Fault Round Modification Analysis of the Advanced Encryption Standard

Jean-Max Dutertre, Amir Pasha Mirbaha, David Naccache, Anne-Lise Ribotta, Assia Tria, Thierry Vaschalde

Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on
http://dx.doi.org/10.1109/HST.2012.6224334

Fault Round Modification Analysis of the Advanced Encryption Standard

Jean-Max Dutertre*, Amir-Pasha Mirbaha*, David Naccache†, Anne-Lise Ribotta*, Assia Tria‡ and Thierry Vaschalde*
*‡Département Systèmes et Architectures Sécurisées (SAS)
*École Nationale Supérieure des Mines de Saint-Étienne (ENSMSE), ‡CEA-LETI, Gardanne, France
{dutertre, mirbaha, ribotta, vaschalde}@emse.fr assia.tria@cea.fr
†Équipe de cryptographie, École normale supérieure (ENS), Paris, France
david.naccache@ens.fr

***Abstract*—This paper describes a new physical analysis technique based on changing the number of the AES rounds. It is an extension of the already known *Round Reduction Analysis* techniques. *Round Modification Analysis* is a specific *algorithm modification* attack. However, the cryptanalysis of the obtained erroneous ciphertexts resorts to the differentiation techniques used by *Differential Fault Analysis*. Faults were induced thanks to a laser in a software AES, either on the round counter itself or on the reference of its total round number, to obtain an increase or a decrease in the number of rounds. We report here successful attacks and their corresponding cryptanalysis.**

I. Introduction

Fault attacks consist in using hardware malfunction to infer secrets from the target's faulty behaviour or outputs. [1] and [2] reported in 1997 the possibility of secret leakage by physical perturbations. [3] presented a differential analysis method to exploit such faults. These active attacks can be performed in different physical manners as reported by [4]. In such attacks, the internal operations of the target integrated circuit (IC) are disturbed to modify behavior or to inject faults into the computations of a cryptographic algorithm. Modifying the behaviour of a device's software refers to *algorithm modification*. This class of active attacks may consist in replacing instructions executed by a microcontroller [5] to circumvent its security features, or in weakening the strength of an encryption algorithm by reducing to one or two the number of its rounds [6]–[8] (*i.e. Round Reduction Analysis* or RRA). This paper proposes an extension of the latter analysis to the advanced encryption standard (AES): the *Round Modification Analysis* (RMA). RMA is based on decreasing or increasing the number of AES rounds or on altering them to retrieve information on the secret key. Both cases were experimentally obtained using laser fault injection. We present in this paper a few instances with their corresponding cryptanalysis. Remarkably, the cryptanalysis of the obtained erroneous ciphertexts resorts to the differentiation techniques used by *Differential Fault Analysis*.

This article is organized as follows. A review of the state-of-the-art of RRA and some remainders on the AES are given in section II. The experimental setup (*i.e.* the targeted IC that embeds a software AES and the laser bench) is described in section III. The principles of the RMA, three significant examples and their corresponding cryptanalysis are reported 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 repeating 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. 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 the first round). As a result, much of the encryption process is skipped and the final ciphertext is the product of few algorithm processes that may easily reveal the key.

Round Reduction Analysis is based on decreasing the number of rounds in an algorithm to ease subsequent cryptanalysis. This method was first presented in [6]: where the authors described a *RR* analysis on AES reducing the total round number from ten to one.

In the following, we first remind the AES' basics before going deeper into the state-of-the-art of *Round Reduction Analysis*.

A. The Advanced Encryption Standard

AES is a Substitution-Permutation Network (*SPN*) block cipher [9]. It processes a 128-bit plaintext and a key of 128, 192 or 256 bits long to produce a 128-bit ciphertext. For the sake of simplicity, we will consider hereafter only the 128-bit AES version: denoted AES or AES-128. The algorithm has two separated processes: One for the `KeyScheduling` 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. A round key is used during the computations of every round. 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). Figure 1 shows the different transformations of the AES algorithm. As the `KeyScheduling` process is of no relevance for this paper, it is not described here.

