced encryptions may reveal the key.

We assume the first attack reduces the number of executed rounds from R_{max} to $i+1$, as a lesser value. Then, the second attack reduces a new encryption of the same plaintext to $i+2$. The sequences of round-reduced encryptions are:

$$\begin{array}{rll} 1^{st}\,encryption: & R_0 \ldots R_i\text{-} & \boxed{R_{f=i+1}} \\ 2^{nd}\,encryption: & R_0 \ldots R_i\text{-} & \boxed{R_{i+1}\text{-}R_{f=i+2}} \end{array}$$

Please note that i is the index of the last executed middle round in the first round-reduced encryption. A differential cryptanalysis over only the framed part of the round-reduced encryptions may reveal the key. We denote two consecutive round-reduced encryptions for a given plaintext M^a by D^a_1 and D^a_2. We show them by eq. 5a and eq. 5b:

$$D^a_1 = \texttt{SR} \circ \texttt{SB}(M^a_i) \oplus K_{i+1} \tag{5a}$$

$$D^a_2 = \texttt{SR} \circ \texttt{SB}(M^a_{i+1}) \oplus K_{i+2} \tag{5b}$$

K_{i+1} and K_{i+2} are unknown values corresponding to valid round keys for the rounds R_{i+1} and R_{i+2}, respectively. By

using two plaintexts, a xor operation between two D_1 and another xor operation between two D_2 equations lead to eq. 6 and eq. 7:

$$D_1^a \oplus D_1^b = \texttt{SR} \circ \texttt{SB}(M_i^a) \oplus \texttt{SR} \circ \texttt{SB}(M_i^b) \quad (6)$$

$$D_2^a \oplus D_2^b = \texttt{SR} \circ \texttt{SB}(M_{i+1}^a) \oplus \texttt{SR} \circ \texttt{SB}(M_{i+1}^b) \quad (7)$$

Eq. 8 is obtained by applying a `MC` transformation to eq. 6 (given the `MC` distributivity property over $\oplus$):

$$M_{i+1}^b = \texttt{MC}(D_1^a \oplus D_1^b) \oplus M_{i+1}^a \quad (8)$$

By replacing M_{i+1}^b from eq. 8 in eq. 7, we obtain eq. 9:

$$\begin{aligned} D_2^a \oplus D_2^b = \texttt{SR} \circ \texttt{SB}(M_{i+1}^a) \oplus \\ \texttt{SR} \circ \texttt{SB}[\texttt{MC}(D_1^a \oplus D_1^b) \oplus M_{i+1}^a] \end{aligned} \quad (9)$$

where D_1^a, D_1^b, D_2^a and D_2^b have known values. Hence, we perform an exhaustive search over 2^8 possible values for each M_{i+1}^a byte. This search leads often to a unique value for each byte. Then, by using the obtained M_{i+1}^a values and eq. 5b, we find K_{i+2}. Therefore, K is revealed using the inverse of `KeyExpansion`.

a) Experiment 2.a Using Scenario I: By using the experimental setup reported in section IV-B, we performed successfully this attack. We reduced R_{max} to several values and obtained K by using the technique explained above.

b) Experiment 2.b Using Scenario II: This attack using scenario II results to two round-reduced encryptions without any valid key at the final rounds. In order to exploit the round-reduced encryptions with short cryptanalysis solutions, the fault must be injected during `ARK` transformation of the last middle round (of each round-reduced encryption). Otherwise (*i.e.* with an earlier fault, before `ARK`), the faulty RC alters key values during two rounds. Therefore, we obtain eq. 10a and eq. 10b, instead of eq. 5a and eq. 5b, respectively for a given plaintext M^a:

$$D_1^a = \texttt{SR} \circ \texttt{SB}(M_i^a) \oplus K'_{y1} \quad (10a)$$

$$D_2^a = \texttt{SR} \circ \texttt{SB}(M_{i+1}^a) \oplus K'_{y2} \quad (10b)$$

A differential cryptanalysis between eq. 10a and eq. 10b leads to finding M_{i+1}^a. However, K'_{y2} is an invalid key, unrelated to K. Thus the analysis cannot release any information for finding K. A solution for exploiting scenario II is to expand this experiment with a third attack, and thus, obtaining a third round-reduced encryption to $i+3$ rounds, denoted by eq. 11:

