Injection of transient faults using electromagnetic pulses Practical results on a cryptographic system

A. Dehbaoui[1], J.M. Dutertre[2], B. Robisson[1],
P. Orsatelli[3], P. Maurine[4], and A. Tria[1]

[1] CEA-LETI, SESAM Laboratory, Centre Microélectronique de Provence. 880 Route de Mimet, 13541 Gardanne, France
amine.dehbaoui@cea.fr
[2] ENSMSE, SESAM Laboratory, Centre Microélectronique de Provence. 880 Route de Mimet, 13541 Gardanne, France
[3] Amesys, 1030 Avenue Guillibert De La Lauziere 13794 Aix En Provence, France
[4] LIRMM, 161 rue Ada. 34392 Montpellier, France

Abstract. This article considers the use of magnetic pulses to inject transient faults into the calculations of a RISC micro-controller running the AES algorithm. A magnetic coil is used to generate the pulses. It induces computational faults without any physical contact with the device. The injected faults are proved to be constant (i.e. data independent) under certain experimental conditions. This behaviour, combined with the ability to choose the faulted bytes thanks to timing accuracy in the fault injection process, makes it possible to implement most of the state-of-the-art fault attack schemes.

1 Introduction

Since the electronic devices that implement cryptography (such as "smart cards") are key components to secure communications, they are subjected to 'attacks'. Among them, *fault attacks* are considered as very powerful. They consist first in modifying the behavior of the chip with dedicated experimental setups and then recovering the secret information by using cryptanalysis techniques such as Differential Fault Analysis [5, 15, 10, 18] (or DFA), safe-error [24], fault sensibility analysis [12], collision [6], round reduction [8], etc.

Various experimental setups are commonly used to modify the behavior of the chip. Under-powering a device or over-clocking it [20, 4] leads to setup time violations resulting in the injection of errors. Transient deviations under nominal values of the power supply or of the clock period (a.k.a. power or clock glitches) [11, 2, 12], also offer similar fault injection abilities, besides providing higher timing accuracy. Even a rise in the temperature of operation may be used to inject faults [11]. Another important means of fault injection is the use of optical radiations: intense white light (e.g. from a flash bulb) or a laser beam [21]. The latter is widely used while assessing the security of cryptographic systems against fault attacks [23] for certification purposes. It may offer the ability

to inject fault affecting a byte or even a single bit of sensitive data [1]. At last, a several scientific articles in the field of hardware security refer to the use of electromagnetic (EM) field in order to induce faults into cryptographic systems.

In this article, an experimental setup, dedicated to such EM fault injection is described. The effect of such injection on a not protected implementation of a cryptographic algorithm is proposed. The results obtained lead to a fault model which appears to be different from those classically used in the literature or from those obtained with other injection methods.

This article is organized as follows. A short review of the state-of-the-art EM active attacks is given in section 2. We describe our experimental setup and provide a short review of the effect of the magnetic pulses in section 4. Whereas the section 5 reports the results of the implementation of the method described in [15] and the section 6 reports the nature of the injected faults. As a conclusion, section 7 summarizes our findings and draws some prospects.

2 EM Active Attacks : State-of-the-Art

As already mentioned in the introduction, the EM medium may be used to conduct active attacks. Two kinds of near-field EM perturbations are usually considered: transient pulses and harmonic emissions.

Regarding transient pulses, two articles report actual results of successful fault injection. J.-J. Quisquater and D. Samyde [17] in 2002 described the use of an active probe to apply an intense and transient magnetic field on a microprocessor. This results in faulting RAM and EPROM memory cells. Moreover, faults on the device's bus were also obtained. More recently, in 2007, J.-M. Schmidt and M. Hutter reported the use of a spark generator to fault a CRT-based RSA algorithm running on an 8-bit micro-controller [19]. The injected fault leads to a successful attack as it permits them to factorize the RSA modulus.

Concerning harmonic emissions, Alaeldine et al. studied the *electromagnetic compatibility* (or EMC) of integrated circuits (or IC) to near-field injection with frequencies up to 1 GHz [3]. They investigated the effects of both electric and magnetic fields along the x-, y-, and z-axes. The immunity criterion they used was a deviation of the amplitude or a jitter of the output signals over 20% and 10% respectively. Their test circuits were found sensitive to both magnetic and, to a greater extent, electric fields. Recently, Poucheret et al. [16] considered the effect of an 1 GHz electric field applied to an IC with an embedded ring oscillator (or RO). The main component of that electric field was the transverse one (i.e. parallel to the surface of the chip). The perturbation impacted the output frequency of the RO. Monitoring the effect of that perturbation enabled them to draw a cartography of the sensitive areas of the chip. A cross examination between the layout of the device and the cartography demonstrates that the coupling between the injection probe and the circuit lies mainly in the power-ground network (PGN). Besides, this kind of direct power injection through the PGN permits them to increase the output frequency by up to 50%. This result

reinforces the threats against the security of secure devices using RO-based true random pattern generator as a source of entropy [13].

