Practical Setup Time Violation Attacks on AES

Nidhal SELMANE Sylvain GUILLEY
Jean-Luc DANGER
Institut Télécom, Télécom ParisTech
Département COMELEC, 46 rue Barrault
75 634 PARIS Cedex 13, FRANCE
Email: < nidhal.selmane@telecom-paristech.fr >

Abstract

Faults attacks are a powerful tool to break some implementations of robust cryptographic algorithms such as AES [8] and DES [3]. Various methods of faults attack on cryptographic systems have been discovered and researched [1]. However, to the authors' knowledge, all the attacks published so far use a theoretical model of faults. In this paper we prove that we are able to reproduce experimentally the random errors model used by G. Piret and J.J. Quisquater [10] to realize practical fault attack on a smart card embedding an AES encryptor by underpowering it. In spite of the fact that this method is a convenient fault injection technique to set up, it does not often appear in the open literature. We argue that the fault model is consistent with a setup violation: errors appear at the end of combinatorial logic cones, caused by an early sampling in the downwards registers. We also carry out an extensive characterization of the faults, in terms of spatial and temporal localization.

1. Introduction

Whenever a cryptographic system can be accessed physically, a couple of attacks that target directly its physical implementation can be devised. This kind of attacks is known as "side channel attacks" (SCA). They can be classified in two types, both of which providing enough information to fully compromise the security. The devices that are concerned are, for instance, smartcards (pay-TV cards, SIMs, *etc.*) or handheld terminals (mobile phones, PDAs, *etc.*). The first type of SCA is called passive attack which consists in observing physical emanations of the system, like power (Differential Power analysis, or DPA [9]) or E/H field (ElectroMagnetic Analysis, or EMA [12]). An off-line analysis of the physical measurements allow to extract the full key, by correlation or pattern matching techniques. The second type of attacks is called active attack which consists in the injection of faults during the execution of a cryptographic algorithm. Faults attacks can of course be used to obtain DoS (Denial of Services). But the real strength of fault attacks is that they enable an attacker to retrieve secret information concealed within the device [4]. From the knowledge of one or multiple couples {correct ciphertext, faulted ciphertext}, some hypotheses on the secret key can be discarded. This generic attack strategy is referred to as DFA (Differential Fault Analysis). Although active attacks were reported later (in 2001 [5]) than passive attacks (in 1998 [9]), it is shown in the literature that this method is faster than passive attack.

There are several techniques known for fault injection in a system; The variations of the supply voltage, the clock frequency, the temperature variation, or the irradiation by a laser beam will most probably lead to a wrong computation result that can be exploited to realize DFA. This kind of attack represents a real threat for the implementation of cryptographic algorithms such as the advanced encryption standard (AES).

The AES algorithm was published by the NIST in 2001 [11]. AES can operate with three different key sizes: 128, 192 and 256 bit. In the sequel, and without loss of generality, we focus on the 128-bit version of AES. AES is a substitution permutation network (SPN) product block cipher. It has an iterative structure, consisting in the repetition of ten rounds which is applied to the 16 bytes data block to be encrypted. The 16 bytes are laid out as a matrix of four columns of four bytes $s_{i,j}$, where $0 \leq i \leq 3$ and $0 \leq j \leq 3$. A round consists of a fixed sequence of transformations. Apart from the first and the last rounds, the other eight rounds are identical and consist of four transformations each. The first and last rounds are incomplete, so as to ease the decryption. The four round transformations are called SubBytes, ShiftRows, MixColumns and AddRoundKey.

