Piret and Quisquater's DFA on AES Revisited

Christophe Giraud[1] and Adrian Thillard[1,2]

[1] Oberthur Technologies,
4, allée du doyen Georges Brus, 33 600 Pessac, France.
c.giraud@oberthur.com
[2] Université Bordeaux I,
351, cours de la Libération, 33 405 Talence cedex, France.
adrian.thillard@etu.u-bordeaux1.fr

Abstract. At CHES 2003, Piret and Quisquater published a very efficient DFA on AES which has served as a basis for many variants published afterwards. In this paper, we revisit P&Q's DFA on AES and we explain how this attack can be much more efficient than originally claimed. In particular, we show that only 2 (resp. 3) faulty ciphertexts allow an attacker to efficiently recover the key in the case of AES-192 (resp. AES-256). Our attack on AES-256 is the most efficient attack on this key length published so far.

Keywords: DFA, AES.

1 Introduction

Since its publication in 1996, Fault Analysis has become the most efficient way to attack cryptosystems implemented on embedded devices such as smart cards. In October 2000, Rijndael was selected as AES and since then many researchers have studied this algorithm in order to find very efficient differential fault attacks. Amongst the dozen DFA on AES published so far, Piret and Quisquater's attack published at CHES 2003 [5] is now a reference which has been used as a basis for several variants published afterwards. In this paper, we make some remarks about P&Q's method which allow us to improve the efficiency of their attack up to the point that our DFA on AES-256 is the most efficient attack published so far.

The rest of this paper is organized as follows. In Section 2, we briefly recall the AES structure as well as the definition of each of its transformations. In Section 3, we describe P&Q's DFA on AES and we show that 2 newly published DFA on AES-128 [2, 4] can be considered as variations of P&Q's attack. In Section 4 we present a new method to reduce the number of faulty ciphertexts required by P&Q attack in the 192 and 256-bit cases. Indeed, we show that one can recover the AES-192 (resp. AES-256) key value by using only 2 (resp. 3) faulty ciphertexts. Finally, we summarize our results and we compare them with state-of-the-art DFA on AES in Section 5.

2 AES Algorithm

2.1 General Description

The AES is a substitution-permutation block cipher algorithm. Its input and output consist of sequences of 128 bits, while its cipher key is a sequence of 128, 192 or 256 bits, depending on the security level required. For each key length, there will be 10, 12 or 14 rounds (resp. for 128, 192 and 256-bit key), each of them being parameterized by a 128-bit round key provided by the key schedule. At each moment, the temporary result can be represented as a 4 x 4 byte matrix, called the *State*.

2.2 A Round

The structure of each round of AES can be reduced to four basic transformations occurring to the elements of the State. Each round consists in applying successively to the State the *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey* transformations. The first round does the same with an extra *AddRoundKey* at the beginning whereas the last round excludes the *MixColumns* transformation.

SubBytes The *SubByte* transformation is a non-linear byte substitution, which if often implemented by a substitution table. We call *SubBytes* the transformation *SubByte* applied to all the bytes of the State.

ShiftRows The *ShiftRows* transformation shifts cyclically to the left the last three rows of the State over different numbers of bytes : the second row is shifted over one byte, the third (resp. the fourth) is shifted over two bytes (resp. three bytes).

MixColumns The *MixColumn* transformation applies a linear transformation to a column of the State. The column is considered as a polynomial of degree 3 with coefficients in $GF(2^8)$ and multiplied with the polynomial $\{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$ modulo x^4+1. We call *MixColumns* the transformation *MixColumn* applied to each of the 4 columns of the State.

AddRoundKey The *AddRoundKey* transformation consists in xoring each byte of the State with the corresponding byte of the round key.