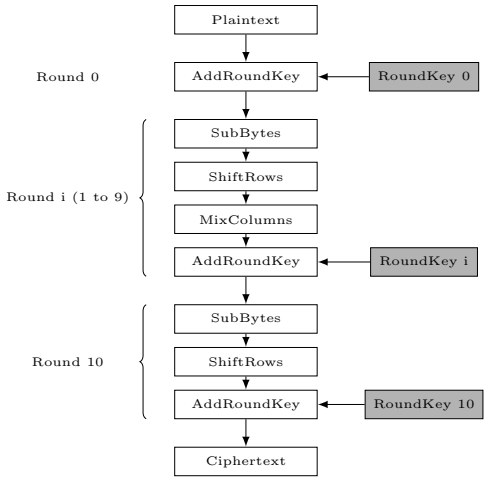

Fig. 1. The AES-128 - General Outline.

To encrypt a plaintext, namely 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 last 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, the algorithm applies a linear transformation to each element and combines it with other values of the same column with a different coefficient of 1, 2 or 3 for each element (`MixColumns` or `MC`) 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 first round, an `ARK` is applied to M and K (*i.e.* `Round 0`). The `MC` transformation is omitted in the last round.

Notation: In the following, we use the "R" prefix plus the round number to refer to the transformations involved in an AES round. 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 $R_0 \dots R_{10}$, represents the rounds of a complete (*i.e.* unmodified) AES. "M_i" represents the AES intermediate state at the beginning of round i. We use $R_{m=j}$ to express that, due to a fault, a round composed of the $\texttt{ARK} \circ \texttt{MC} \circ \texttt{SR} \circ \texttt{SB}$ transformations (where $_m$ stands for middle round) is using an incorrect round key of index j. Note that j may be higher than the number of rounds. $R_{f=j}$ has the same meaning for a round without the `MC` transformation ($_f$ stands for final round).

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

Round Reduction Analysis was first noticed by H. Choukri and M. Tunstall in 2005 [6]. Their work shows a round reduction attack using faults on an AES. There are very few other works exploiting round reduction attacks, the most notable being two other attacks that we present briefly here.

1) Choukri and Tunstall's attack: [6] shows that a transient glitch on the power supply of a microcontroller may change the round counter value of an iterative cipher. If the opponent changes the round counter (hereafter RC) of an AES program at the beginning of algorithm execution to its final value, the ciphertext will be the product of a single round (plus the initial round): R_0-R_m (according to the notation introduced in II-A). Its complexity does not corresponds anymore to the cryptanalysis of a correct execution of the 10 AES rounds. Moreover, they introduced a cryptanalysis technique that make it possible to retrieve the secret key. [6] obtained eq. 1 by xoring two faulty 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) Monnet et al.'s attack: Y. Monnet et al. report in [7] another round reduction attack on two asynchronous cryptoprocessors running DES encryption. The attack was done by laser fault injection. Between the two DES asynchronous cryptoprocessors, the model with countermeasures was found more resistant against attacks during the experiments. However, the attack succeeded on both circuits.

3) Park et al.'s attack: The attack of Park et al. reported in [8] is a laser fault attack on an embedded AES on an Atmega128 microcontroller. The AES implementation is compliant with the algorithm structure proposed in [9].

[8] reported a successful attack that consists in jumping from R_1 to R_{10}. The faulty execution path being $R_0 - R_1 - R_{10}$. Therefore, an additional round is executed in comparison to [6] that includes only $R_0 - R_m$.

The associated cryptanalysis requires data from ten different reduced encryptions. Calculations involve four steps of exhaustive search of 2^{40}, 2^{32}, 2^{24}, and 2^{32} steps respectively. This takes approximately ten hours on a PC.