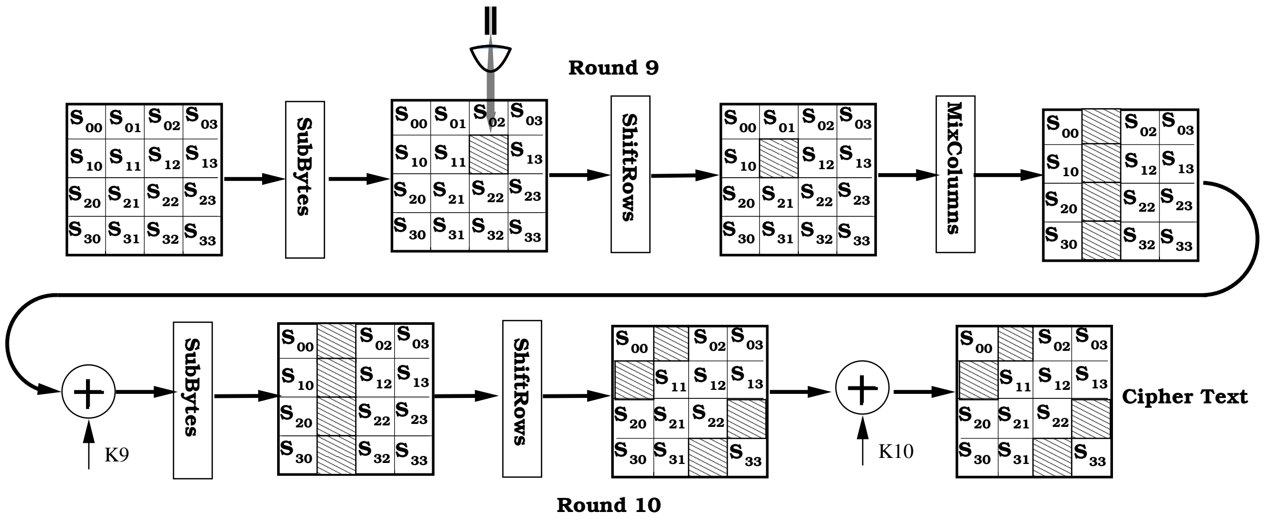

Figure 1. Effect of a fault in last but one round of AES-128.

At the conference CHES 2002, Skorobogatov [13] shows that lasers can be used to induce local faults in circuits. Following this publication, many authors presented theoretical attacks against AES exploiting faults in data-path [7, 8] and key-path [6, 14]. The number of faulty ciphertext required to break the key is comprised between 50 and 100 but the attack that requires the least number of faults is that of Gilles Piret and Jean-Jacques Quisquater (later referred to as Piret's DFA) [10]. In addition, as opposed to previously reported attacks, which require reproducible faults on insulated bits, Piret assumes only a random "byte-flip" in the last or penultimate encryption round. In fact due to the linearity of the MixColumns transformation there are 255×4 possible differences at the output of the MixColumns. If we suppose a fault on one input byte of the last MixColumns it will affect 4 bytes of the output as shown in Figure 1 and if we use a pair of correct ciphertext C and a faulty ciphertext F, we can make a guess about 4 bytes in the key using Piret's algorithm [10]. Eight single faults, located by pairs in the four columns before the MixColumns stage of the ninth round (further referred to Round9) permits the unveiling of the whole key. Alternatively, for this attack, only two single faults located before the MixColumns of the eighth round (further denoted as Round8) allows the attacker to find the 16 bytes of the key.

The rest of the paper is organized as follows. In section 2, we explain our methodology for experimentally inducing faults by under-powering the smart-card. Section 3 describes the faults analysis software used to localize faults covered by a "byte-flip" model. Section 4 presents the results from a fault acquisition campaign, in terms of spatio-temporal localization. Finally, the section 5 concludes the paper and opens perspectives for better protecting sensitive cryptographic implementations.

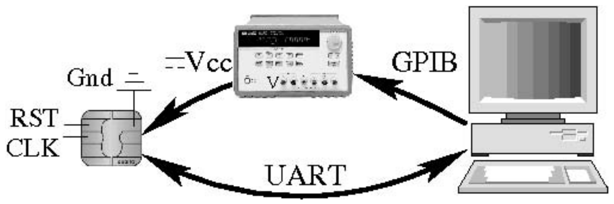

Figure 2. Experimental platform.

2. Setup Time Violations by Under-Power

2.1. Experimental Setup

The setup consists in a regular communication with a smart-card, except that the power generator supplies the card with a non-nominal continuous voltage Vcc. Figure 2 sketches the experimental setup. The power supply is furthermore controllable remotely, in such a way that various values of Vcc can be tested successively.