$$D_3^a = \texttt{SR} \circ \texttt{SB}(M_{i+2}^a) \oplus K'_{y3} \quad (11)$$

Thus, M_{i+2}^a is revealed by a differential cryptanalysis between eq. 10b and eq. 11. Consequently, K_{i+2} is obtained using M_{i+1}^a and M_{i+2}^a values in eq. 12:

$$K_{i+2} = M_{i+2}^a \oplus \texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_{i+1}^a) \quad (12)$$

Therefore, two separate differential cryptanalyses by using three round-reduced encryptions of D_1, D_2 and D_3 for 3 plaintexts M^a, M^b and M^c reveal the secret key. The calculation time is less than one second with our PC.

c) Experiment 2.c Using Scenario II: Another option exists for this experiment by using only 2 given plaintexts M^a and M^b, instead of 3. In this case, the cryptanalysis between D_1 and D_2 (for two corresponding pairs) reveals often two hypotheses for each pair of corresponding M_{i+1}^a and K'_{y2} bytes at the first step. Then, the cryptanalysis between D_2 and D_3 (for two corresponding pairs) reveals often two hypotheses for each byte of M_{i+2}^a. At the third step, each 2 hypotheses for one byte on M_{i+1}^a creates 2^4 hypotheses for its corresponding column values. Thus, with 2 hypotheses for each of corresponding bytes on M_{i+2}^a, 2^8 hypotheses are made for the corresponding bytes on K_{i+1}. Therefore K is revealed after testing all the $(2^8)^4$ hypotheses for entire K_{i+1}. This exploitation needs calculation and testing of other round keys for each of the hypotheses. It needs in average less than 3 hours and 30 minutes with our PC.

d) Experiment 2.d Using Both of the Scenarios I & II: Another possibility for performing this attack, is using both of the scenarios I and II, in a manner that the first attack targets the RC and the second attack changes the R_{max}. In this case, eq. 13a and eq. 13b present the faulty encryptions for a given M^a:

$$D_1^a = \texttt{SR} \circ \texttt{SB}(M_i^a) \oplus K'_y \quad (13a)$$

$$D_2^a = \texttt{SR} \circ \texttt{SB}(M_{i+1}^a) \oplus K_{i+2} \quad (13b)$$

K'_y is an invalid round key. However, this attack exploitation is similar to experiment 2.a using scenario I. Please refer to section III-B5 for this attack exploitation. This experiment with *combined scenarios* shows the threat of existing flexible targets for the opponent in order to obtain the required encryption sequences.

6) Differential Two-Round Analysis of the Round-Reduced Encryptions (Experiment 3): Another attack possibility is a differential analysis between round-reduced encryptions which differ in two rounds. For brevity in this article, we report this attack only in table I.

7) Round Reduction to Eight Rounds (Experiment 4): Here, we review round reduction attack to 8 rounds.

a) Experiment 4.a Using Scenarios I: It is possible to modify R_{max} in order to execute 8 rounds. The encryption includes all the rounds until R_7, then it performs R_8 as the final round, *i.e.* without `MC`: $R_0 \ldots R_7\text{-}R_{f=8}$. The attack can be exploited as a simple case by our technique for scenario II (experiment 4.b).

b) Experiment 4.b Using Scenario II: This is a round reduction experiment by targeting RC during `ARK` of R_7 with any fault value greater than `0x07`, except `0x0f`. Due to the fault, RC was incremented to R_{max} or a higher value at the beginning of following round. Therefore, the encryption sequence was: $R_0\text{-}R_1 \ldots R_7\text{-}R_f$. This attack exploitation needs at least three pairs of correct and round-reduced ciphertexts: (C^a,D^a), (C^b,D^b) and (C^c,D^c).