III. Experimental Set Up

A. Software AES

For our experiments, we used a device communicating with smart card standards built in our laboratory. The board is composed of an 8-bit 0.35 μm RISC microcontroller with an integrated 128 KB flash program memory, 4KB EEPROM and 4KB SRAM. The device runs the *Simple Operating System for Smartcard Education* [10] for simulating the smart card environment.

A software AES has been added to this OS. 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. The encryption process is written in C code. Its structure is given in algorithm 1:

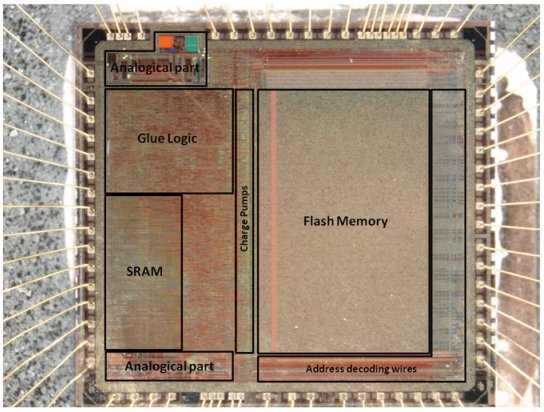

Fig. 2. View of the microcontroller and of its SRAM area.

Algorithm 1 Software AES: algorithm

$C \leftarrow M$
$C \leftarrow C \oplus K$
$RC = 1$
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. Moreover, it is compared to the total round number reference, R_{max}, to end the iterative loop preceding the final round. Note that R_{max} is not a constant number to permit the use of different values (10, 12, or 14 rounds). RC and R_{max} are stored in the circuit SRAM. That's the entry point we used to modify the AES behaviour using a laser. Figure 2 highlights the chip's SRAM area.

B. 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 [11]. 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 [12]. 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, SRAM are subject to bit-flip when exposed to a laser beam [4], [11], [13]. We have reported in [14] 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 section IV.

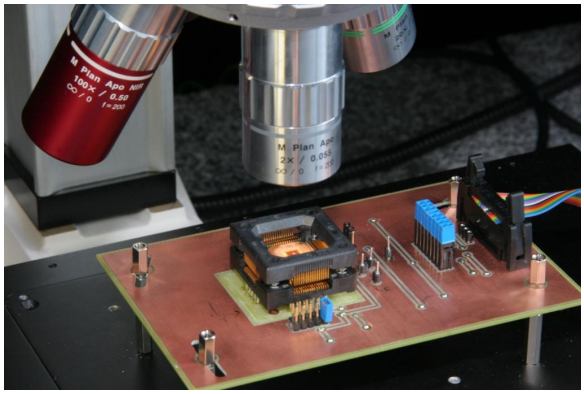

Fig. 3. The target circuit, installed on the laser bench for front side injection.

Our experiments were conducted with green (532nm) or infrared (1064nm) wavelengths, respectively through the front and rear sides of the chip (obviously after a proper decapsulation). The laser beam was about $\varnothing 4\mu$m for a $\simeq 10pJ$ energy per shot (before the lens' attenuation). Figure 3 shows the circuit installed on the laser bench.

A synchronization card provides a jitter of 10ns at the instant of injection. Hence, given the clock period of the device, 280ns, a very precise timing is achieved.

IV. Round Modification Analysis

Previous round reduction attacks were based on the cryptanalysis of an AES reduced to one or two rounds. In our experiments, we surveyed about all the feasible attacks by reducing or increasing the number of executed rounds on our implementation.

Considering the implementation of our AES, several possibilities for single-bit or single-byte laser fault injection are conceivable. To that end, we refer the reader to the results of our previous experiments reported in [14].

As already mentioned in III-A, the round counter is used for counting only the middle rounds (R_m), *i.e.* the rounds between R_1 and R_9. The initial round (R_0) and the final round (R_{10}) are implemented separately as shown in algorithm 1. Hence, even with complete removal of middle rounds, the initial and the final rounds will be still executed.