In a complementary way with this previous works, we plan

1. to focus the fault injection on a small part of the targeted device,
2. to focus a very precise instant,
3. to set the energy of the fault injection.

The first and the third features are necessary, for example, to override countermeasures that detect glitches on the power supply or to avoid disturbing the circuit in its entirety (and thus to reduce the risk of crash or system denial). All the three features are also important to impact only a particular internal computation of the cryptographic algorithm and thus to be able to apply a as wide as possible number of cryptanalysis techniques.

Those features lead us to mount an electromagnetic injection bench able to generate localized short transient pulses with tunable energy.

3 Experimental setup

In this section, the electromagnetic injection bench and the circuit under attack are described.

3.1 Magnetic pulse injection bench

The fault injection bench is built of: a control PC, the targeted device (i.e. the victim), a motorized stage, a smart card reader, a pulse generator, and a magnetic probe (see Figure 1). The target (described in section 3.2) is fixed on the x-y-z motorized stage. Every element of the bench is controlled by the control PC, and the communication with the target is established through the smart card reader.

The pulse generator is used to deliver voltage pulses (with low jitter) to the magnetic coil. It has a constant rise and fall transition time of 5ns. The amplitude range (respectively the width) of the pulses extends from 1V to 100V (respectively from 10ns to 100ns). We use a magnetic probe (composed of 10 turns with a diameter of 1mm Figure 2) in order to disturb only a small part of the targeted device. This spatial accuracy is possible thanks to the directive magnetic flux in the near field domain. This magnetic probe incorporates a ferromagnetic core, which acts as a flux concentrator. The resulting magnetic flux is then enhanced (according to the value of the material's permeability μ_r) by comparison with an air coil [22]. However, this enhancement is obtained at the expense of sensor linearity. In our case, this parameter is not impeding.

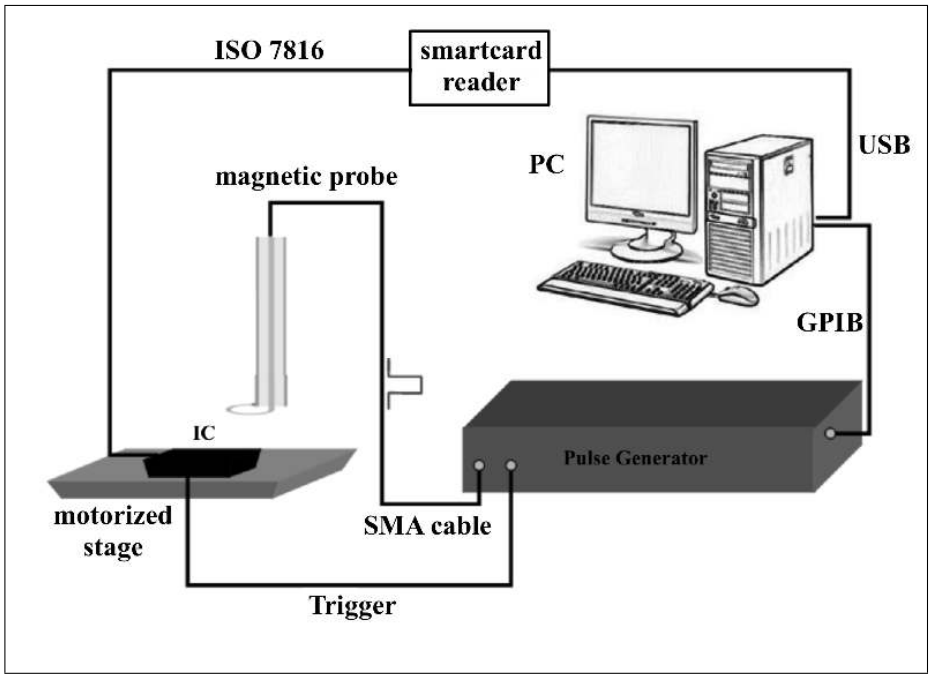

Fig. 1. EM pulse injection bench.