In our experiments, the smart card is a 130 nm ASIC, embedding, amongst others, an AES co-processor. The nominal voltage of the chip is 1.2 volts. We observe that the circuit remains functional for Vcc as low as 700 mV: at this low voltage, the CPU, in charge of controlling I/Os and of delegating encryption to the AES module, crashes. However, the AES module starts to output erroneous results for a voltage Vcc of about 800 mV and beneath. In the sequel, we take the opportunity that the smart card remains functional within the range [775:825] millivolts to explore the faults for several chosen intermediate voltages. The power supply can deliver a voltage with an accuracy of half a millivolt. Therefore, we have conducted the following acquisition campaign:

- The triples {message, key, ciphertext} are recorded for 20,000 encryptions at each 100 values of Vcc by step of 0.5 mV.
- In a view to simulate an attack, we decided to keep the key at a constant value.
- At the opposite, to collect representative results, the input message varies randomly at each encryption.

As a result, the entire acquisition campaign consists in 2,000,000 encryptions.

2.2. Motivation for this Modus Operandi

In most fault attacks on cryptographic devices, the fault model is stringent. The attacker is expected to be able to inject "single" faults in "precise" words or rounds. These constraints can be relaxed in some attack scenarios. For instance, in the Piret's DFA adversary model, the attacker needs to know neither where nor when the errors occurred. However to meet this fault model based on "byte-flip" a surgical fault injection is often carried out [13]. At the opposite we investigated a low-cost "global" fault injection technique based on under-powering. This method is not surgical but rather "global" in the sense that the whole algorithm is targeted simultaneously and continuously during the whole encryption process. The voltage reduction allows the attacker to generate single faults that most DFA models are based on. The reason, basically is that when the stress caused by the insufficient power supply remains gentle, dysfunctions do not appear suddenly and thus complex multi-faults do not show up.

3. Analysis of Faults

In this section, we analyze the two million results obtained from the acquisition campaign described in section 2.1. In order to analyze the faults, we assume that the message and the key are known by the attacker, and that they are not faulted. In other words, the faults concern only the encryption, and thus can affect only the ciphertext. We use a fast register transfer level (RTL) model of the AES ("encrypt" function), adapted to corrupt one byte of the state

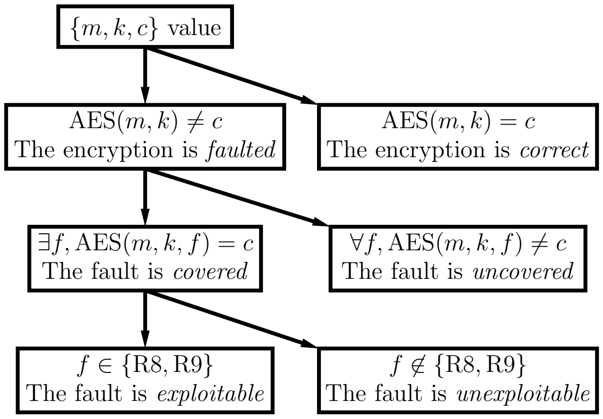
$\{m, k, c\}$ value
$\mathrm{AES}(m,k) \neq c$
The encryption is *faulted*
$\mathrm{AES}(m,k) = c$
The encryption is *correct*
$\exists f, \mathrm{AES}(m,k,f) = c$
The fault is *covered*
$\forall f, \mathrm{AES}(m,k,f) \neq c$
The fault is *uncovered*
$f \in \{\mathrm{R8}, \mathrm{R9}\}$
The fault is *exploitable*
$f \notin \{\mathrm{R8}, \mathrm{R9}\}$
The fault is *unexploitable*

Figure 4. AES faults analysis.

matrix at any round. The function is sketched in the C++ code snippet in figure 3.

In this model we can see that the f value XORed with the message permits any fault injection. The array `fault_t` is defined as:

```
// Mask ready to be applied
// to every round' state
typedef reg_128 fault_t[10];
```

where `reg_128` is typedef'ed for `char[4][4]`. To simulate a single fault injection, the `fault_t f` is initialized to full zero, with the exception of a single byte (`char`) within the `reg_128` state for a single round index. The fault analysis consists in calling this function for all the possible faults values, and compare with the ciphertext obtained from the experimental platform. Indeed, the only evidence that a fault has occurred experimentally is a faulted ciphertext. The exhaustive search of single "byte-flip" requires $(2^8 - 1) \times 16 \times 10 = 40\,800$ calls to the "encrypt" routine. This allows to:

- Find the experimental faulty ciphertext. In this case, the fault is called "covered", or
- Declare that the fault is "uncovered", if the faulty ciphertext matches none of the function "encrypt". This fault is called "uncovered".

