Differential Analysis of Round-Reduced AES Faulty Ciphertexts
Amir-Pasha Mirbaha, Jean-Max Dutertre, Assia Tria

To cite this version:

HAL Id: emse-01109144
https://hal-emse.ccsd.cnrs.fr/emse-01109144
Submitted on 24 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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
††3541 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 9th 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 M). 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 1st 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}, ..., 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_{max} 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=0}, has the same meaning for a round without the MC transformation (“f” stands for final round). In addition, “Mi” 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 ← M |
| C ← C ⊕ K0 |
| RC ← 1 |
| while (RC < R_{max}) do |
| C ← SB(C) |
| C ← SR(C) |
| C ← RC(C) |
| C ← C ⊕ K_{RC} |
| RC ← RC + 1 |
| end while |
| C ← SB(C) |
| C ← SR(C) |
| C ← C ⊕ 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_{0} and D_{f} (M^{a} and M^{b}, being the corresponding plaintexts):

\[MC^{-1}(D_{0} + D_{f}) = SB(M^{a} + K) \oplus SB(M^{b} + K) \quad (1)\]

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_{1}→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^{10}, 2^{32}, 2^{24}, and 2^{12} 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}→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. Chouki 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 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 $R_{\text{max}}$ 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 $R_{\text{max}}$ 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 $R_{\text{max}}$ 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 $R_{\text{max}}$ during AES execution. $R_{\text{max}}$ is accessed only once per round, at the beginning of the while loop in algorithm 1. Depending on the resulting $R_{\text{max}} \oplus e$ 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 $R_{\text{max}} \oplus e < R_{\text{max}}$ and $RC < R_{\text{max}} - 1 \Rightarrow$ Round reduction or suppression of one or several rounds to max {$R_{\text{max}} \oplus e, RC+1$} rounds. For instance: if $e=8$ is injected, when $RC = 5$, then $R_{\text{max}} \oplus e=2$ and the AES execution will be: $R_0...R_5-R_f=6$. Therefore, as $R_{\text{max}} \oplus e < RC$, the following round after the attack will be executed as the final round and any remaining round will be suppressed.

Case b: If $R_{\text{max}} \oplus e > R_{\text{max}}$ and $RC < R_{\text{max}} \Rightarrow$ Round addition or execution of additional rounds. For instance: if $e=4$ is injected, when $RC < 10$, then $R_{\text{max}} \oplus e=14$ and the AES execution will be: $R_0...R_9-R_m=10-R_\text{m}=11-R_\text{m}=12$.

Faulting $R_{\text{max}}$ when $RC < R_{\text{max}} - 1$ results always in round reduction or addition. A fault in $R_{\text{max}}$ 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: $R_0...R_7-R_5-R_7...R_{10}$. 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: $R_0...R_5-R_6-R_7...R_6-R_7...R_{10}$.

Case b: If $RC \oplus e > RC$ when $RC < R_{\text{max}} - 1 \Rightarrow$ Round reduction. For example: if $RC = 4$ and $e = 2$ (before $ARK$) then $RC \oplus e = 6$ and the faulty AES execution will be: $R_0...R_3-R_6...R_{10}$. 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 $RC = R_{\text{max}} - 1 \Rightarrow$ 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 after $ARK$, then $RC \oplus e = 11$ and AES execution will be: $R_0...R_9-R_m=11-R_f=12$. 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 $R_{\text{max}}$, 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 \( R_{\text{max}} \) 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 \( R_{\text{max}} \), the opponent may reduce the encryption sequence to 2 rounds, i.e.: \( R_0 \rightarrow R_1 \rightarrow R_f = 2 \), [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 \( BC \) during the execution of \( R_1 \). If the fault is injected during \( \text{ARK} \) of \( R_1 \), the encryption sequence becomes: \( R_0 \rightarrow R_1 \rightarrow R_f \), [4] may again be used for the exploitation. Besides, if the fault is injected before \( \text{ARK} \) of \( R_1 \), the encryption sequence becomes \( R_0 \rightarrow R_{\text{max}} \rightarrow R_f \). 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: \( R_0 \rightarrow R_{\text{max}} \rightarrow R_f \) 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 (\( M_a, D_0 \), \( M_b, D_0 \) and \( M_c, D_0 \)). Considering a plaintext \( M_a \), the corresponding faulty ciphertext is given by eq. 3:
\[
D_a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_a \oplus K) \oplus K'_y] \oplus K'_y \tag{3}
\]

In eq. 3, \( K'_y \) and \( K''_y \) are unknown constant values corresponding to invalid round keys, for the rounds \( R_{\text{max}} \) and \( R_f \), respectively. By xorring eq. 3 and the similar equation for \( D_b \), we discard \( K'_y \) values and we obtain eq. 4. We repeat the xorring for eq. 3 and for \( D^n \) equation.
\[
D^n \oplus D^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_a \oplus K) \oplus K'_y] \oplus \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_b \oplus K) \oplus K'_y] \tag{4}
\]

