Security analysis and fault injection experiment on AES

Olivier Faurax[1,2], Traian Muntean[2]

[1]*École des Mines de St Étienne - Site Georges Charpak, Laboratoire SESAM, Avenue des Anémones, 13120 GARDANNE, FRANCE*
[2]*Université de la Méditerranée, "Systèmes Informatiques Communicants", 13288 MARSEILLE, FRANCE*
E-mail: faurax@emse.fr

Robustness of cryptographic circuits against fault attacks is a great concern to ensure security. In this paper, we present a security analysis of such circuits and a fault injection methodology and tool (PAFI). We apply them to AES as a case study and show that injected faults that lead to known fault attacks match our analysis.

Mots-clés: fault attacks, cryptography, AES

1 Introduction

Cryptographic circuits are often a foundation of security in nowadays systems. As a consequence, attacks on them are critical and can be used to defeat security policies.

In this context, the protection against attacks is a major concern. A fault attack uses a physical perturbation of the circuit in order to obtain faulty computations. These miscomputed results can enable cryptanalysis and reveal secret data. Several cryptosystems are concerned by this type of attacks : RSA [BDL97], DES [BS97] and AES [Gir04][DLV03][PQ03].

The robustness against fault attacks must be evaluated to ensure fault tolerance and security. This can be achieved by injecting faults in the system in order to validate its behavior under fault attacks. It is possible to do this using physical fault [GKT89][AAA+90][MRMS94], but this can also be done using built-in debug mechanisms [FSK97][BPRR98].

Another approach is to use fault injection during simulation to provide robustness evalutation before silicon IC manufacturing in an relatively unexpensive manner. Simulating the circuit permits to inject fault by modifying its description [JAR+94][LH00][ZME03] or to add a custom fault injector in the design [FMR06]. Our approach is to use an unmodified description of the circuit to be very accurate regarding to the corresponding physical circuit.

However, some properties of cryptographic algorithm can be used to predict the temporal sensitivity of circuits. In this paper, we propose a metric of sensitivity against fault attacks for circuits and validate it using fault injection in simulation on AES.

This paper is organized as follows : the algorithm AES and its implementation is introduced in section 2. We describe our analysis of sensitivity in section 3 and apply it to AES. Then, we present our injection methodology and tool in section 4. To conclude, future work is described in section 5.

2 AES

2.1 AES algorithm

AES [DR98] is a well-known cryptographic block cipher. It is a substitution-permutation network that takes as input a 128-bit plain text of and a 128-, 192- or 256-bit cipher key.

In this paper, we consider the variant with a 128-bit key length. This AES is made of 10 rounds where the input is scrambled using a round key. Each round, the next round key is computed from the current round key. The first input is the plain text and the first round key is the cipher key.

The part of the round that deals with the data can be divided in four steps : AddRoundKey, SubBytes, ShiftRows, MixColumns. The 128 bits of data are viewed as a 4x4 matrix of 16 bytes.

1. AddRoundKey computes the XOR between the input and the round key. This is the only step of the round that takes into account the value of the key.

2. SubBytes performs a substitution over $GF(2^8)$ of each byte. This substitution provides the non-linearity of the round.

3. ShiftRows operates on each 4-byte row : it cyclically shifts the 4 bytes of a row by 0, 1, 2, 3 respectively. This scrambles the columns at each round by making them to become diagonals.

4. MixColumns applies a linear transformation over $GF(2^8)$ on each 4-byte column. The diffusion of this step is only on a quarter of the data state. However, two MixColumns combined with a ShiftRows provide a complete diffusion of the AES data state.

The keyschedule of AES generates round key for step $n+1$ using the round key of step n, the round key 1 being the cipher key. The whole process is described in details in [DR98]. The diffusion of the keyschedule is at least four bytes, depending of the byte considered.

2.2 Implementation of AES

Our AES circuit takes four 32-bit words for the data and four 32-bit words for the cipher key. Then it calculates the result and output it as four 32-bit words.