$$C^a = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (14)$$

$$D^a = \texttt{SR} \circ \texttt{SB}(M_7^a) \oplus K'_y \quad (15)$$

An xor between D^a and D^b, followed by a `MC` gives eq. 16:

$$\texttt{MC}(D^a \oplus D^b) = \texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_7^a) \oplus \texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_7^b) \quad (16)$$

Eq. 16 can be also expressed by M_8 values, as shown in eq. 17:

$$\texttt{MC}(D^a \oplus D^b) = M_8^a \oplus M_8^b \tag{17}$$

By xoring eq. 14 with similar eq. for C^b, and reversing `ShiftRows` operations, we obtain eq. 18:

$$\texttt{SR}^{-1}(C^a \oplus C^b) = \texttt{SB}(M_9^a) \oplus \texttt{SB}(M_9^b) \tag{18}$$

By replacing M_9^a and M_9^b from eq. 17 in eq. 18, we obtain eq. 19:

$$\begin{aligned}\texttt{SR}^{-1}(C^a \oplus C^b) = \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_8^a) \oplus K_9] \oplus \\ \texttt{SB}\{\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}[\texttt{MC}(D^a \oplus D^b) \oplus M_8^a] \oplus K_9\}\end{aligned} \tag{19}$$

In eq. 19, C^a, C^b, D^a and D^b have known values. We perform an exhaustive search among $(2^8)^4$ possible values for each $\texttt{SR}(M_8^a)$ column and its corresponding column of K_9 values. By using three corresponding pairs of round-reduced ciphertexts, in average, two column values for each column of K_9 can be found. Then, K_9 and consequently K can be found by testing all the 2^4 column hypotheses. This attack exploitation is similar to the experiment 3, reported in table I. The complete cryptanalysis takes about 4 hours and 10 minutes with our PC.

8) Round Reduction to Nine Rounds: A round reduction to nine rounds (after the initial round) can be considered as a similar attack of round reduction to one round, reported in [3].A successful experiment of this attack is reported by K.S. Bae et al. in [5].

IV. Laser induced RRA on the microcontroller

A. Laser fault injection

The use of a laser to inject faults into the calculations of a secure circuit was introduced by S. Skorobogatov and R. Anderson in 2002 [10]. Laser faults arise from the photoelectric effect caused by a laser beam passing through silicon provided that its photon energy is greater than the silicon bandgap [11]. This effect generates electron-hole pairs in silicon. These charges may create a transient current when exposed to the strong electric fields found in the PN junctions of CMOS transistors. Then, this transient current turns into a voltage transient that may travel through the circuit's logic. It may affect the computations of the target circuit or some of its memory elements. Hence, SRAMs are subject to bit-flip when exposed to a laser beam [10], [12]. We have reported in [13] experiments showing our ability to inject single byte and even single bit faults in the SRAM of the same device. We took advantage of the knowledge acquired during this previous work to realize the experiments reported in this section.

B. Experimental Setup for laser induced RRA

Target: We used a device communicating with smart card standards as our target. It is an 8-bit $0.35\mu m$ RISC microcontroller with an integrated 128KB flash program memory, 4KB EEPROM and 4KB SRAM. It should be noted that this microcontroller does not embed any countermeasure against fault injection. The device runs the *Simple Operating System for Smartcard Education* [14] for simulating the smart card environment. The microcontroller operates at a frequency of 3.59 MHz. It runs the software AES described in algorithm 1 (in section II-A2) which uses pre-calculated round-keys. In our implementation, the AES secret key is embedded in the code. After each circuit reset, the AES' round keys are derived and stored in the microcontroller's SRAM.

RC and R_{max} are also stored in the circuit's SRAM. That is the entry point we have used to modify the AES behaviour by laser fault injection. We have already reported ([13], [15]) bytewise and bitwise fault injection experiments in this SRAM. As a consequence, laser proved to be a suitable fault injection means in order to corrupt RC and R_{max} for the purpose of carrying out RRA. Figure 1 highlights the chip's SRAM area.