3.2 Target

We use a smart card emulation board built in the laboratory. It is composed of an 8-bit 0.35μm AVR micro-controller with integrated $128KB$ flash program memory, 4KB EEPROM and 4KB SRAM.
This none secure micro-controller has an operating voltage in the range $4.5-5.5V$ and runs at 3.59 MHz frequency. A smart-card-like OS (called SOSSE [7]) is used for communication purposes. A software implementation of the AES encryption algorithm [14] using a key size of 128 bits (hereafter the AES-128) is embedded. The AES-128 ciphers a 128-bit long data (called *plaintext* or *input*) by using a 128-bit long key to obtain a 128-bit long data (called *ciphertext* or *output*). The ciphering consists first in transforming the input data into a two-dimensional array of bytes, called the "State", as illustrated in Figure 3. This array, denoted by the symbol s, is composed of four rows and four lines. Each individual byte $s_{r,c}$ has two indices, with its row number r in the range $0 \leq r < 4$ and its column number c in the range $0 \leq c < 4$.

Next, after a preliminary XOR between the input and the key, the AES-128 executes 10 times a function (called *round*) which operates on the "State". The operations used during these rounds (which are identical except the last one) are the following:

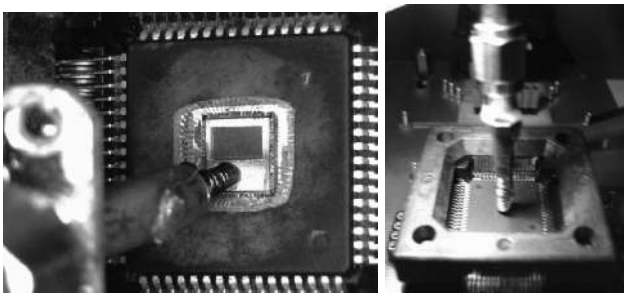

Fig. 2. Magnetic injection probe over an open device (left) and an untampered device (right).

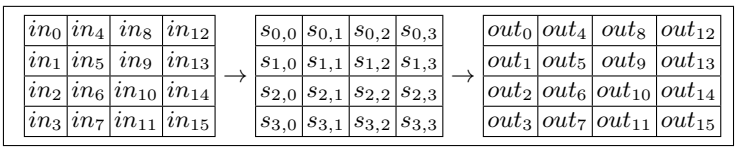

Fig. 3. AES Notations

- `SubBytes` is a non-linear transformation working independently on individual bytes of the State. The state resulting of this operation at round i is noted $round[i].s\_box$
- `ShiftRows` is a rotation operation on each row of the State. The result of this operation at round i is noted $round[i].s\_row$
- `MixColumns` is a linear matrix multiplication working on each column of the State. The result of this operation at round i is noted $round[i].m\_col$
- `AddRoundKey` is a byte-wise XOR between the State and a value computed from the key (according to a transformation called *Key Expansion*). The result of this operation is the start of the next round noted $Round[i+1].start$.

At the end of the ciphering operations, the "State" is copied to the output as depicted in Figure 3.

4 Preliminary experiments

In this section, two experiments are conducted on the target described in 3.2. During both of these experiments, the probe was located at a fixed position above the surface of the circuit (500 μm). This corresponds to an area containing the Arithmetic and Logic Unit (ALU) and the SRAM.

4.1 Experiment 1

First, we investigated the relationship between the parameters of the voltage pulses delivered by the pulse generator (amplitude and width) and their effect

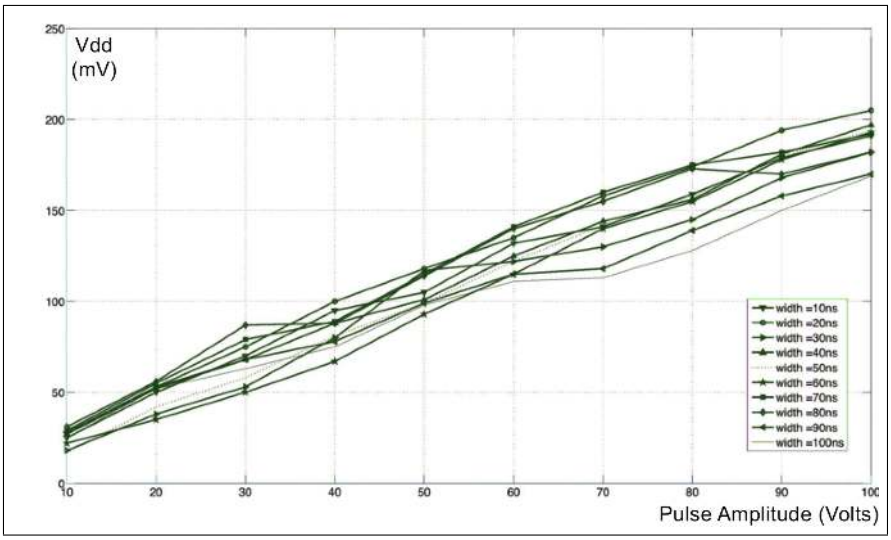

Fig. 4. Amplitude of the measured perturbation versus the amplitude of the voltage transient issued from the pulse generator(given for pulse width ranging from 10ns to 100ns).

on the target. Note that both the rise and fall times of the pulses were equal to 5ns. We measured the induced transient voltage between the power supply and ground pins of the unpowered chip. The pulse duration and amplitude were set to values ranging from 10ns to 100ns and from 1V to 100V respectively. By doing so, we intended to identify the maximum coupling between the injection probe and the victim without recording the noise caused by the internal operations of the chip. Figure 4 reports the maximal amplitude of the induced voltage as a function the pulse voltage amplitude.