Nevertheless, \( K'_y \) remained in eq. 4 and in corresponding equation for \( D^n \oplus D^c \). The solution for resolving them consists in creating hypotheses on both of \( K \) and \( K'_y \) where \( M_a, M_b, M_c, D^n, D^b, \) 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 \( \text{SR}(K) \). The search results to only 1, 2 or 3 hypotheses for each corresponding column of \( \text{SR}(K) \) and \( K'_y \). 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 \( K(i) \) may be inexistant 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 (<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 \( R_{\text{max}} \) 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:

\[
\begin{align*}
\text{1st encryption:} & \quad R_0 \rightarrow \ldots \rightarrow R_i \rightarrow R_{f=+1} \\
\text{2nd encryption:} & \quad R_0 \rightarrow \ldots \rightarrow R_i \rightarrow R_{f=+1} \rightarrow R_{f=+2}
\end{align*}
\]

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 \( D^a \) and \( D^b \). We show them by eq. 5a and eq. 5b:
\[
\begin{align*}
D^a &= \text{SR} \circ \text{SB}(M^a) \oplus K_{i+1} \tag{5a} \\
D^b &= \text{SR} \circ \text{SB}(M^a+1) \oplus K_{i+2} \tag{5b}
\end{align*}
\]

\( K_{i+1} \) 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:

$$D_1^i + D_1^j = \text{SR} \circ \text{SB}(M_{a}^i) \oplus \text{SR} \circ \text{SB}(M_{b}^j) \quad (6)$$

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

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

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

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

$$D_2^i + D_2^j = \text{SR} \circ \text{SB}(M_{a}^i) \oplus \text{SR} \circ \text{MC}(D_1^i \oplus D_1^j) \oplus M_{a+1}^i \quad (9)$$

where $D_1^i$, $D_1^j$, $D_2^i$ and $D_2^j$ have known values. Hence, we perform an exhaustive search over $2^{3}$ possible values for each $M_{a+1}^i$ byte. This search leads to a unique value for each byte. Then, by using the obtained $M_{a+1}^i$ 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 $M^a$:

$$D_1^i = \text{SR} \circ \text{SB}(M_{a}^i) \oplus K_{y1}' \quad (10a)$$

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

A differential cryptanalysis between eq. 10a and eq. 10b leads to finding $M_{a+1}^i$. 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:

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

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

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

Therefore, two separate differential cryptanayses by using three round-reduced encryptions of $D_1$, $D_2$ and $D_3$ for 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 $M_{a+1}^i$ 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 $M_{a+2}^i$. At the third step, each 2 hypotheses for one byte on $M_{a+1}^i$ creates $2^{4}$ hypotheses for its corresponding column values. Thus, with 2 hypotheses for each of corresponding bytes on $M_{a+2}^i$, $2^{8}$ 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 $M^a$:

$$D_1^i = \text{SR} \circ \text{SB}(M_{a}^i) \oplus K_{y_i}' \quad (13a)$$

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

$K_{y_i}'$ 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 $R_5$, then it performs $R_6$ as the final round, i.e. without MC: $R_0\ldots R_5 R_{f=6}$. 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: $R_0\ldots R_7 R_{f=8}$. The attack exploitation needs at least three pairs of correct and round-reduced ciphertexts: $(C^a, D^a)$, $(C^b, D^b)$ and $(C^c, D^c)$.

$$C^a = \text{SR} \circ \text{SB}(MC \circ \text{SR} \circ \text{SB}(M_{a}^c) \oplus K_9) \oplus K_{10} \quad (14)$$

$$D^a = \text{SR} \circ \text{SB}(M_{a}^c) \oplus K_{y_i}' \quad (15)$$

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

$$\text{MC}(D^a \oplus D^b) = \text{MC} \circ \text{SR} \circ \text{SB}(M_{a}^c) \oplus \text{MC} \circ \text{SR} \circ \text{SB}(M_{a}^b) \quad (16)$$
Eq. 16 can be also expressed by $M_8$ values, as shown in eq. 17:

$$MC(D^a \oplus D^b) = M_8^* \oplus M_8^b$$  \hspace{1cm} (17)

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

$$SR^{-1}(C^a \oplus C^b) = SB(M_8^a) \oplus SB(M_8^b)$$  \hspace{1cm} (18)

By replacing $M_8^a$ and $M_8^b$ from eq. 17 in eq. 18, we obtain eq. 19:

$$SR^{-1}(C^a \oplus C^b) = SB[MC \circ SR \circ SB(M_8^a) \oplus K_9] \oplus SB[MC \circ SR \circ SB(M_8^b) \oplus K_9]$$  \hspace{1cm} (19)

In eq. 19, $C^a$, $C^b$, $D^a$ and $D^b$ have known values. We perform an exhaustive search among $(2^k)^4$ possible values for each $SR(M_8^c)$ 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 $2^k$ 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 laser fault injection. We have already reported ([13], [15]) bytwise 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 $\phi 4 \mu m$ with $\sim 10 p J$ 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 0.1$\mu m$ 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 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 (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]) bytwise 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.