2.3 Key Schedule

The role of the Key Schedule is to create a set of round keys (one per round). To do so, the AES key is firstly expanded into a linear array of 4-byte columns, this array being then split into the various round keys. For the 128-bit (resp. 192-bit) version, the first 4 (resp. 6) columns contain the AES key. The next

columns are defined recursively by xoring the previous column and the column 4 (resp. 6) positions before. For columns in positions multiple of 4 (resp. 6), a transformation is applied to the previous column before the xoring. This transformation consists of a cyclical shift of one byte to the top followed by a *SubByte* application to the 4 bytes of the column and a xor with a round constant.

For more information about the AES, the interested reader may refer to [1].

3 Description of P&Q's DFA on AES

In [5], Piret and Quisquater describe a very efficient DFA on AES which has served as a basis for many variants published afterwards. In this section, we describe this attack on AES-128 as presented in [5].

3.1 Basic attack

The principle of this attack is to induce a fault resulting in a differential of 1 byte at the input of the last *MixColumns*. By guessing 4 bytes of the last round key, the attacker tests if the corresponding differential at the output of the last *MixColumns* corresponds to a 1-byte differential at its input.

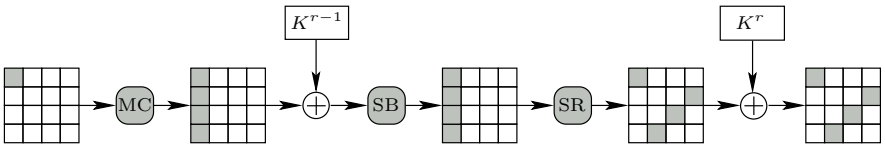

Fig. 1. Propagation of the differential by assuming a byte fault at the input of *MixColumns* of round $r-1$.

To mount this attack, the attacker firstly computes a list $\mathcal{D}$ of possible differences at the output of the *MixColumn* transformation, assuming a 1-byte difference at its input. As the input of the *MixColumn* transformation is composed of 4 bytes, the list $\mathcal{D}$ thus contains $4 * 255$ 4-byte elements. This operation is done only once and can be used for future attacks.

Secondly, the attacker obtains a pair of correct and faulty ciphertext $(C, C^{\nmid})$, $C^{\nmid}$ obtained from a random byte fault injected on the input of the *MixColumns* transformation of round $r-1$. For the sake of simplicity, we assume that the fault is injected on the first column of this State, the extension to other columns being straightforward. In such a case, $C^{\nmid}$ differs from C in bytes at position 0, 7, 10 and 13. The attacker then considers the diagonal of the last round key composed of the four key bytes K_0^r, K_7^r, K_{10}^r, K_{13}^r and for each of the 2^{32} candidates he

computes:

$$\begin{aligned}
\Delta_0 &= SB^{-1}(C_0 \oplus K_0^r) \oplus SB^{-1}(C_0^\ell \oplus K_0^r) \\
\Delta_{13} &= SB^{-1}(C_{13} \oplus K_{13}^r) \oplus SB^{-1}(C_{13}^\ell \oplus K_{13}^r) \\
\Delta_{10} &= SB^{-1}(C_{10} \oplus K_{10}^r) \oplus SB^{-1}(C_{10}^\ell \oplus K_{10}^r) \\
\Delta_7 &= SB^{-1}(C_7 \oplus K_7^r) \oplus SB^{-1}(C_7^\ell \oplus K_7^r)
\end{aligned} \tag{1}$$

The 4-byte result (Δ_0, Δ_{13}, Δ_{10}, Δ_7) is then compared with the 1020 elements contained in the list $\mathcal{D}$. The candidates $(K_0^r, K_{13}^r, K_{10}^r, K_7^r)$ for which a match is found are gathered in a list $\mathcal{L}$.

With one pair (C, C^ℓ), the list $\mathcal{L}$ contains $1036 \approx 2^{10}$ elements on average[3]. By using another pair (C, C^ℓ) with a fault injected on the same column, the corresponding diagonal of the last round key is uniquely determined with a 98% probability.

Therefore the last round key can be recovered by using 8 faulty ciphertexts with faults induced at chosen locations.