The maximum voltage of the perturbation increases linearly with the pulse amplitude. At 100V, a perturbation between 170mV and 205mV is obtained, depending on the pulse width. The maximum coupling is obtained for the smallest durations. The explanation lies in the closeness of the rise and fall edges of the pulses (the coil acts as a differentiator). Figure 5 presents the shape of the induced perturbation resulting from a pulse with a 20ns width and a 50V amplitude.

These voltage variations may seem quite small to be able to induce faults into the device computations. However, because these measures were done outside the chip package, a large part of the perturbation is filtered out.

4.2 Experiment 2

Second, we targeted the execution of the last round of the AES (i.e. the 10th round for the AES-128). We intended to identify the type of faults induced by the EM pulses. Figure 6 shows the power consumption of the target during the

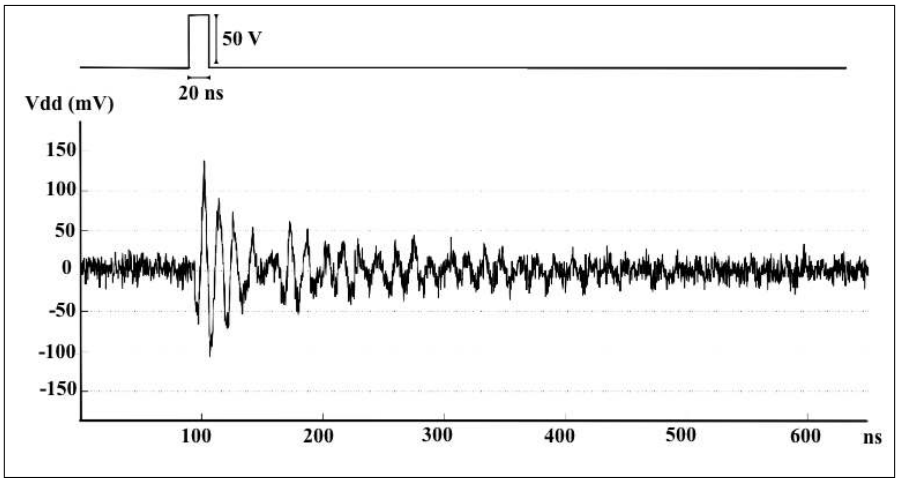

Fig. 5. Pertubation induced by a magnetic pulse on the targeted IC.

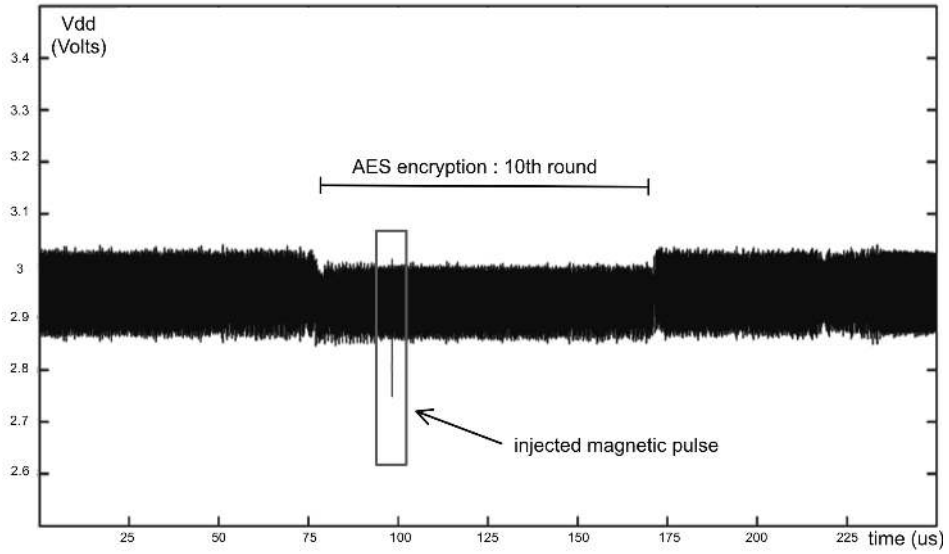

Fig. 6. Power consumption trace during magnetic fault injection.

magnetic fault injection. A pulse of 20ns width and 50V amplitude was injected. A spike of about 150 mV was obtained.