Bench: Our experiments were conducted with green (532nm) or infrared (1064nm) wavelengths, through the front and rear sides of the chip respectively (obviously after a proper decapsulation). The laser beam for injecting single-bit faults was about $\varnothing 4\mu$m with $\simeq 10pJ$ energy per shot (before passing through the lens). The laser pulse duration was set to 5ns. The target was mount on an XY motorized stage with a 0.1μm resolution. It makes it possible to scan the target's SRAM with high accuracy. Figure 1 shows the circuit installed on the laser bench. A synchronization card provides a jitter of 10ns at

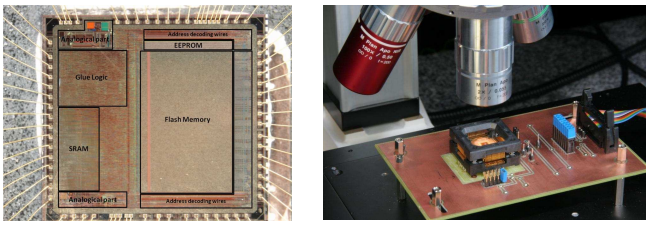

Fig. 1. *On the left:* View of the 8-bit microcontroller and of its SRAM area. *Right:* The target circuit, installed on the laser bench for front side injection.

the instant of injection. Hence, given the clock period of the device, 280ns, a very precise timing is achieved.

C. Realizations

One key point in performing RRA is to find out the kind of round modification induced by the fault injection (*i.e.* the number of rounds actually processed). It was achieved by precisely measuring the time elapsed between the end of the encryption command (sent by our communication interface) and the beginning of the card status answered by the test chip. Each encryption command includes a plaintext. The card's chip receives the command and then encrypts the plaintext. As soon as the encryption finishes, the card's chip sends the status "`61 10`" to the reader. This status means that the encryption was normally completed and that `0x10` bytes of extra data are available (this number of bytes refers to the ciphertext's length).

In our implementation and experiments, any change in the total number of rounds is related only to a decrease (or eventually an increase) of the middle rounds number. Therefore, the time elapsed between the end of the encryption command and the beginning of the status on the I/O signal is a function of the number of middle rounds actually processed. Figure 2 shows the differences between the duration of the circuit's I/O and power consumption signals for a normal AES encryption, for one shortened encryption to only two rounds (after the initial round), and for a round-reduced encryption

to 8 rounds. The I/O signal remains constantly high for about 2.20ms during the AES encryption. Before this interval, the last byte of plaintext has been sent to the circuit (from the reader). Then, just after the AES encryption, the first byte of the card status is sent to the reader (from the circuit). The observed time difference between the correct encryption and the reduced ones is denoted by Δt. For a decrease of 8 middle rounds, Δt is about 1.48ms. When the encryption skips two rounds, Δt is about 0.37ms. Thus, we were always able to discover any decrease (or even increase) in the round number by comparison with an unmodified encryption. We also monitored the chip's power consumption for checkout purposes.