3.2 Extended Attack

To extend the attack described above, Piret and Quisquater noticed that if a random byte fault is induced between *MixColumns* of rounds $r-3$ and $r-2$, the corresponding differential at the input of *MixColumns* of round $r-1$ has 4 non-zero bytes, one per column of the State array (cf. Fig. 2).

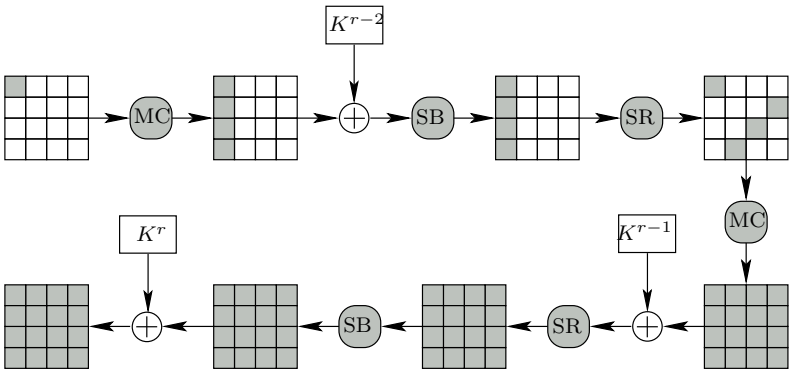

Fig. 2. Propagation of the differential assuming a random byte fault at the input of *MixColumns* of round $r-2$.

Each of these non-zero bytes can thus be exploited by using the method described in Section 3.1 and releases information about a diagonal of the last

[3] In [5, §3.3], it is claimed that $\#\mathcal{L} = 1046$, value obtained experimentally. However, we find that the list $\mathcal{L}$ contains 1036 elements on average by using 1 000 000 simulations. This value matches the theoretical analysis done in [5, §2 Complexity Analysis].

round key. By using such a fault, one pair of correct and faulty ciphertexts allows the attacker to reduce the number of possible values for the last round key to $1036^4 \approx 2^{40}$ from which an exhaustive search to recover a 128-bit AES key can be done. With two pairs of correct and faulty ciphertexts, the last round key is uniquely identified with a 92% probability[4].

3.3 Improvement of Extended Attack

In this section, we explain how the attack presented in Section 3.2 can be improved by using the properties of *ShiftRows* and *MixColumns* transformations.

The set $\mathcal{D}$ has been defined in [5] as a list of possible differences at the output of the *MixColumn* transformation, assuming a 1-byte difference at its input. Therefore $\mathcal{D}$ can be seen as the union of the 4 lists $\mathcal{D}_i$, $i = 0, \cdots, 3$, where $\mathcal{D}_i$ corresponds to the list of possible differences at the output of *MixColumn* assuming a difference on the i^{th} byte of its input.

In [5], the wrong-key distinguisher corresponds to the membership of the 4-column differential at the output of the last *MixColumns* to $\{\mathcal{D}, \mathcal{D}, \mathcal{D}, \mathcal{D}\}$. However, one may note that due to the *ShiftRows* property, this differential belongs to $\mathcal{S} = \{\mathcal{D}_0, \mathcal{D}_3, \mathcal{D}_2, \mathcal{D}_1\} \cup \{\mathcal{D}_3, \mathcal{D}_2, \mathcal{D}_1, \mathcal{D}_0\} \cup \{\mathcal{D}_2, \mathcal{D}_1, \mathcal{D}_0, \mathcal{D}_3\} \cup \{\mathcal{D}_1, \mathcal{D}_0, \mathcal{D}_3, \mathcal{D}_2\}$ which is a much smaller set (cf. Fig. 2). By using this remark, the attacker can improve the wrong-key distinguisher used by Piret and Quisquater by testing if the 4-column differential at the output of the last *MixColumns* belongs to $\mathcal{S}$ or not.