In this experiment, the trigger signal is activated at the beginning of the 10th round for ease of synchronization. We swept the instant of the EM pulse injection from this beginning to the end of the 10th round (90μs) by steps of 10ns. For each of these steps, a ciphering was executed with and without fault

injection. The results of these two computations were compared and the faulted byte determined. Figure 7 indicates the timing of the EM perturbations resulting in faulting every bytes of the AES state. As reported, we were able (a) to create a fault on only one byte of the computation (thanks to the 8-bit architecture) and (b) to select the faulty byte by choosing the right instant of injection.

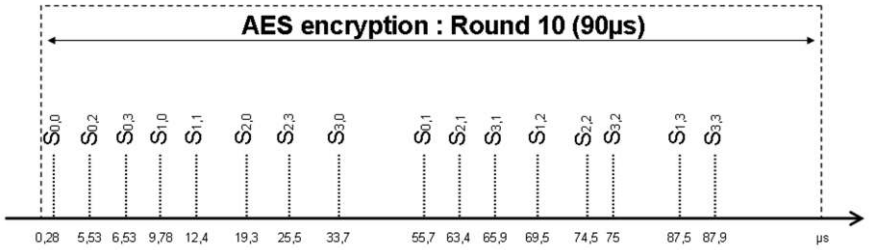

Fig. 7. Timing map of the obtained erroneous bytes.

Then, we tested the correctness of this ability to inject byte-wise faults at every byte location of the AES state during the 9th round (see section 5).

5 Attack

In this section, we describe the attack path that we have chosen and the experiments that have been done in order to validate this attack path on the target described in 3.2.

5.1 Chosen DFA scheme

The experiments described in section 4.2 show that our injection method is:

- able to create a fault on only one byte of the computation,
- able to select the faulty byte by choosing the right instant of injection.

These two properties seems perfectly adequate to mount the first attack described in [15] (further called the P&Q method). According to this scheme, the attacker has to fault one byte of the AES state between the start of the 9th round and the `MixColumns` transformation. Due to the diffusion properties of the AES, a ciphertext that contain 4 erroneous bytes is then outputted. These 4 errors leak information on 4 bytes of the 10th round key. Depending on the location of the faulted byte inside the AES state of the 9th round, 4 different error patterns are encountered:

- Error pattern "0" (hereafter denoted p_0) is obtained for a byte wise fault injected among bytes $s_{0,0}$, $s_{1,0}$, $s_{2,0}$, or $s_{3,0}$ either on state "round[9].start" or on state "round[9].m_col" resulting in faulting bytes $s_{0,0}$, $s_{3,1}$,$s_{2,2}$, $s_{1,3}$ of the ciphertext (i.e. the error pattern),

– similarly error patterns p_1, p_2, p_3 are depicted on figure 8.

Every pattern also indicates the location of the information leakage among the bytes of the 10th round key. For every pattern, by applying the P&Q method on two pairs of correct/faulty ciphertexts, 4 bytes of the 10th round key are retrieved with a success rate of 97%. As a consequence, to discover the whole key the attack had to be completed for the 4 patterns. It was achieved successfully by changing the fault injection time (and consequently the location of the faulted bytes in the AES state). Note that a more efficient implementation of this attack consist in inducing the fault before the `MixColumns` transformation of the 8th round.

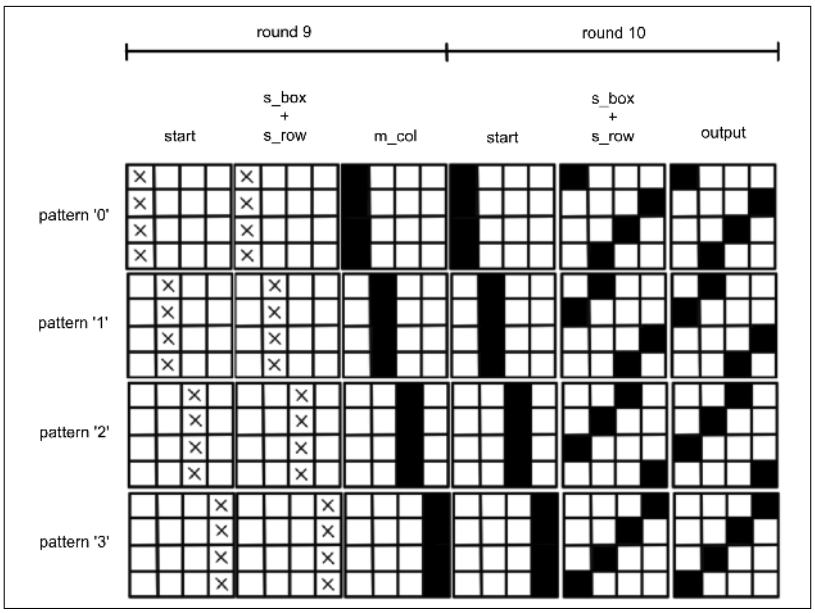

Fig. 8. Fault propagation depending on the location of the fault on the state "round[9].start".