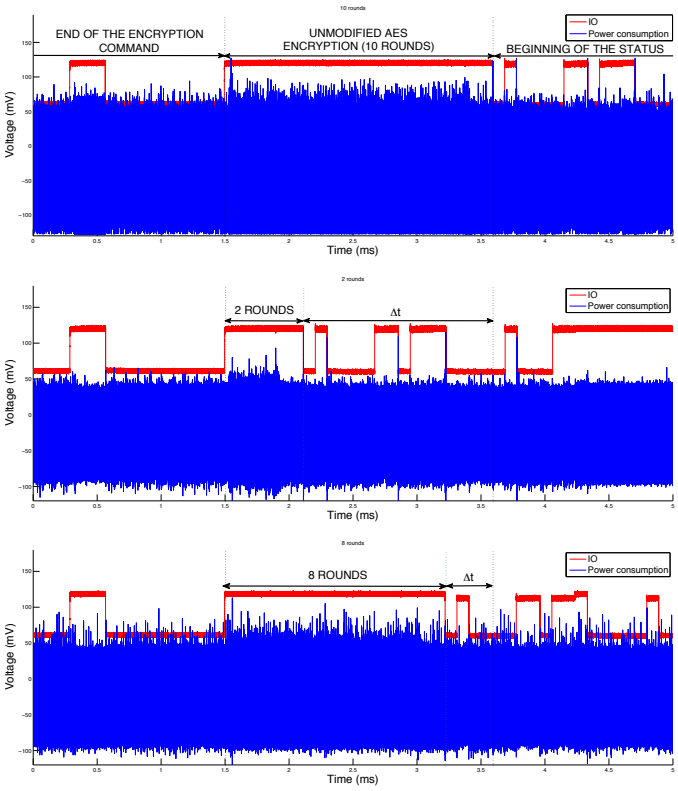

Fig. 2. The I/O and power consumption signals for three encryptions of a same plaintext, given from top to bottom for a correct, a 2-rounds and an 8-rounds encryptions.

Target points: Finding R_{max} and RC on the circuit is a sensitive step for carrying out the experiments. We realized during our previous experiments that this SRAM is implemented by blocks of 512 bytes. In each block of bytes, the bits of same index are implemented together for all the bytes. Thus, for each byte, the corresponding bits are distributed in 8 blocks of bits on the Y-axis with a fixed distance between them. Besides, each line on the Y-axis contained the bits of several bytes. We refer our reader to [10] for a detailed experiment reported by S. Skorobogatov on the SRAM of another microcontroller.

For finding the target points, we scanned the spatial coordinates for high value bits (*i.e. bits 4, 5, 6 and 7 for the faults equal to* $2^4, 2^5, 2^6$ *or* 2^7*, respectively*) by laser fault injection in the middle of AES encryption (preferably $R_4 \ldots R_6$). As soon as a fault was injected either into the R_{max} or into the RC, a significant round modification was induced. If the RC was faulted, the encryption was stopped one round later. Otherwise, if the fault was injected into the R_{max}, the encryption was increased to 26 rounds or more. Therefore, we found the spatial coordinates of both of the targets on the X-axis. For finding the other bits of RC and R_{max}, we displaced the laser spot on the Y-axis and found their coordinates. Please refer to figure 3. We needed to enlarge the beam on the Y-axis to $\sim 45\mu$m (or respectively to $\sim 90\mu$m) in order to inject faults on 2 (or 3 resp.) neighboring bits. No additional opening above the initial 4μm on the X-axis was necessary.

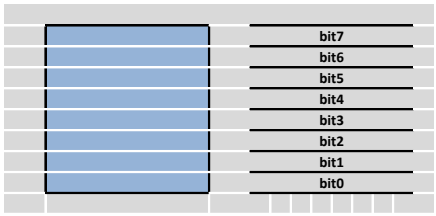

Fig. 3. A SRAM block of 512 bytes on our circuit. The bits of each byte are implemented on distinct rows.

Another key point in performing RRA is the ability to induce several times the same fault (what we shall call *fault repeatability*). Indeed, the RRA schemes introduced in section III require to gather faulty ciphertexts obtained from an identically modified sequence of rounds but with a different plaintext. To ascertain the fault repeatability rate we were able to obtain, we have run several fault injections with the same experimental settings (including the data: plaintext and key). When the laser was positioned at correct coordinates of our target (either R_{max} or RC) thanks to the accurate X-Ystage, faults are injected into the target itself and often a very limited set of few neighboring bytes without any effect on the encryption. In at least 30% of irradiations, a fault is injected into the target. By using the above-mentioned laser spot (and even larger spots up to 10 times greater), all the injected faults were only single-bit faults. Thus, they were identical, as required in our attacks. Therefore, in almost all the successful single-bit fault injection into the target at a fixed X-Y stage coordinates, the fault value was repeated. According to several SRAM architectures (*e.g.* in our microcontroller), the bits of a unique value are designed and built close together for a block of bytes in the memory array. In theses implementations, usually the distance of two bit cells of same index in a block of bytes (*e.g.* 256 or 512 bytes) is much closer than the distance of a bit with its neighbor bits of the same byte. This is a weakness for the security of SRAM contents against single-bit fault injection. Please refer to our results reported in [15] for more details. Among various modifications of the AES algorithm obtained by laser fault injection, we have chosen to report the 4 most significant round reduction attacks (including two generalized analyses). All these experiments were realized by single-bit laser fault injection and accurate time-tuning of the laser irradiations. However, they can be extended to multiple-bit faults.