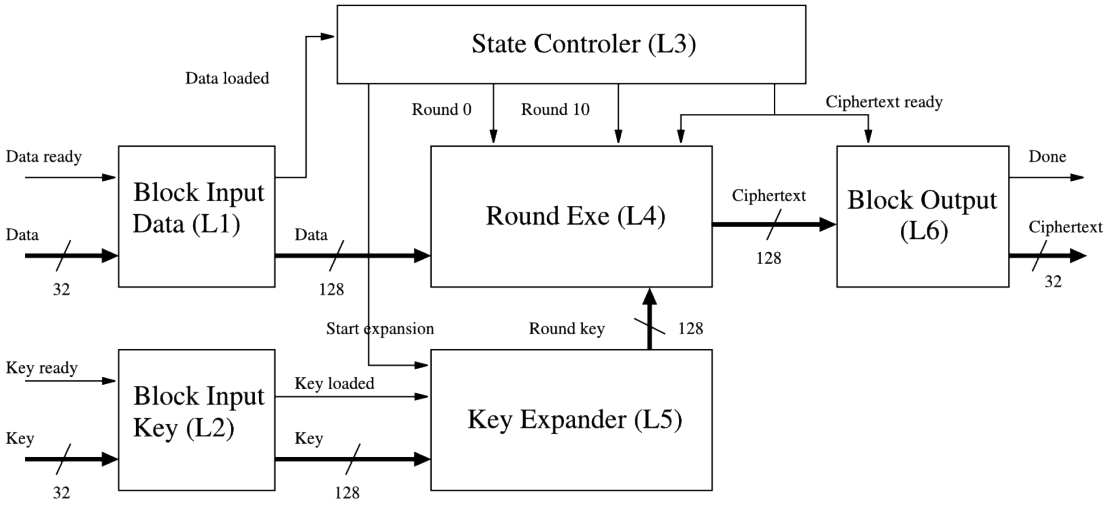

Fig. 1: Architecture of AES

The AES is composed of six parts labeled L1 to L6 in figure 1.

- L1 and L2 are input blocks for the plain text and the cipher key. Data and key are 128-bit long but the input is 32-bit wide, so L1 and L2 have to buffer the data and key during the loading process.

- L3 is a state controler that drive the synchronisation between the other parts.

- L4 contains the four AES substeps described in 2 with a 128-bit register representing the current state.
- L5 is a block that computes round key from the key provided.
- L6 outputs the 128-bit long result into four 32-bit words in a similar way as L1 and L2 for the input.

3 Analysis of security of AES

For an intruder, a fault in a system can reveal useful information, only if the effect of the fault is known and limited.

A fault that induces a faulty output similar to a byzantine error is difficult to exploit as an attacker has to make a lot of assumptions on the parameters used. That is why the diffusion of the algorithm is important for robustness.

Exploitable faults are usually close to the outputs because it makes the faulty computation more accessible.

The execution of a synchronous circuit is a set of successive states that outputs a final state.

- The diffusion (d_1) is the part of the final state that depends of a bit of the current state. d_1 is 1 when the whole final state depends of the current state.
- The distance (d_2) is a distance metric between the current state and the output. d_2 is linear, starting at 1 on the beginning of the circuit and 0 at the end of the computation.
- The sensitivity (s) is a metric of the sensitivity of the circuit.

When $d_1 = 0$ or $d_2 = 0$, the current state is the final state. The computation is finished, secret data is not involved, the sensitivity is 0.

Otherwise, we define the value of the sensitivity as

$$s = \frac{1}{d_1 d_2}$$

.

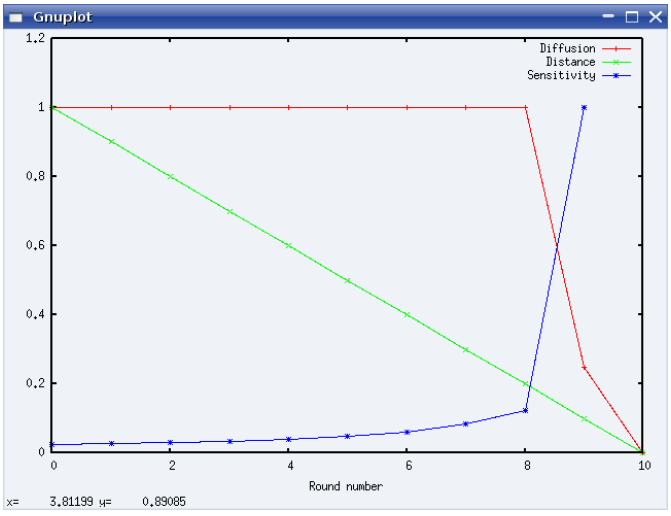

Fig. 2: Theorical sensitivity of AES

The results on AES are displayed on the figure 2 (the sensitivity has been reduced by 40 to fit in the window).

We can see that the AES sensitivity is high during the 8th and 9th round. This will be confirmed by our fault injection experiment.

