HAL
archives-ouvertes.fr

Material Emulation of Faults on Cryptoprocessors

Julien Francq, Pascal Manet, Jean-Baptiste Rigaud

▶ **To cite this version:**

Julien Francq, Pascal Manet, Jean-Baptiste Rigaud. Material Emulation of Faults on Cryptoprocessors. SAME'06, Oct 2006, Nice, France. pp.ISBN 2-9524014-1-1, 2006. <emse-00494246>

HAL Id: emse-00494246
https://hal-emse.ccsd.cnrs.fr/emse-00494246

Submitted on 22 Jun 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SAME 2006 Forum

SESSION: TOOLS & METHODOLOGIES

Material Emulation of Faults on Cryptoprocessors

Julien Francq[1] Pascal Manet[1] Jean-Baptiste Rigaud[2]

[1] CEA-LETI [2] Ecole Nationale Supérieure des Mines de St Etienne

[1,2] Centre Microélectronique de Provence, Laboratoire SESAM

Avenue des Anémones - Quartier Saint-Pierre, 13541 GARDANNE

E-mail: francq@emse.fr – Telephone number: +33 (0)4 42 12 68 74

Abstract: this paper describes a block that can be added to a crypto-processor embedded on a FPGA. This block enables to simulate the co-processor behaviour when faults are injected. Three fault models are used and an example with AES is given. The aim of such a block is to speed up the test of countermeasures on a FPGA before running the chip in fab.

1. Introduction

Some works have proven that injecting faults during a cryptographic process (AES for example) is an efficient way to recover the Cipher Key ([1], [2]): this is called the "Differential Faults Analysis" (DFA). There are many known techniques to induce faults into a device: variations in supply voltage, variations in the clock frequency, temperature, laser…This latter technique is very often used because a small area of a circuit can be targeted precisely thanks to laser properties [3].

In order to protect crypto-processors against fault attacks, we focus on using prototyping to evaluate the strengths of countermeasures. This approach is a trade-off between long Hardware Description Language simulations and expensive real chip laser benchmarks [4].

This paper deals with a way to induce faults during a cryptographic algorithm. A parallel block called "fault generator" is added: it can simulate different kinds of faults at any step during the course of the execution.

In Section 2, we present the different fault models we decided to insert in our block. Section 3 describes our block in more detail. Section 4 illustrates with an example of application, which is its insertion in AES algorithm. The last section concludes this paper.

2. Different fault models

The "fault generator" can inject three different types of fault chosen by the user (evaluator): the "bit-flip" model (the value of the affected bit is inverted), the "stuck-at-fault zero" model (SAF0: the value is forced to 0) and the "stuck-at-fault one" model (SAF1: the value is forced to 1). Other kinds of faults could be implemented as well, but we consider that these models emulate most of the faults used in existing fault attacks.

The choice of these faults is not harmless. The "stuck-at" model is very convenient, because it is representative of numerous kinds of physical failures [5]. It can also be used to emulate a laser attack. For example, if a circuit is attacked during a long time by a laser, we assume that some wires may be stuck at a constant value.

The "bit-flip" model is the most used by attackers.

3. Description of the "fault generator"

The specifications of such a block are the following: first, it has to be as generic as possible in order to be adapted to any cryptographic system; second, it must consider the three fault models described in Section 2; third, it has to inject these faults at any moment during the execution.

Obviously, the insertion of this block must have a limited impact on the execution time of our initial system. Area and power consumption are not critical parameters, because we implement this structure in a FPGA platform, not in an ASIC.

Thanks to this structure, the user will be able to choose the address of corrupt bits, the kind of faults and the moment of injection. Here is described the structure of our block. The "fault generator" is built with two blocks called "n_bits" and "decoder" (see Figure 1).

Thanks to the decoder, the user can induce three different kinds of faults: bit-flip, SAF0, and SAF1. This decoder has a clock, an asynchronous reset and an input which initializes the start of an internal counter. The result computed by the decoder, called "injected_fault", is sent to one of the inputs of the block "n_bits".

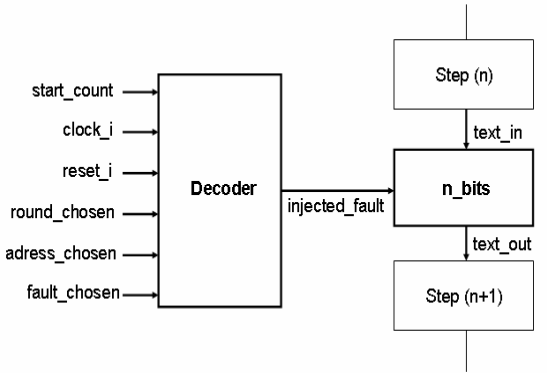

Figure 1: Structure of the "fault generator"

The result "injected_fault" has a particular size. Indeed, the kind of fault for each input bit is a 2-bit vector.

The decoder has 3 components, which are called "counter", "comparator" and "fault_selector" (see Figure 2). The component "counter" is a counter which starts with a synchronisation signal provided by the cryptographic algorithm. "Comparator" compares the counter with the injection time chosen by the user: when equal, it enables the "fault_selector". Thus, the chosen fault is injected into the circuit at the right time.