By using this new wrong-key distinguisher and if the position of the disturbed column is known, the number of candidates for each diagonal of the last round key is then reduced to 2^8 by using 1 faulty ciphertext with a byte fault induced between *MixColumns* of round $r-3$ and $r-2$. If the position of the disturbed column is unknown, this number has to be multiplied by 4, i.e. the number of candidates for the last round key is about 2^{34} (to compare with the 2^{40} candidates indicated in the original paper).

From our experiments, an exhaustive search on AES-128 amongst 2^{34} key candidates takes about 8 minutes on average on a 4-core 3.2Ghz Xeon by using a non-optimised C code. Therefore such an attack is practical.

One may note that the attacks presented in [2,4] are exactly the same as the attack described above. Therefore they can be considered as variants of P&Q's attack.

4 Improvement of P&Q's DFA on AES-192 and AES-256

In this section, we firstly make a comment about the original extension of P&Q's attack to the 192 and 256-bit cases before presenting our method which significantly improves P&Q's attack on those key lengths.

[4] In [5, §3.3], it is claimed that this probability is 77%, value obtained experimentally. However, we find that this probability is in fact 92% by using 1 000 000 simulations. We believe that our result is correct since the probability to uniquely identify 4 bytes of the last round key is 98% (cf. Section 3.1) and we have $(98\%)^4 \approx 92\%$.

4.1 A Remark About the Trivial Extension to 192 and 256-bit Cases

In [5], it is written that the extension of the attack to the 192 and 256-bit cases is "trivial" but no more details were given:

"In this paper we will only deal with the 128-bit block, 128-bit key variant, as it is the most widely used. Our attack can be extended trivially to other variants."

Since then, many researchers (including ourselves) have been asking why. In this section, we report the explanation of Piret about this point.

The triviality of the extension of Piret and Quisquater's attack comes from the fact that, since *MixColumns* is linear, one can rewrite the last two rounds of the AES as depicted in Fig. 3.

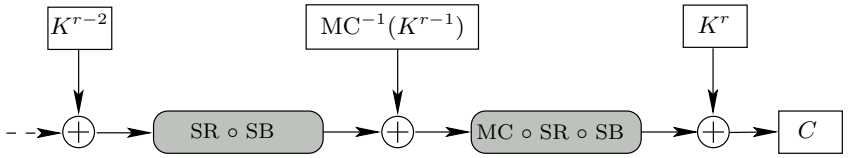

Fig. 3. Another view of the last two rounds of the AES.

By using the above remark, one may note that the AES key can be uniquely recovered by using 4 faulty ciphertexts whatever its length. Indeed, by using 2 faulty ciphertexts with a fault induced between *MixColumns* of round $r-3$ and $r-2$, one can uniquely identify the last round key by using the attack presented in Section 3.2.
Then by using this information, one can decrypt any ciphertext by the last round including the last *MixColumns* (cf. Fig. 3).

In a second step, the attacker obtains two other pairs of correct and faulty ciphertexts $(C, C^\nmid)$ by injecting random byte faults between *MixColumns* of rounds $r-4$ and $r-3$. Therefore for each pair $(C, C^\nmid)$, the attacker computes:

$$\begin{cases} A &= MC^{-1}(SR^{-1}(SB^{-1}(C \oplus K^r))) \\ A^\nmid &= MC^{-1}(SR^{-1}(SB^{-1}(C^\nmid \oplus K^r))) \end{cases} \tag{2}$$

and applies the method described in Section 3.2 with A (resp. $A^\nmid$) instead of C (resp. $C^\nmid$) to recover $MC^{-1}(K^{r-1})$. The penultimate round key K^{r-1} is then obtained by computing the image of $MC^{-1}(K^{r-1})$ through *MixColumns*. Finally the whole AES key is computed from the last two round keys.

To conclude, the original P&Q's DFA on AES can uniquely identify the AES key in the 192 and 256-bit cases by using 4 faulty ciphertexts.