4 Fault injection and results

4.1 Yet another fault injection tool : PAFI

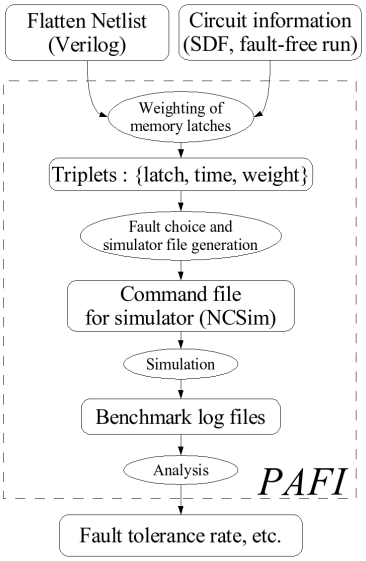

Fig. 3: PAFI

The purpose of PAFI (Prototype of Another Fault Injector, figure 3) is to use unmodified modelization of the circuit to take into account very accurate details of the circuit (gates, delays). Our approach is to evalutate the circuit dependability in simulation while being close to the physical design.

The circuit is defined in a Verilog netlist. The purpose of the first step is to parse the netlist and to find the latches and their logical cone. Then, a structural weight can be computed according to the fault model the user wants to focus on.

At the same time, the netlist can be simulated without injecting faults in order to provide information about the fault-free run. This information are used to compute a dynamic weight that can be combined with structural weight to take in account expected activated region.

Then, the possible faults and their weight are presented to the user that can choose the amount of injection to perform, depending on the time he can spend and the accuracy he needs.

From the chosen list of faults, our tool generates a command file for the simulator used (Cadence NCSim). This command file describes the simulations and the fault injections. As we do not modify the circuit, the only versatile fault injection possibility is to use the built-in command of the simulator. The circuit execution is simulated until the fault injection time. Then, the state of the circuit is modified by the similator and the simulation is resumed. This part of the tool is specific to the simulator.

However, our tool is designed to be modular : each step is performed by an independant program whose results are stored in an XML files. This allows, for instance, to easily replace one module to adapt it to another simulator.

During each simulation, the circuit under test is manipulated by a benchmark that provide input signals, check output signals and log results on files.

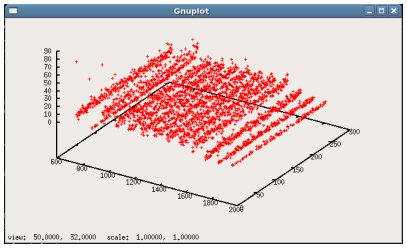

Fig. 4: Output of PAFI using Gnuplot

A custom analyser reads these log files and produces the expected computation of safety rate and user-defined analysis. The outcome can be graphically displayed (Cf. figure 4).

4.2 Injections on AES

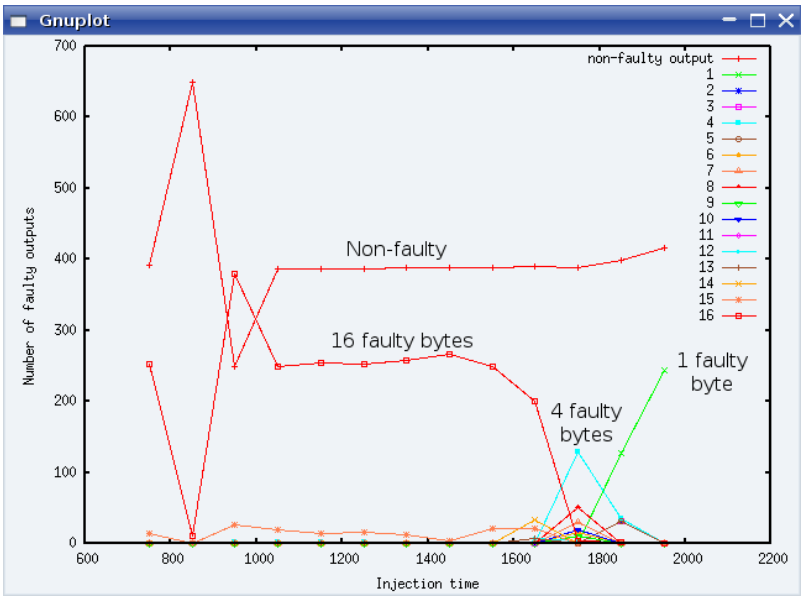

Fig. 5: Faulty output bytes on AES

We made an exhaustive injection on AES. Our benchmark is to input known data and cipher key, to log the output (if any) and to compare it with the expected result.