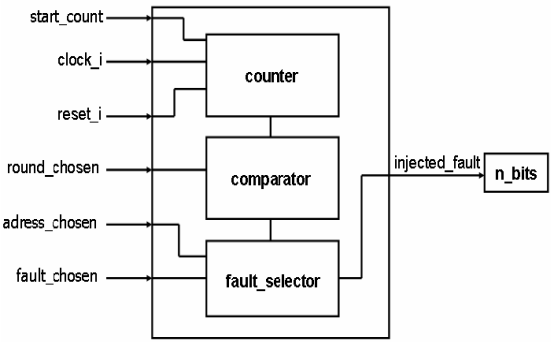

Figure 2: Structure of the "decoder"

4. Illustration: Fault injection on AES

In order to inject some faults at the right round and at the right transformation of the AES [6], we have to insert the block "n_bits" before each transformation of the algorithm, which are called SubBytes, ShiftRows, MixColumns, and AddRoundKey (and in the Key Schedule). However, due to the linearity of ShiftRows and AddRoundKey transformations, there is no use to implement the "n_bits" block before AddRoundKey and ShiftRows (see Figure 3).

All the blocks "n_bits" are linked to their own decoder. Thus, a fault can be injected at the right transformation thanks to synchronisation signal sent to it.

Moreover, the synchronisation input of the counter is linked to the Key Schedule start signal. Thus, this block is able to put a fault in any round of the AES algorithm: it allows a space-time insertion of faults.

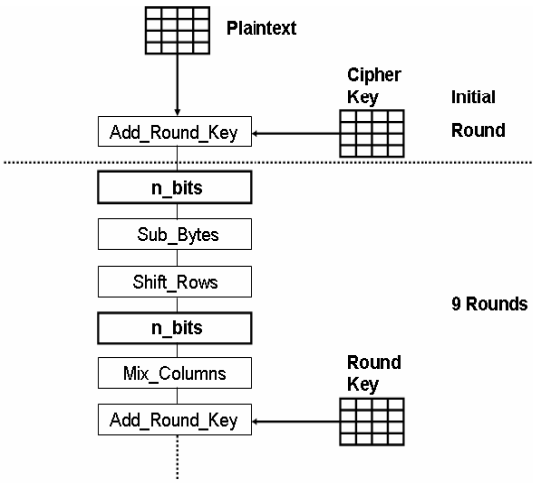

Figure 3: Insertion of the block "n_bits" in AES

The user of this block should only respect the following format: the input format of the block "n_bits" must be adapted to the input format of the temporary results of transformations (which are called "states") to avoid conflicts. Thus, in our application, the states format is a subtype in a package, defined as a (4×4)-byte matrix. Moreover, another subtype must be defined for the "injected_fault", which is a (4×4) 16-bit matrix.

5. Experimental results

We implemented our structure on a XCV2000E+ Logic Module from Xilinx, Virtex − E™ family. The design was developed with the Xilinx ISE framework. The synthesis was performed with XST application and all the simulations (functional, post-synthesis and post place and route) with ModelSim.

First, a reference version of AES with no fault injection was implemented on this FPGA. This implementation takes 1005 slices for a 60 MHz clock frequency. A slice is made up of two logic cells which are the basic elements in a Xilinx's family FPGA. This basic element is composed of a four variable logic function generator, a carry logic and a memory element.

A second version of the AES with the fault generator was realized. It takes 1150 more slices.

Area was not a critical criterion; this version represents less than 10% of the total amount of available space on the device. The speed requirement matched: only two gates were added in the critical path and the required clock frequency stays unchanged.

6. Conclusion

The block "fault generator" has been approved thanks to its implementation on a FPGA: it respects our specifications.

Thus, we are able to validate countermeasures against DFA faster than before: the required time to execute simulations is greatly decreased due to this method.

Unfortunately, this block is able to inject at most three faults at each round. The next step is to modify our structure in order to inject the number of faults we want, at any moment of the execution. This new architecture is under development.

The results concerning this new block will be provided during the "Academic Poster session".

Table of contents

References

[1] Eli Biham and Adi Shamir. *Differential Fault Analysis of secret key cryptosystems*. In B.S. Kaliski Jr., editor, Advances in Cryptology − CRYPTO'97, volume 1294 of Lecture Notes in Computer Science, pages 513–525. Springer, 1997.

[2] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall and Claire Whelan. *The sorcerer's apprentice guide to fault attacks.* In First Workshop on Fault Detection and Tolerance in Cryptography − FDTC 2004, Florence, Italy, June 2004.

[3] Sergei P. Skorobogatov and Ross J. Anderson. *Optical fault induction attacks.* In B.S. Kaliski Jr., C¸ .K. Ko¸c, and C. Paar, editors, Cryptographic Hardware and Embedded Systems − CHES 2002, volume 2523 of Lecture Notes in Computer Science, pages 2–12. Springer, 2002.

[4] Olivier Faurax, Laurent Freund, Frédéric Bancel and Traian Muntean. *Une méthode générique pour l'injection de fautes dans les circuits.* To be published in Journées Nationales du Réseau Doctoral en Microélectronique 2006 − JNRDM 2006, May 2006.

[5] F. Azaïs, S. Bernard, Y. Bertrand, M.L. Flottes, S. Pravossoudovitch, C. Landrault, M. Renovell, P. Girard, L. Latorre and B. Rouzeyre, *Test de Circuits et de Systèmes Intégrés*. EGEM Collection, Hermès Editor, 2004.

[6] NIST. *Announcing the Advanced Encryption Standard (AES).* Federal Information Processing Standards Publication, n. 197, November 26, 2001.