Besides, a given error pattern may be produced for a byte wise fault induced either in the states "start", or "s_box", and "s_row" of the 9th round.

5.2 DFA scheme in practice

As described above, the P&Q method requires a 1-byte fault to be injected before the "MixColumns" transformation of the AES-128 9th round. Note that a trigger signal is activated during the 9th round for ease of synchronization. An error pattern among the 4 defined in 5.1 is selected. Let's consider, for example, the pattern p_0. The instant of the EM pulse injection is swept from the beginning

of the 9th round over a range of 200μs by steps of 10ns. For each of these steps, a ciphering is executed with and without fault injection. The results of these two encryptions are compared and the error pattern is determined. If the retrieved pattern is not p_0, the injection time is incremented to the next time step. Whereas, if it corresponds to pattern p_0, the time step is recorded: noted $F[p_0].first$. The P&Q method is also applied to recover the corresponding bytes of the 10th round key. The same experiment is performed for the three other error patterns. The gathered data make it possible to recover the full 10th round key. The algorithm which describes more precisely the attack is reported in Annex (Algorithm 1).

Algorithm 1 DFA scheme in practice

Require: Input K : Fixed unknown key
Require: Input S : Time steps (vector of size $M = 9 * 10^3$)
Require: Input P : Types of errors (vector of size 4 and $P = [p_0, p_1, p_2, p_3]$
Ensure: Output: K_r retrieved value of the key K
Ensure: Output F : Statistics related with the 4 types of errors
Ensure: With $F[p_i].first$, the first time step when an error of type p_i is detected (scalar)
  for All type $p_i \in P$ of error **do**
    Set $s = 0$
    Set Stop=FALSE
    while While Stop==FALSE **do**
      Select a text t at random
      Compute the associated ciphertext $c = AES(t, K)$ without fault injection
      Inject a pulse at step $S[s]$ during the ciphering of t with K. Let c' being the result of this computation.
      if If $c \oplus c'$ is an error of type p_i **then**
        Set Stop=TRUE
        Set $F[p_i].first = S[s]$
        Perform the attack described in [15] to retrieve 4 bytes of the key K_r by injecting fault at time step $S[s]$ during the ciphering of approximately 2 plain texts chosen randomly
      end if
      Increment s
    end while
  end for

6 Towards a fault model

In this section, we describe a method to compute some statistics about the faults induced by the EM pulses. Then a fault model is proposed.

6.1 Characterization method

We intended to study the effect of the handle data on the value of the injected fault. The experiment described in this section was conducted for 4 different injection times corresponding to the 4 error patterns. For the sake of clarity, we will consider in the following the error pattern p_0. The magnetic pulse was induced at the corresponding time step $F[P_0].first$ for 1000 encryptions made with random plaintexts (however the key was kept constant). Then statistics were built. We determined the percentage of erroneous ciphertexts obtained during these encryptions. Moreover, since we knew the key we were able to run backward the AES encryption steps from the erroneous ciphertext to retrieve the byte value of the injected faults. However, from that analysis a doubt remains on the exact step of the fault injection among the transformations of the AES. We have not investigated further. The fault value we consider is calculated at the step "start". The algorithm which describes more precisely the characterization method is reported in Annex (Algorithm 2).

6.2 Results

The results obtained by the experiments described in the algorithm 2 are summarized in the Table 9. The results show that the proposed EM injection:

- enables to create a fault on only one byte of the state during round 9,
- enables to fault the 16 bytes of the state during round 9 independently (by selecting an adequate injection time step),
- induces a fault that is independent of the value of the faulted byte.

Fig. 9. Results of the fault characterization step

Error pattern (p_i)	First time when an error of type p_i occurs ($F[p_i].first$)	Occurrence ($F[p_i].occurence$)	Most frequent fault values (occurrence %, location) ($[p_i].faults$)
p_0	$11,7\mu s$	100%	CA_H (100%, $s_{1,1}$)
p_1	$9,81\mu s$	100%	EF_H (100%, $s_{0,1}$)
p_2	$27,5\mu s$	100%	22_H (100%, $s_{1,3}$)
p_3	$25,5\mu s$	100%	22_H (100%, $s_{0,3}$)

6.3 Fault model

From the results reported above, it appears that a magnetic pulse injected during the 9th round is able to force a byte of the AES state to a constant value (i.e. independent from the plaintext). However, this value differs from one byte to the