The purpose of this implementation is to identify the typology of faults that occurs in the smart card, as shown in Figure 4. In this figure, the AES function with two arguments is the regular implementation of AES-128, whereas the AES function with three arguments is the "encrypt" routine detailed previously.

```
aes_128 aes_128::encrypt(reg_128 const& m, reg_128 const& k, fault_t const& f)
{
  aes_128 aes (m, k); // The initial state of the message and the key
  aes.AddRoundKey ();
  aes.KeySchedule (0);
  for (int round = 1; round < 9; ++round)
    {
      aes.set_state (aes.get_state () ^ f[round - 1]); // Fault injection
      aes.SubBytes ();
      aes.ShiftRows ();
      aes.MixColumns ();
      aes.AddRoundKey ();
      aes.KeySchedule (round);
    }
  aes.set_state (aes.get_state () ^ f[9]); // Fault injection
  aes.SubBytes ();
  aes.ShiftRows ();
  aes.AddRoundKey ();
  return aes; // The faulted encryption (value 'c') is complete
}
```

Figure 3. Adapted encrypt function.

4. Experimental Results

4.1. Coverage Estimation Software Tool

In this section, the fault analysis is used to find the occurrence of a single byte fault that affects the state matrix of AES. Attention is focused on the data-path, while the key schedule is assumed here to be fault-free. This choice is motivated by our goal to reproduce Piret's DFA on a real smartcard. It is straightforward to adapt the results obtained in this section to other attacks, such as attacks on the key schedule [8]. The purpose of this study is to understand the effect of under-powering the device on the faults generation throughout the encryption process.

Figure 5 shows the occurrence of faults. It appears that within about 60 mV, the device moves from an error free state to a fully erroneous behavior. As already explained in Fig. 4, faults are partitioned into "single" and "multiple", depending whether they are covered by the "encrypt" function for a fault f. Single faults have a distribution in a "bell-shape". The maximum is reached at voltage ≈800 mV, where 30 % of detected faults are single. This behavior is compatible with a fault model where errors are caused by a setup violation on critical combinatorial path. The lower the power supply, the more likely a critical path is violated, and thus the more frequent are single faults. Nevertheless, below the ≈800 mV threshold, multiple critical paths are violated. Hence an augmentation of multiple faults, and a

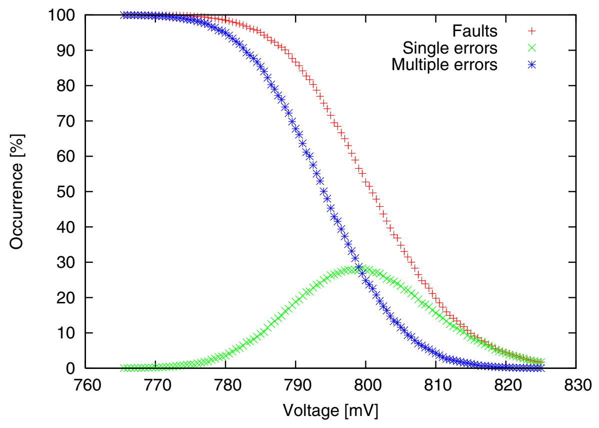

Figure 5. Occurrence of faults.

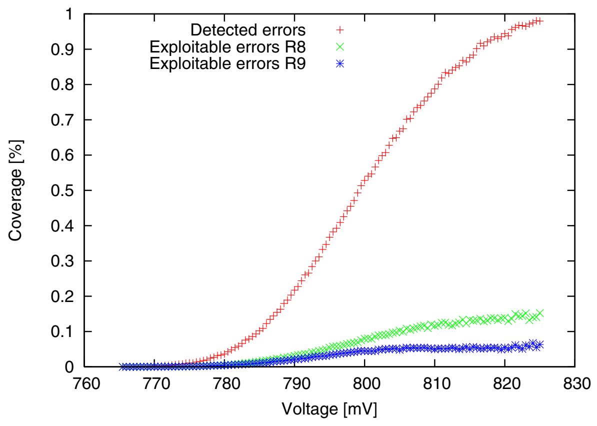

Figure 6. Coverage of single faults, and detail of exploitable faults.