V. Conclusion and Perspectives

In this paper we introduced some new round reduction analysis attacks on an AES implementation, similar to the proposed one by NIST in [8]. Our experiments were carried on

TABLE I. COMPARISON BETWEEN PREVIOUS WORKS AND OUR MOST SIGNIFICANT EXPERIMENTS.

Attack	Target	Attack time	Execution(s)	Required texts	Key search average runtime
H. Choukri et al. [3]	RC	Beginning of R_1	R_0-R_m	2	$\simeq$ 1 second
J.H. Park et al. [4]	RC	During R_1	R_0-R_1-R_{10}	10	$\simeq$ 10 hours
K.S. Bae et al. [5]	RC	During R_8	$R_0 \dots R_8$-R_{10}	2	$\simeq$ 1 second
Experiment 1.a	R_{max}	During R_0-R_1, if e=0x08; during R_1, if $e \in \{0x0a, 0x0b\}$	R_0-R_1-$R_{f=2}$	3	$\simeq$ 3 hours and 55 minutes
Experiment 1.b	RC	During R_1	R_0-R_m-R_f	3	$\simeq$ 3 hours and 55 minutes
Experiment 2.a	R_{max}	During $R_0 \dots R_i$ or only R_i, according to e_1 (for D_1) During $R_0 \dots R_{i+1}$ or only R_{i+1}, according to e_2 (for D_2)	$R_0 \dots R_i$-$R_{f=i+1}$ & $R_0 \dots R_{i+1}$-$R_{f=i+2}$	2	$\simeq$ 1 second
Experiment 2.b	RC	During ARK of R_i (for D_1) During ARK of R_{i+1} (for D_2) During ARK of R_{i+2} (for D_3)	$R_0 \dots R_i$-$R_{f=y1}$ & $R_0 \dots R_{i+1}$-$R_{f=y2}$& $R_0 \dots R_{i+2}$-$R_{f=y3}$	3	$\simeq$ 1 second
Experiment 2.c	RC	During ARK of R_i (for D_1) During ARK of R_{i+1} (for D_2) During ARK of R_{i+2} (for D_3)	$R_0 \dots R_i$-$R_{f=y1}$ & $R_0 \dots R_{i+1}$-$R_{f=y2}$& $R_0 \dots R_{i+2}$-$R_{f=y3}$	2	$\simeq$ 3 hours and 30 minutes
Experiment 2.d	RC & R_{max}	During ARK of R_i (for D_1) During $R_0 \dots R_{i+1}$ or only R_{i+1}, according to e_2 (for D_2)	$R_0 \dots R_i$-$R_{f=y}$ & $R_0 \dots R_{i+1}$-$R_{f=i+2}$	2	$\simeq$ 1 second
Experiment 3	R_{max}, RC or R_{max} & RC	According to target and e_1 (for D_1) According to target and e_2 (for D_2)	$R_0 \dots R_i$-$R_{f=y1}$ & $R_0 \dots R_{i+2}$-$R_{f=y2}$	3	$\simeq$ 4 hours and 10 minutes
Experiment 4.a	R_{max}	During $R_0 \dots R_7$, if e=0x02; during R_7, if 0x08$\leq e \leq$0x0f	$R_0 \dots R_7$-$R_{f=8}$	3	$\simeq$ 4 hours and 10 minutes
Experiment 4.b	RC	During ARK of R_7	$R_0 \dots R_7$-R_f	3	$\simeq$ 4 hours and 10 minutes

by laser fault injection. *Round Reduction* analysis techniques based on reducing the AES round number to 1, 2 or 9 were previously proposed. However, it may be a difficult task for an attacker to successfully induce faults that make possible RRA by jumping directly to the AES end from its very beginning. This may lead secure designers to underestimate the risk of such an *algorithm modification* attack or to setup incomplete countermeasures. We intend in this article to issue a warning by reporting different round reduction cases. Many cryptanalysis techniques exist (sometimes relatively easy to set up) which makes it possible to retrieve the AES key from erroneous outputs of a round-modified execution. Table I shows a comparison between previous works and our most significant experiments reported in this paper.