However, the index of the round key used by the ARK transformation at any round, even at the final round, is given by the round counter value. So, when $1 \leq$ RC ≤ 10, the algorithm xors the temporary ciphertext and K_{RC}. But if RC takes a value greater than 10, the algorithm searches the 16 stored bytes in memory that correspond to an address calculated by the same formula for K_{RC}. So, the value of this 16 bytes block will be used, although it does not match any valid key value. We recall that any round key with an index greater than 10

cannot exist logically for AES-128. Therefore, the temporary ciphertext is xored with a bloc of unknown values.

A. Attack Scenarios

Two scenarios are addressed for changing the number of total rounds in our AES implementation by fault injection. The targets are the round counter value and the total round number reference.

1) Attacks on the round counter value: This attack scenario changes the round counter during AES execution. Therefore it changes the index of the current executing round. Depending on the moment of fault injection, various changes can occur during algorithm execution. We assume in this paper a bit-flip fault model, where the injected fault, e, is xored with RC. Any change in the RC value often leads to a change in the total number of executed rounds, by adding, suppressing or even repetitively executing several rounds:

- 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 and the AES execution will be:

  $$R_0 \ldots R_5\text{-}R_6\text{-}R_5\text{-}R_6\text{-}R_7 \ldots R_{10}$$

  The rounds 5 and 6 will be executed twice and the total number of executed rounds will be incremented to 12.

- If RC$\oplus e >$ RC with RC $< R_{max}-1 \Rightarrow$ *Round reduction*. For example: if RC=4 and e=2 then RC$\oplus e$=6 and the faulty AES execution will be:

  $$R_0 \ldots R_3\text{-}R_6 \ldots R_{10}$$

  Therefore the rounds 4 and 5 will be skipped and the total number of executed rounds will be reduced to 8.

- If RC$\oplus e >$ RC with RC $= R_{max} - 1 \Rightarrow$ *Round alteration*: no change in the total number of rounds, but effects on `AddRoundKey` of the final round and maybe the penultimate round. For instance: if RC=9 and e=2 then RC$\oplus e$=11 and AES execution will be:

  $$R_0 \ldots R_8\text{-}R_{m=11}\text{-}R_{f=12}$$

  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.

2) Attacks on the round number reference: The second attack scenario targets the reference number of the rounds, R_{max}, during the execution of AES. This reference number is accessed only once per round at the beginning of the `while` loop. Faulting R_{max} may induce an increase or a decrease in the total round number. Its alteration can never prevent the execution of the final round. However, depending on the resulting $R_{max} \oplus e$ value, the final round might not use the 10^{th} round key in its `ARK` transformation.

In the following, we illustrate three significant attacks of both scenarios with their associated lightweight cryptanalysis (we did not develop the cryptanalysis for the more complex cases, e.g. $R_0 \ldots R_6\text{-}R_5 \ldots R_{10}$).

B. Realizations

Among various modifications of the AES algorithm obtained thanks to laser fault injection, we have chosen to report three of the most significant: based on a decrease and an increase of the AES round number (exp. 1 and 3 respectively), and on an alteration of the round keys indexes (exp. 2). Remarkably, as opposed to [6] and [8], we found ourselves almost unable to reduce the AES to only one or two rounds. However, we don't claim that it is a general fact. This result is strongly linked to our experimetal bench and to our choice to target RC and R_{max}.

The main difficulty regarding actual realization of RMA is to find out the kind of round modification induced by the laser shot. 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. Thus, we were able to discover any increase or decrease in the round number by comparison with an unfaulted encryption. We also monitored the chip's power consumption for checkout purposes.

1) Experiment 1 using Scenario 1: In this experiment, the RC was targeted in order to induce a decrease in the round number from 10 to 9. The fault was induced at the end of the 8^{th} round just before the increment of RC from 8 to 9. This was achieved with a single bit fault, e=1: RC$\oplus e$ = 8$\oplus$1 = 9. Therefore, the RC was incremented to 10 at the end of 8^{th} round. As a result the sequence of rounds was: R_0-$R_1 \ldots R_8$-R_{10}.