Algorithm 2 Characterization method

Require: Input K : Fixed known key
Require: Input S : Time steps (vector of size $M = 20 * 10^3$)
Require: Input P : Pattern of errors (vector of size 4 and $P = [p_0, p_1, p_2, p_3]$
Require: Input N : Number of tests (scalar)
Ensure: Output F : Statistics related with the 4 pattern of errors
Require: With $F[p_i].first$, the first time step when an error of type p_i is detected (scalar)
Ensure: With $F[p_i].occurrence$, the number of errors of type p_i which occur at time step $F[p_i].first$ (scalar)
Ensure: With $F[p_i].faults$, the faults generated at time step $F[p_i].first$ (vector)
  for All pattern $p_i \in P$ of error **do**
    Set $F[p_i].occurrence = 0$
    for $m = 1$ to N **do**
      Select a text t at random
      Compute the associated ciphertext $c = AES(t, K)$ without fault injection
      Inject a pulse at step $F[p_i].first$ during the ciphering of t with K. Let c' being the result of this computation.
      if If $c \oplus c'$ is an error of type p_i **then**
        Increment $F[p_i].occurrence$
      end if
      Compute offline all the intermediate variables of $AES^{-1}(c, K)$. Let $Round[2].isbox(c, k)$ be the value of the state "isbox" during the second round of the deciphering process(equivalent of the $Round[9].start(t, k)$ in the ciphering process)
      Compute offline all the intermediate variables of $AES^{-1}(c', K)$. Let $Round[2].isbox(c', k)$ the value of the state "isbox" during the second round of the deciphering process.
      Store the "reconstructed" value of the fault in $F[p_i].faults[m] = Round[2].isbox(c, k) \oplus Round[2].isbox(c', k)$
    end for
  end for

other. The value depends also on the injection time, as the way to fault different bytes is to sweep the instant of injection. In other words, the function which associates the value x of a byte and the faulted value when the EM injection is performed at time t_i is the function $f_{t_i} : x \leftarrow D$ with $D \in 00_H, \ldots, FF_H$ for all $x \in 00_H, \ldots, FF_H$. Such a type of fault could be considered as a combination of set and reset faults which modify all the bits of the byte. Note that the value of D may be known by the attacker if, like in the side-channel attack called "template", he is able to performed previous measurements on the same circuit than the target but with the access on the value of the key.

6.4 Other possible attack schemes

In the previous Sections, we have shown that our EM fault attack setup allow us to inject a fault on one byte, with a high accuracy, and we have experimentally

verified that, assuming a precise temporal trigger, the probability of the fault repeatability is close to 100%.

Template EM attack Let's consider that the attacker is able to perform measurements on a circuit and on a software implementation of the AES which are strictly identical to the target but with a known key. During a first step (called *characterization step*), the attacker scans the round 10 in the time domain to detect the instant of the EM injection which modifies a byte of the output (exactly as it has been done in 4.2). Because the value of the key is known, he is able to compute a fault on the state "round[10].s_row" which could have produced such an output. By doing the same for the all the bytes, he's able to associate each byte j of the output with the value of the fault (equivalent to a fault created on the state "round[10].s_row" and noted D_i) and with an instant of injection (noted t_j).
During the attack step, the attacker performs the fault injection at one of those different instants t_j. He verifies that the byte j is impacted at the output. If it is the case, the value of this faulted byte out_j is stored. The byte j of the round key 10 is computed by using the formula $D_j \oplus out_j$. He performs the same for the others bytes.

Combined Attack to defeat protected implementation A recent work of Roche et al. [18] propose a combined fault and side-channel attack, allowing to defeat protected implementations of AES. More precisely, the attack is able to defeat a masked and fault protected implementation, by injecting a fault during the penultimate key schedule, and by measuring the leakage of the faulty ciphertext. Nevertheless, they need to be able to inject a fault with a good repeatability, in other ways it means that the effect of the fault on the faulted byte can be rewritten as a XOR with a constant value. Considering this requirement, our experimental results seem showing that the good repeatability of the fault we observe could allow us to practically perform this attack.

Meet-in-the-Middle and impossible DFA on AES Other fault attack schemes seem to be practically applied in the same way. For instance, the recently published work of Derbez et al. [9] propose several new Differential Fault Attacks on AES, allowing to retrieve the key by injecting random faults on one byte between the 6th and 7th MixColumns (in the case of AES128). Our practical results show that the fault model we get could allow us to perform these attacks. Considering the constant fault model, the attack proposed needs only 5 faults and its complexity in memory is reduced to 2^{24} while other fault models require either 1000 or 45 faults depending on the fault model and recover the secret key with a time and memory complexity around 2^{40}.