It should be noted that most of the results we have obtained depend on the AES implementation we used: `KeyExpansion` performed once prior to the encryptions (in order to save both computation time and power consumption). However, even if some of the scenarios we have presented may become impracticable, similar cryptanalysis may be derived for an AES implementation using on-the-fly key scheduling (to save memory consumption). Besides, if a fully unrolled AES (*i.e.* without any loop) is immune to RRA through RC or R_{max} modification, RRA should still be performed by faulting the program counter of the microcontroller. Moreover, the `KeyExpansion` iterative process is also a potential target. Further work has to be done based on these findings in order to propose countermeasures against RRA.

REFERENCES

[1] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, 2012.

[2] J. Balasch, B. Gierlichs, and I. Verbauwhede, "An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs," in *Fault Diagnosis and Tolerance in Cryptography – Proceedings of FDTC'2011*. IEEE, 2011, pp. 105–114.

[3] H. Choukri and M. Tunstall, "Round reduction using faults," *Fault Diagnosis an Tolerance in Cryptography FDTC 2005*, pp. 13–24, 2005.

[4] J. Park, S. Moon, D. Choi, Y. Kang, and J. Ha, "Differential fault analysis for round-reduced AES by fault injection," in *ETRI Journal*, vol. 33. ETRI, pp. 434–442.

[5] K. Bae, S. Moon, D. Choi, Y. Choi, D. Choi, and J. Ha, "Differential fault analysis on AES by round reduction," in *Computer Sciences and Convergence Information Technology – Proceedings of ICCIT'2011*. IEEE, pp. 607–612.

[6] J.-M. Dutertre, A.-P. Mirbaha, D. Naccache, A.-L. Ribotta, A. Tria, and T. Vaschalde, "Fault round modification analysis of the Advanced encryption standard," in *Hardware-Oriented Security and Trust – Proceedings of HOST'2012*. IEEE, 2012, pp. 140–145.

[7] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," in *Security Protocols – Proceedings of SPW'1998*, ser. LNCS. Springer-Verlag, 1998, vol. 1361, pp. 125–136.

[8] NIST, "Announcing the Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication, n. 197, nov 2001.

[9] Y. Monnet, M. Renaudin, R. Leveugle, C. Clavier, and P. Moitrel, "Case study of a fault attack on asynchronous DES crypto-processors," in *Fault Diagnosis and Tolerance in Cryptography – Proceedings of FDTC'2006*, ser. LNCS, vol. 4236. Springer-Verlag, 2006, pp. 88–97.

[10] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," *Cryptographic Hardware and Embedded Systems – Proceedings of CHES 2002*, vol. 2523.

[11] D. H. Habing, "The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits," in *IEEE Transactions on Nuclear Science*, vol. 12, no. 5, 1965.

[12] F. Darracq, T. Beauchene, V. Pouget, H. Lapuyade, D. Lewis, P. Fouillat, and A. Touboul, "Single-event sensitivity of a single SRAM cell," in *Radiation and Its Effects on Components and Systems – Proceedings of RADECS'2001*, 2001, pp. 387–391.

[13] M. Agoyan, J.-M. Dutertre, A.-P. Mirbaha, D. Naccache, A.-L. Ribotta, and A. Tria, "How to flip a bit?" in *On-Line Testing – Proceedings of IOLTS'2010*. IEEE, 2010, pp. 235–239.

[14] M. Bruestle, "SOSSE - simple operating system for smartcard education," *http://www.mbsks.franken.de/sosse/index.html*, 2002.

[15] M. Agoyan, J.-M. Dutertre, A.-P. Mirbaha, D. Naccache, A.-L. Ribotta, and A. Tria, "Single-bit DFA using multiple-byte laser fault injection," in *Technologies for Homeland Security – Proceedings of HST'2010*. IEEE, 2010, pp. 113–119.