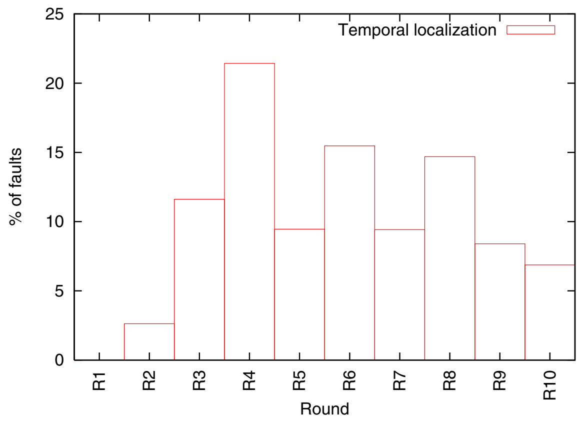

Figure 7. Temporal localization of single faults.

subsequent diminution of single faults.

4.2. Experimental Evaluation of Piret's DFA

The figure 6 presents the coverage of single faults, *i.e.* the ratio between single and detected faults. The first faults (for the highest voltage values), are almost all single. This can be seen in Fig. 6 by the fact that the coverage is close to one hundred percent. As the voltage decreases, the coverage degrades, attesting the gradual appearance of multiple faults.

Besides, we observe that, amongst the "covered" (i.e. single) faults, 16 % fall in the round 8 and 4 % in the round 9, where Piret's DFA can exploit them. As a consequence, **Piret's DFA is practical**.

Furthermore, we are now able to quantify the experimental efficiency of Piret's DFA. We can state that the average number $\mathrm{N}_{\mathrm{experimental}}$ of faults to collect experimentally (at Vcc=820 mV) in order to successfully mount Piret's DFA is:

- $\mathrm{N}_{\mathrm{experimental}} = \lceil \mathrm{N}_{\mathrm{theory}}/0.16 \rceil = \lceil 2/0.16 \rceil = 13$ for an attack in round 8, and
- $\mathrm{N}_{\mathrm{experimental}} = \lceil \mathrm{N}_{\mathrm{theory}}/0.04 \rceil = \lceil 8/0.04 \rceil = 200$ for an attack in round 9.

This proves that, in practice,the key can be found with only 13 ciphertexts. this collection is typically realised in a couple of seconds. In contrast, the full key can be extracted from the same circuit using DPA with 40K encryptions, followed by 15 minutes of computation [2].

4.3. Spatio-Temporal Characterization of Faults

The "covered" faults are analyzed in terms of spatio-temporal locality:

- in which rounds (from R1 to R10) are they more likely? (refer to Fig. 7) and
- in which byte of state are they more likely? (refer to Fig. 8).

We can see that the first round (R1) is never affected by faults. This observation was indeed predictable, since the first is special: it consists in the AddRoundKey transformation alone. Therefore, the critical path is not in this round.

It can seem counter-intuitive that faults occur at one round and not at the others. In a static timing analysis (STA) of a design, the critical path is the same for every iteration. Therefore, one might expect that if a critical path is violated at one round, then all the rounds (7 or 8) will be faulty. However, we observe single errors localized at a given round. The reason could be that the critical path is highly data-dependant.

From the analysis of Fig. 7 and 8, we have shown that the faults are not uniformly distributed over time and space. This observation, albeit not general since our setup is very particular, can be a valuable information for the designers in charge of implementing counter-measures.

We can see that the first round is never affected by faults. We show that the faults are not uniformly distributed over time and space.