In the next section, we show how this attack can be improved. In particular, we show that the AES key in the 192 and 256-bit cases can be recovered by using only 2 and 3 faulty ciphertexts respectively.

4.2 Improved DFA on AES-256

When analysing the attacks of Sections 3.2, 3.3 and 4.1, we noticed that the faulty ciphertexts with a byte fault induced between *MixColumns* of rounds x and $x+1$ are only used to recover information on the $x+3^{\text{th}}$ round key. Our main idea to improve those results is to notice that these faulty ciphertexts can also be used to recover information on the $x+2^{\text{th}}$ round key by using the method presented in Section 3.1. As shown below, this methodology leads us to significantly improve existing results.

By using 2 Faulty Ciphertexts Let us describe here how one can optimize the number of faulty ciphertexts required by the attack described in Section 4.1. To do so, the attacker re-uses the first 2 faulty ciphertexts to obtain information on the penultimate round key instead of performing the second step of the attack. Indeed, such faulty ciphertexts produce a 1-byte difference at the input of one column of *MixColumns* of round $r-2$. Therefore for each faulty ciphertext, the attacker can apply the attack of Section 3.1 which reduces the number of candidates for a diagonal of $MC^{-1}(K^{r-1})$ to 2^{10}. With a 75% probability, these 2 faulty ciphertexts affect 2 different columns and thus reduce the number of candidates for the AES-256 key to $(2^{10})^2 * (2^{32})^2 = 2^{84}$.

By using 3 Faulty Ciphertexts Now let us assume that we have another faulty ciphertext with a fault induced one round ahead (i.e. between *MixColumns* of rounds $r-4$ and $r-3$). If the location of the disturbed column is known, one can apply the attack of Section 3.3 to reduce the number of candidates for the two diagonals of $MC^{-1}(K^{r-1})$ which have 2^{10} possibilities to 1 and the number of candidates for the two other diagonals to 2^8. This gives a total number of possible candidates for the penultimate round key of $1^2 * (2^8)^2 = 2^{16}$. This number has to be multiplied by 4 if the position of the disturbed column is unknown, i.e. 2^{18}.

4.3 Improved DFA on AES-192

In this section, we use the specificities of AES-192 key scheduling to adapt the attacks of Section 4.2 to the AES-192 case.

By using 2 Faulty Ciphertexts In the 192-bit case, one may note that the knowledge of the last round key allows the attacker to recover the first half of the penultimate round key. In such a case, the attacker knows 2 bytes of each diagonal of K^{r-1}. Therefore each diagonal of $MC^{-1}(K^{r-1})$ has only 2^{16} possible values. By applying the first attack of Section 4.2, two diagonals of $MC^{-1}(K^{r-1})$ will have their number of possible candidates reduced to 2^5. Therefore, this attack reduces the number of candidates for the AES-192 key to $(2^5)^2 * (2^{16})^2 = 2^{42}$ by using only 2 faulty ciphertexts.

From our experiments, an exhaustive search on AES-192 amongst 2^{42} key candidates takes about 1.5 day on average on a 4-core 3.2Ghz Xeon by using a non-optimised C code. Therefore such an attack can be classified as practical.

By using 3 Faulty Ciphertexts The efficiency of the attack described above can be improved by using another faulty ciphertext with a byte fault induced between *MixColumns* of round $r-4$ and $r-3$. As for the second attack of Section 4.2, the attacker can thus apply the method of Section 3.3 to reduce the number of candidates for the two diagonals of $MC^{-1}(K^{r-1})$ which have 2^5 possibilities to 1 and the number of candidates for the two other diagonals to 2^5. Therefore, this extra faulty ciphertext allows the attacker to reduce the number of candidates for $MC^{-1}(K^{r-1})$ to $1^2 * (2^5)^2 = 2^{10}$.

To conclude, one can reduce the number of candidates for the AES-192 key to 2^{10} by using 3 faulty ciphertexts.