The cryptanalysis of this attack scheme requires at least two pairs of correct and faulty ciphertexts (C^a,D^a), (C^b,D^b). Where:

$$C^a = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (2)$$

$$D^a = \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_{10} \quad (3)$$

Combining eq. 2 and 3 we get:

$$\mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(C^a \oplus K_{10}) = \mathtt{MC}[D^a \oplus K_{10}] \oplus K_9 \quad (4)$$

Then by expressing the relation between C^b and D^b in a similar way and by xoring it to eq. 4 we obtain eq. 5 where K_9 has been removed from:

$$\mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(C^a \oplus K_{10}) \oplus \mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(C^b \oplus K_{10}) = \mathtt{MC}(D^a \oplus D^b) \quad (5)$$

C^a, C^b, D^a and D^b being known values.

Eq. 5 is quite similar to eq. 1 found by Choukri et al. We refer the reader to their paper [6] where the full cryptanalysis is described. Note that, they have suggested the possibility of this attack scheme in their concluding remarks.

2) Experiment 2 using Scenario 1: In this experiment, the RC was targeted in order to induce an error on the index of the round keys used during the penultimate and final rounds. The fault was induced during the 9^{th} round before the ARK transformation. This was achieved with a bytewise fault (different from {01, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F}). As a result the sequence of rounds was: R_0-$R_1 \dots R_8$-R_m-R_f.

This attack changes the index of searched round key during the penultimate and final rounds to invalid values. Therefore, the AES is executed in its original sequence, but the two last ARK transformations are done with incorrect round keys. The obtained faulty ciphertext is equivalent to an encryption with fully faulted K_9 and K_{10}.

Considering a plaintext M^a, the corresponding correct and faulty ciphertexts are given by eq. 6 and 7 respectively:

$$C^a = \mathtt{SR} \circ \mathtt{SB}(M_9^a) \oplus K_{10} = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (6)$$

$$D^a = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K'_x] \oplus K'_y \quad (7)$$

K'_x and K'_y are unknown constant values corresponding to invalid round keys. For the sake of clarity, we express K'_x as the xoring between K_9 and a 16 bytes error matrix E_9: $K'_x = K_9 \oplus E_9$.

Note that, K'_x and K'_y are kept constant for any encryption with other plaintexts provided that the experimental setup ensures the injection of the same fault in the AES calculations. It was the case with our settings.

Hence, eq. 7 is rewritten in eq. 8:

$$D^a = \mathtt{SR} \circ \mathtt{SB}[(M_9^a) \oplus E_9] \oplus K'_y \quad (8)$$

The cryptanalysis of this attack scheme requires three pairs (labelled ^{a}, ^{b}, and ^{c}) of correct and faulty ciphertexts obtained from three different plaintexts. Eq. 9 and 10 are obtained by respectively xoring two faulty and two correct ciphertexts:

$$D^a \oplus D^b = \mathtt{SR} \circ \mathtt{SB}[(M_9^a) \oplus E_9] \oplus \mathtt{SR} \circ \mathtt{SB}[(M_9^b) \oplus E_9] \quad (9)$$

$$C^a \oplus C^b = \mathtt{SR} \circ \mathtt{SB}(M_9^a) \oplus \mathtt{SR} \circ \mathtt{SB}(M_9^b) \quad (10)$$

Eq. 10 is rewritten to express M_9^b:

$$M_9^b = \mathtt{SB}^{-1}(\mathtt{SR}^{-1}(C^a \oplus C^b) \oplus \mathtt{SB}(M_9^a)) \quad (11)$$