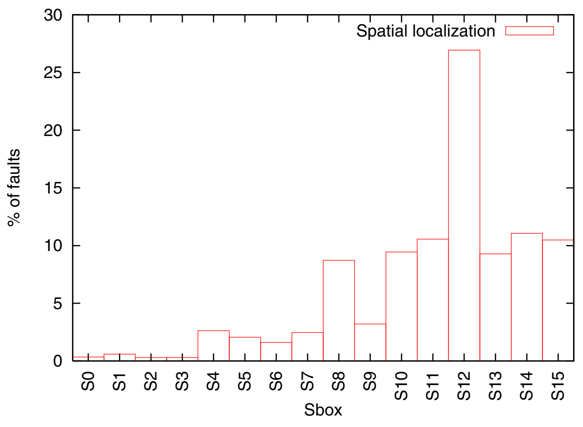

Figure 8. Spatial localization of single faults. The SubBytes box $s_{i,j}$ has index $4 \times i + j$ in the histogram.

Future researches will focus on characterizing which feature of the design (datapath or control) causes this heterogeneity.

5. Conclusion

This paper reports the first "fully-documented" experimental demonstration of Piret's DFA on real hardware. This definitely proves that Piret's DFA is practical on smart cards. Additionally, we have observed that rounds and sboxes of AES are not uniformly affected by faults.

We have not investigated the occurrence of faults in the key schedule. Some authors have presented effective attacks targeting the last rounds of the key expansion [14, 6]. In software implementations, a random injected fault is likely to happen equally in the data-path or in the key-path, because of the serial execution of the algorithm. Nevertheless, in an ASIC implementation where the key is computed in parallel with the data-path and using a global perturbation such as the one described in this paper, it is not obvious that setup violations happen in key path. Future researches will concentrate on characterizing the likelihood of these classes of attacks.

Acknowledgements

We are thankful to Guido Marco BERTONI and Filippo MELZANI for their power attack results on the smart card, and to the anonymous reviewers for their constructive comments.

References

[1] H. Bar-el, H. Choukri, D. Naccache, M. Tunstal, and C. Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006.

[2] G. M. BERTONI and F. MELZANI. Differential power analysis on the unprotected AES of SECMAT chip". STMicroelectronics AST division internal report, 2005.

[3] E. Biham and A. Shamir. Differential Fault Analysis of secret key cryptosystems. *CRYPTO 97, Springer 1997*, 1294:1513–1521, 1997.

[4] J. Blömer and J.-P. Seifert. Fault based cryptanalysis of the Advanced Encryption Standard. In *Financial Cryptography, LNCS Springer 2003*, volume 2779, pages 103–114, 2003.

[5] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(2):101–119, 2001.

[6] C.-N. Chen and S.-M. Yen. Differential fault analysis on AES key schedule and some countermeasures. In *Information Security and Privacy, LNCS Springer 2003*, volume 2727, pages 118–129, 2003.

[7] P. Dusart, G. Letourneux, and O. Vivolo. Differential Fault Analysis on A.E.S. In *Applied Cryptography and Network Security, LNCS Springer*, volume 2846, pages 293–306, 2003.

[8] C. Giraud. DFA on AES. In *Advanced Encryption Standard (AES) conference, LNCS springer*, volume 3773, pages 27–41, february 2005.

[9] P. Kocher, J. Jaffe, and B. Jun. "Differential Power Analysis". In *CHES99, LNCS Springer*, volume 1666, pages 388–397, 1999.

[10] G. Piret and J.-J. Quisquater. A differential fault attack technique against SPN structures, with application to the AES and Khazad. In *CHES'03, Springer, LNCS*, volume 2779, pages 77–88, 2003.

[11] F. I. P. S. F. Publication. *Announcing the Advanced Encryption Standard (AES)*. Number 197. November 2001.

[12] J.-J. Quisquater and D. Samyde. "ElectroMagnetic Analysis (EMA) Measures and Counter-Measures for Smart Cards". In *e-SMART, LNCS Springer*, volume 140, pages 200–210, 2001.

[13] S. P. Skorobogatov and R. J. Anderson. Optical Fault Induction Attacks. In *CHES'02, Springer 2002*, volume 2523, pages 2–12, 2002.

[14] J. Takahashi, T. Fukunaga, and K. Yamakoshi. DFA Mechanism on the AES Key Schedule. In *FDTC 2007 Workshop*, pages 62–74, 2007.