5 Summary Tables

Table 1 presents the results of Piret and Quisquater's DFA on AES as originally presented in their paper and Table 2 presents our improved version of their attack.

Key length	Fault Impact	Fault Location	Nb of faulty ciphertexts	Number of key candidates left
128	Random byte	Random between MC of rounds 7 and 8	1	2^{40}
			2	1
192	Random byte	Random between MC of rounds 9 and 10 + Random between MC of rounds 8 and 9	$2+2$	1
256	Random byte	Random between MC of rounds 11 and 12 + Random between MC of rounds 10 and 11	$2+2$	1

Table 1. Original Piret and Quisquater's DFA on AES results.

Key length	Fault Impact	Fault Location	Nb of faulty ciphertexts	Number of key candidates left
128 (§ 3.3)	Random byte	Random between MC of rounds 7 and 8	1	2^{34}
192 (§ 4.3)	Random byte	Random between MC of rounds 9 and 10	2	2^{42}
		Random between MC of rounds 9 and 10 + Random between MC of rounds 8 and 9	$2+1$	2^{10}
256 (§ 4.2)	Random byte	Random between MC of rounds 11 and 12 + Random between MC of rounds 10 and 11	$2+1$	2^{18}

Table 2. Improved Piret and Quisquater's DFA on AES results.

As shown in Table 3, our attack in the 256-bit case is the most efficient one published on this key length in terms of number of faulty ciphertexts and number of key candidates left.

Reference	Fault Impact	Fault Location	Nb of faulty ciphertexts	Number of key candidates left
[6]	Random byte	Random on a chosen temporary variable (input of round 12) in encryption + in decryption	$2+2$	2^{13}
[3]	Random byte	Random between MC of rounds 11 and 12 + Random between MC of rounds 10 and 11	$2+1$	2^{32}
This paper (§ 4.2)	Random byte	Random between MC of rounds 11 and 12 + Random between MC of rounds 10 and 11	$2+1$	2^{18}

Table 3. Characteristics of the three most efficient DFA on AES-256.

6 Conclusion and Further Research

In this paper, we revisit Piret and Quisquater's DFA on AES and we show that the original attack can be widely improved for every key length. Indeed, when using a 128-bit key, we show that only one faulty ciphertext reduces the number of key candidates to 2^{34}. Concerning the AES-192, we present a method which reduces the number of key candidates to 2^{42} (resp. 2^{10}) by using only 2 (resp. 3) faulty ciphertexts. Finally, our extension in the 256-bit case is the most efficient attack published so far. Indeed by using 3 faulty ciphertexts only, one can reduce the number of key candidates to 2^{18}.

Regarding possible improvements, one may note that the third faulty ciphertext used in the 192 and 256 bits cases to obtain information on the $r-1^{\text{th}}$ round key can also be used to obtain information on the $r-2^{\text{th}}$ round key. This could be exploited to significantly reduce the number of key candidates left.

References

1. J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer, 2002.
2. T. Fukunaga and J. Takahashi. Practical Fault Attack on a Cryptographic LSI with ISO/IEC 18033-3 block ciphers. In L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography – FDTC 2009*, pages 84–92. IEEE Computer Society, 2009.
3. C. H. Kim. Differential Fault Analysis against AES-192 and AES-256 with Minimal Faults. In L. Breveglieri, M. Joye, I. Koren, D. Naccache, and I. Verbauwhede, editors, *Fault Diagnosis and Tolerance in Cryptography – FDTC 2010*, pages 3–9. IEEE Computer Society, 2010.

4. D. Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. In B. Preneel, editor, *AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 421–434. Springer, 2009.
5. G. Piret and J.-J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In C. Walter, Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.
6. J. Takahashi and T. Fukunaga. Differential Fault Analysis on AES with 192 and 256-Bit Keys. Cryptology ePrint Archive, Report 2010/023, 2010. `http://eprint.iacr.org/`. English version of the publication in the Japanese domestic symposium, SCIS 2010.