7 Conclusion

In this article, a fault injection technique based on the generation of an electromagnetic field close to a circuit, has been presented. This technique enables to fault every byte of the AES state on a non protected software implementation of an AES, running on an 8-bit micro-controller. We have shown that this feature enables to mount a classical DFA scheme on our target but also should be suitable to implement any DFA method that requires the injection of the byte-wise fault. Moreover, a careful review of the nature of the faults created on micro-controller showed that they were constant (i.e. data independent) for each erroneous byte considered individually. Yet, these results were found to be reproducible with a success rate close to a hundred percent. Thus, it may be used to implement fault attacks that are based on a constant fault model. At last, in comparison with other optical fault injection means, the proposed EM injection experiments were carried out both on decapsulated (i.e. frontside opening of the package) and on untampered ICs without any noticeable differences in their responses to the perturbations. We now plan to create fault with the EM injection technique described in this paper on an hardware implementation of cryptographic algorithms and to apply this technique against protected implementations.

References

1. Agoyan, M., Dutertre, J.M., Mirbaha, A.P., Naccache, D., Ribotta, A.L., Tria, A.: How to flip a bit? In: On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International. pp. 235 – 239 (2010)
2. Agoyan, M., Dutertre, J.M., Naccache, D., Robisson, B., Tria, A.: When clocks fail: On critical paths and clock faults. In: CARDIS'10. pp. 182–193 (2010)
3. Alaeldine, A., Ordas, T., Perdriau, R., Maurine, P., Ramdani, M., Torres, L., Drissi, M.: Assessment of the Immunity of Unshielded Multi-Core Integrated Circuits to Near-Field Injection. In: EMC Zurich 2009 (2009)
4. Barenghi, A., Bertoni, G., Parrinello, E., Pelosi, G.: Low voltage fault attacks on the rsa cryptosystem. In: FDTC. pp. 23–31 (2009)
5. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: CRYPTO. pp. 513–525 (1997)
6. Blömer, J., Krummel, V.: Fault based collision attacks on aes. In: FDTC. pp. 106–120 (2006)
7. Bruestle, M.: sosse - simple operating system for smartcard education (2002)
8. Choukri, H., Tunstall, M.: Round reduction using faults. FDTC pp. 13–24 (2005)
9. Derbez, P., Fouque, P.A., Leresteux, D.: Meet-in-the-middle and impossible differential fault analysis on aes. In: CHES. pp. 274–291 (2011)
10. Giraud, C.: Dfa on aes. Advanced Encryption Standard - AES, 4TH International Conference, AES 2004 3373, 27–41 (2003)
11. Hamid, H.B.E., Choukri, H., Tunstall, M., Naccache, D., Whelan, C.: The sorcerer's apprentice guide to fault attacks 94 (2004)

12. Li, Y., Sakiyama, K., Gomisawa, S., Fukunaga, T., Takahashi, J., Ohta, K.: Fault sensitivity analysis. In: Mangard, S., Standaert, F.X. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010, Lecture Notes in Computer Science, vol. 6225, pp. 320–334. Springer Berlin / Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-15031-9_22, 10.1007/978-3-642-15031-9_22
13. Markettos, A.T., Moore, S.W.: The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators, pp. 317–331 (2009)
14. NIST: Announcing the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication, n. 197 (Nov 26, 2001)
15. Piret, G., Quisquater, J.J.: A differential fault attack technique against spn structures, with application to the aes and khazad. In: Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings. pp. 77–88 (2003)
16. Poucheret, F., Tobich, K., Lisart, M., Robisson, B., Chusseau, L., Maurine, P.: Local and direct em injection of power into cmos integrated circuits. In: FDTC'11 (2011)
17. Quisquater, J.J., Samyde, D.: Eddy current for Magnetic Analysis with Active Sensor. In: Esmart 2002, Nice, France (9 2002)
18. Roche, T., Lomne, V., Khalfallah, K.: Combined fault and side-channel attack on protected implementations of aes. In: CARDIS'11 (2011)
19. Schmidt, J.m., Hutter, M.: Optical and em fault-attacks on crt-based rsa: Concrete results. In: Austrochip 2007, 15th Austrian Workhop on Microelectronics. pp. 61–67 (2007)
20. Selmane, N., Guilley, S., Danger, J.L.: Practical setup time violation attacks on aes. In: EDCC. pp. 91–96. IEEE Computer Society (2008)
21. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks 2523, 2–12 (2002)
22. Tumanski, S.: Induction coil sensors, a review. Measurement Science and Technology 18(3), R31 (2007), http://stacks.iop.org/0957-0233/18/i=3/a=R01
23. Van Woudenberg, J.G., Witteman, M.F., Menarini, F.: Practical optical fault injection on secure microcontrollers. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on. pp. 91 –99 (2011)
24. Yen, S.M., Joye, M.: Checking before output may not be enough against fault-based cryptanalysis. IEEE Transactions on Computers 49(9), 967–970 (2000)