Then these expression of M_9^b is replaced in 9 to obtain eq. 12:

$$\mathtt{SR}^{-1}(D^a \oplus D^b) = \mathtt{SB}[(M_9^a) \oplus E_9] \oplus \mathtt{SB}[\mathtt{SB}^{-1}(\mathtt{SR}^{-1}(C^a \oplus C^b) \oplus \mathtt{SB}(M_9^a)) \oplus E_9] \quad (12)$$

A second similar equation (eq. 13) is obtained similarly from (C^a,D^a) and (C^c,D^c):

$$\mathtt{SR}^{-1}(D^a \oplus D^c) = \mathtt{SB}[(M_9^a) \oplus E_9] \oplus \mathtt{SB}[\mathtt{SB}^{-1}(\mathtt{SR}^{-1}(C^a \oplus C^c) \oplus \mathtt{SB}(M_9^a)) \oplus E_9] \quad (13)$$

Hence, eq. 12 and 13 form a system of equations where C^a, C^b, C^c, D^a, D^b, and D^c are known values.

Finally, we perform an exhaustive search over 2^8 possible values for each M_9^a byte and over 2^8 possible values for each corresponding E_9 byte. This exhaustive search leads often to an unique value for each M_9^a byte and another unique value for the corresponding E_9 byte. Then, by using these M_9^a byte values and using equation 14, we find K_{10} byte value. Equation 14 is calculated from the first correct ciphertext equation 6:

$$K_{10} = \mathtt{SR} \circ \mathtt{SB}(M_9^a) \oplus C^a \quad (14)$$

3) Experiment 3 using Scenario 2: In this experiment, the total round number R_{max} was targeted in order to induce an increase in the round number from 10 to 11. The fault was injected before the last comparison of the 9^{th} round. This was achieved with a single bit fault, e=1: $R_{max} \oplus e = 10 \oplus 1 = 11$. As a result the sequence of rounds was: $R_0 \dots R_9$-$R_{m=10}$-$R_{f=11}$.

Considering M_9^a the AES state at the beginning of the 9^{th} round obtained from a plaintext M^a, the corresponding correct and faulty ciphertexts are given by eq. 15 and 16 respectively:

$$C^a = \mathtt{SR} \circ \mathtt{SB}(M_9^a) \oplus K_{10} \quad (15)$$

$$D^a = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_9^a) \oplus K_{10}] \oplus K'_{f=11} \quad (16)$$

Xoring a second faulty ciphertext D^b (obtained from a faulted encryption with another plaintext M^b) with eq. 16 gives eq. 17:

$$D^a \oplus D^b = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_9^a) \oplus K_{10}] \oplus \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_9^b) \oplus K_{10}] \quad (17)$$

We reverse ShiftRows operations in eq. 17, replace corresponding values of C^a and C^b and use the MixColumns' distributivity law to get equation 18:

$$\mathtt{SR}^{-1}(D^a \oplus D^b) = \mathtt{SB}[\mathtt{MC}(C^a) \oplus \mathtt{MC}(K_{10}) \oplus K_{10}] \oplus \mathtt{SB}[\mathtt{MC}(C^b) \oplus \mathtt{MC}(K_{10}) \oplus K_{10}] \quad (18)$$

A second similar equation (eq. 19) obtained from a third pair of correct and faulty ciphertexts (C^c and D^c) is required to ease the cryptanalysis process:

$$\mathtt{SR}^{-1}(D^a \oplus D^c) = \mathtt{SB}[\mathtt{MC}(C^a) \oplus \mathtt{MC}(K_{10}) \oplus K_{10}] \oplus \mathtt{SB}[\mathtt{MC}(C^c) \oplus \mathtt{MC}(K_{10}) \oplus K_{10}] \quad (19)$$

C^a, C^b, C^c, D^a, D^b and D^c being known values.