Using a clock at 10 MHz, data and key are loaded in 750 ns. The computation occurs between 750 ns and 1950 ns, with a clock cycle of 100 ns. The results is expected at 1950 ns (first 4 bytes) until 2350 ns.

We injected on the 664 latches of the design during the computation that took 13 clock cycles, leading to 13*664 simulations, injecting one bit-flip for each simulation.

The results are shown on figure 5. On average, 60.2% of injected faults do not lead to faulty results and 27.6% of faults induce 16-byte faulty outputs.

The faults injected at the beginning of the computation (750 and 850 ns) produce errors that are dependent of the design of the AES and are not relevant here due to the fact that the first round of the AES starts at 950 ns.

Between 950 ns and 1750 ns, results are fault-free or 16-byte faulty outputs. Some injections near the end of the computation generate 4-byte errors : this verify the diffusion properties of AES (4 bytes by round, all bytes after 2 rounds).

The outputs with only one faulty byte at the end are due to an injection just before the output of the result.

According to known attacks ([DLV03][PQ03]), AES is sensitive to fault attack when partially faulted output are computed.

Our analysis matches the results of fault injections : the exploitable 4-byte errors occurs at the 9th round.

5 Conclusion

In this work, we proposed a security analysis that aims cryptographic circuits and validated it on AES using PAFI, our fault injection tool. The fault injections leading to known attacks match the prediction of our analysis.

Further studies will be focused on selecting injected faults to be able to extend the fault model to multiple faults without increasing the time needed to perform the simulations.

References

[AAA+90] Jean Arlat, Martine Aguera, Louis Amat, Yves Crouzet, Jean-Charles Fabre, Jean-Claude Laprie, Eliane Martins, and David Powell. Fault injection for dependability validation: A methodology and some applications. *IEEE Trans. Softw. Eng.*, 16(2):166–182, 1990.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. *Lecture Notes in Computer Science*, 1233:37–51, 1997.

[BPRR98] A. Benso, P. Prinetto, M. Rebaudengo, and M. Sonza Reorda. EXFI: a low-cost fault injection system for embedded microprocessor-based boards. *ACM Transactions on Design Automation of Electronic Systems.*, 3(4):626–634, 1998.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[DLV03] P. Dusart, G. Letourneux, and O. Vivolo. Differential fault analysis on a.e.s. Cryptology ePrint Archive, Report 2003/010, 2003. http://eprint.iacr.org/.

[DR98] J. Daemen and V. Rijmen. Aes proposal: Rijndael, 1998.

[FMR06] Julien Francq, Pascal Manet, and Jean-Baptiste Rigaud. Material emulation of faults on cryptoprocessors. In *Proceedings of Sophia Antipolis forum of MicroElectronics (SAME) 2006*, 2006.

[FSK97] Peter Folkesson, Sven Svensson, and Johan Karlsson. Evaluation of the thor microprocessor using scan-chain-based and simulation-based fault-injection, 1997.

[Gir04] Christophe Giraud. Dfa on aes. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2004.

[GKT89] U. Gunneflo, J. Karlsson, and J. Torin. Evaluation of error detection schemes using fault injection by heavy-ion radiation. In *Proceedings of the 19th International Symposium on Fault Tolerant Computing, (FTCS-19), IEEE, Austin, Texas, USA*, pages 340–347, 1989.

[JAR+94] Eric Jenn, Jean Arlat, Marcus Rimen, Joakim Ohlsson, and Johan Karlsson. Fault injection into VHDL models: The MEFISTO tool. In *Proceedings of the 24th International Symposium on Fault Tolerant Computing, (FTCS-24), IEEE, Austin, Texas, USA*, pages 66–75, 1994.

[LH00] R. Leveugle and K. Hadjiat. Optimized generation of vhdl mutants for injection of transition errors. In *SBCCI '00: Proceedings of the 13th symposium on Integrated circuits and systems design*, page 243, Washington, DC, USA, 2000. IEEE Computer Society.

[MRMS94] Henrique Madeira, Mario Zenha Rela, Francisco Moreira, and Joao Gabriel Silva. RIFLE: A general purpose pin-level fault injector. In *European Dependable Computing Conference*, pages 199–216, 1994.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against spn structures, with application to the aes and khazad. In *Cryptographic Hardware and Embedded Systems − CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.

[ZME03] Hamid R. Zarandi, Seyed Ghassem Miremadi, and Alireza Ejlali. Dependability analysis using a fault injection tool based on synthesizability of hdl models. In *DFT '03: Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 485–492. IEEE Computer Society, 2003.