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.
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 $\Delta t$. For a decrease of 8 middle rounds, $\Delta t$ is about 1.48ms. When the encryption skips two rounds, $\Delta t$ 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.

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-Y stage, 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 these 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
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


---

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

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Target</th>
<th>Attack time</th>
<th>Execution(s)</th>
<th>Required key search</th>
<th>Average runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1.a</td>
<td>$R_{max}$</td>
<td>During $R_{0}...R_{1}$, if $e=0x08$, during $R_{1}$, if $e \in (0x0a, 0x0b)$</td>
<td>$R_{0}...R_{1}$</td>
<td>$e=0x02$ &amp; $0x06$</td>
<td>3 ≤ 1 second and 55 minutes</td>
</tr>
<tr>
<td>Experiment 1.b</td>
<td>$R_{max}$</td>
<td>During $R_{0}...R_{1}$ or only $R_{e}$, according to $e$ for $(D_{1})$</td>
<td>$R_{0}...R_{1}$</td>
<td>$e=0x02$ &amp; $0x06$</td>
<td>3 ≤ 1 second and 55 minutes</td>
</tr>
<tr>
<td>Experiment 2.a</td>
<td>$R_{max}$</td>
<td>During $R_{0}...R_{1}$ or only $R_{e}$, according to $e$ for $(D_{2})$</td>
<td>$R_{0}...R_{1}$</td>
<td>$e=0x02$ &amp; $0x06$</td>
<td>3 ≤ 1 second</td>
</tr>
<tr>
<td>Experiment 2.b</td>
<td>$R_{max}$</td>
<td>During $R_{0}...R_{1}$ or only $R_{e}$, according to $e$ for $(D_{3})$</td>
<td>$R_{0}...R_{1}$</td>
<td>$e=0x02$ &amp; $0x06$</td>
<td>3 ≤ 1 second</td>
</tr>
<tr>
<td>Experiment 2.c</td>
<td>$R_{max}$</td>
<td>During $R_{0}...R_{1}$ or only $R_{e}$, according to $e$ for $(D_{4})$</td>
<td>$R_{0}...R_{1}$</td>
<td>$e=0x02$ &amp; $0x06$</td>
<td>3 ≤ 1 second</td>
</tr>
<tr>
<td>Experiment 2.d</td>
<td>$R_{max}$</td>
<td>During $R_{0}...R_{1}$ or only $R_{e}$, according to $e$ for $(D_{5})$</td>
<td>$R_{0}...R_{1}$</td>
<td>$e=0x02$ &amp; $0x06$</td>
<td>3 ≤ 1 second</td>
</tr>
<tr>
<td>Experiment 3</td>
<td>$R_{max}$</td>
<td>During $R_{0}...R_{1}$ or only $R_{e}$, according to $e$ for $(D_{6})$</td>
<td>$R_{0}...R_{1}$</td>
<td>$e=0x02$ &amp; $0x06$</td>
<td>3 ≤ 1 second</td>
</tr>
<tr>
<td>Experiment 4.a</td>
<td>$R_{max}$</td>
<td>During $R_{0}...R_{1}$ or only $R_{e}$, according to $e$ for $(D_{7})$</td>
<td>$R_{0}...R_{1}$</td>
<td>$e=0x02$ &amp; $0x06$</td>
<td>3 ≤ 1 second</td>
</tr>
</tbody>
</table>

**Notes:**

- Experiment 1.a: This attack targets the AES algorithm during the initial key scheduling phase.
- Experiment 1.b: This attack targets the AES algorithm during the key scheduling phase, depending on the value of $e$.
- Experiment 2.a: This attack targets the AES algorithm during different key scheduling phases, with varying conditions.
- Experiment 2.b: This attack targets the AES algorithm during different key scheduling phases, with varying conditions.
- Experiment 2.c: This attack targets the AES algorithm during different key scheduling phases, with varying conditions.
- Experiment 2.d: This attack targets the AES algorithm during different key scheduling phases, with varying conditions.
- Experiment 3: This attack targets the AES algorithm during different key scheduling phases, with varying conditions.
- Experiment 4.a: This attack targets the AES algorithm during different key scheduling phases, with varying conditions.