We replace C^a, C^b, C^c, D^a, D^b and D^c by their corresponding values in equations 18 and 19. Then, we examine

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

Attack	Type	Execution	Text # required for cryptanalysis	Key search average runtime
Choukri et al. [6]	Round Reduction	R_0-R_m	2	$\simeq$ 1 second
Park et al. [8]	Round Reduction	R_0-R_1-R_{10}	10	$\simeq$ 10 hours
Experiment 1	RMA: Round Reduction	$R_0 \dots R_8$-R_{10}	2	$\simeq$ 1 second
Experiment 2	RMA: Round Alteration	$R_0 \dots R_8$-R_m-R_f	3	$\simeq$ 1 second
Experiment 3	RMA: Round Addition	$R_0 \dots R_9$-$R_{m=10}$-$R_{f=11}$	3	$\simeq$ 1 hour and 30 minutes

all the possible K_{10} values and `MixColumns`(K_{10}) on any single column by an exhaustive search in equations 18 and 19.

This exhaustive search through $(2^8)^4$ values for each column of K_{10} leads to a set of 2^8 hypotheses. Repeating these operations for the three other columns creates three further sets of 2^8 column values hypotheses for the subsequent columns.

Then, a second exhaustive search between column hypotheses reveals a unique value for K_{10}. It requires a procedure for verifying of all the column hypotheses on K_{10}. For each combination of 4 columns hypothesis on K_{10}, all the previous round keys must be calculated by inversion of the `KeyScheduling` process. Then, we must encrypt one of the plaintexts M to examine the validity of the current key hypothesis. As soon as we find C as the result of encryption, the key is revealed and exhaustive search is interrupted.

Consequently, an exhaustive search of 2^{34} values (followed by a second search of 2^{31} on average) is required to discover the secret key. In our experiment, both searches take in average less than 90 minutes using a PC running with an Intel Core i5-2410M microprocessor at 2.30GHz.

V. CONCLUSION AND PERSPECTIVES

In this paper we introduced a new analysis technique based on changing the number of the AES rounds using fault injection: the *Round Modification Analysis. Round Reduction Analysis* techniques based on reducing the AES round number to 1 or 2 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 set up incomplete countermeasures. We intend in this article to issue a warning by reporting the following three cases (among several that we have observed): increase and decrease of the total round number as well as alteration of the round keys indexes. Many cryptanalysis techniques exist (sometimes relatively easy to set up) which makes it possible to retrieve the AES key from erroneous outputs of a 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: Key Expansion 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 RMA through RC or R_{max} modification, RMA should still be performed by faulting the program counter of the microcontroller.

Moreover, in our opinion, RMA may be extended to other iterative algorithms like DES (*see* [7]). Besides, the `KeyScheduling` process is also a potential target. Further work has to be done based on these findings in order to propose countermeasures against RMA.

REFERENCES

[1] D. Boneh, R. DeMillo, and R. Lipton, "New threat model breaks crypto codes," *Bellcore Press Release*, 1996.

[2] D. Boneh, R. A. Demillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," *EUROCRYPT'97*, vol. 1233, pp. 37–51, 1997.

[3] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Lecture Notes in Computer Science*, vol. 1294, pp. 513–525, 1997.

[4] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proceedings of the IEEE*, vol. Proceedings of the IEEE 94, no. 2, pp. 370–382, 2006.

[5] 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 (FDTC), 2011 Workshop on*, pp. 105 –114, sept. 2011.

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

[7] 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 (FDTC)*, vol. 4236, pp. 88–97, Springer Berlin / Heidelberg, 2006.

[8] 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, pp. 434–442, 2011.

[9] NIST, "Announcing the Advanced Encryption Standard (AES)." Federal Information Processing Standards Publication, n. 197, Nov. 26, 2001.

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

[11] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," *4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002*, vol. 2523, pp. 2–12, 2002.

[12] D. H. Habing, "The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits," in *Nuclear Science, IEEE Transactions on*, vol. 12, pp. 91 –100, 1965.

[13] 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, 2001. 6th European Conference on*, pp. 387 – 391, sept. 2001.

[14] 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 Symposium (IOLTS), 2010 IEEE 16th International*, pp. 235 –239, 2010.