A generic method for fault injection in circuits

Olivier Faurax[1,3], Laurent Freund[1], Assia Tria[2], Traian Muntean[3], Frédéric Bancel[4]
[1] École des Mines de St Étienne - Site Georges Charpak, Laboratoire SESAM,
Avenue des Anémones, 13120 GARDANNE, FRANCE, E-mail: faurax@emse.fr
[2] CEA-LETI, Laboratoire SESAM, Avenue des Anémones, 13120 GARDANNE, FRANCE
[3] Université de la Méditerranée, "Systèmes Informatiques Communicants", 13288 MARSEILLE, FRANCE
[4] STMicroelectronics, Division Smartcard, Zone Industrielle de Rousset, 13106 ROUSSET Cedex, FRANCE

Abstract— **Microcircuits dedicated to security in smartcards are targeted by more and more sophisticated attacks like fault attacks that combine physical disturbance and cryptanalysis. The use of simulation for circuit validation considering these attacks is limited by the time needed to compute the result of the chosen fault injections. Usually, this choice is made by the user according to his knowledge of the circuit functionnality. The aim of this paper is to propose a generic and semi-automatic method to reduce the number of fault injections using types of data stored in registers (latch by latch).**

I. Introduction

A fault attack on a circuit consists in a physical perturbation done on one or more parts of it in order to exploit changes in the result. The main idea is to get results of related computations that differ only by one fault to be able to extract critical data by cryptanalysis. The first works about fault attacks are DFAs (*Differential Fault Analysis*) leading to attacks on RSA [1] and on DES [2]. Then, several attacks were performed on AES [3][4][5].

To design fault tolerant circuits, one musts take in account its behaviour when faults occur. We suggest simulation-based fault injection that can be used before silicon IC, that generates reproductable faults in a less expensive way.

The circuit behaviour is simulated with the chosen fault model (bit-flip, stuck-at, etc.). In practice, the model is provided to the simulator. Then, the simulation runs until the injection moment is reached. The circuit state is modified and the simulation follows. When simulation stops, circuit behaviour can be analyzed.

However, current circuit complexity (gates number, metal levels, etc.) does not permit simulation of all possible faults. Injection points number can be high, especially if all injection times are considered. This number increases too with fault multiplicity. For single faults, simulation time is linear in fault locations number and in fault times number.

In our work, we investigate the criterions of injection points choice according to their functionality in the circuit. The target is to guide, *a priori*, the injection campaign to the faults that have a high probability to have an impact on the system behaviour.

This article is organized as follows : some related works on fault injection are commented in section 2. We present our functional weight system of critical latches in section 3. Then, we discuss the relative importance of faults in the propagation paths of these critical latches in section 4. In section 5, we show our generic and semi-automatic method. To conclude, future works are described in section 6.

II. Related works

Fault injection tools and methods exist for about twenty years. We briefly present a non-exhaustive set of such tools.

MEFISTO [6] is one of the most important contribution as it was able to simulate multi-level faults on circuits described in VHDL, using saboteurs and mutants.

VERIFY [7] proposes a new fault injection technique by extending the VHDL signals syntax. This does not need recompilation as MEFISTO's mutants. Nevertheless, this technique needs a specific compiler that understands these extensions.

SINJECT [8] is a simulation-based injection tool that uses MEFISTO's mutants and saboteurs, adding support for non-synthetizable mixed-mode (VHDL/Verilog) circuit descriptions.

FITSEC [9] divides the circuit to emulate one part on a FPGA and simulates the remaining part. This technique drastically decreases the injection time by a factor of 100.

This tools enable one to conduct simulation-based fault injections without helping him to choose faults to reduce computation time.

Some other tools guide the user to a subset of possible injections. Two approaches have been proposed : to select injections that have a high probability of disturbing the system (to obtain a lower bound of the coverage rate) or to select representative injections (to obtain a coverage rate close to the real one).

DEPEND [10] reduces the number of faults by analysing the workload.

Güthoff and Sieh [11] analyse the execution of instructions without fault (golden run) on a processor to simulate faults only on registers used very often and only when theirs values make sense.

The technique of fault expansion [12] groups faults to simulate only one representative of each group. However, this method is only effective when each fault equivalence class is a significant portion of the fault population [13].

Our approach advances the art because it is based on types of data stored in latches to deduce injection points of choice.

1-4244-0898-9/06/$20.00 ©2006 IEEE.

III. Functional weight of latches

Our model is transient faults that ends in a change of the system's state. The global state of a circuit is stored in the set of its latches. So, the faults of our model are the ones that perturb these memory elements.

To distinguish critical latches, our approach adds a weight to every latch to represent its impact on the circuit behaviour. This weight will help the user to choose important latches to perturbate during the simulation.

The weight takes into account structural and functional factors of circuits.

Structural factors can be deduced from the circuit netlist : logical cone associated to a latch, type of latch, etc. There are not treated in this paper.

Functional factors deal with the use of the latch regarding to the whole circuit : data type, critical value for circuit behaviour, etc. This informations must be supplied by the user as they cannot be deduced from the circuit description. They are needed to compute the functional part of the weight.

We consider three critical data types : secret data, control data and outputs data.

Secret data has to be isolated from the outside in order to avoid an information leakage. If there exists observable data whose value depends on secret data, it has to be protected. Indeed, a wrong result generated by a perturbation could allow to deduce the secret data by cryptanalysis.

During the circuit execution, control data is results of tests and drives the circuit behaviour. A perturbation on corresponding latches impacts security, for example by validating wrong authentication or by giving access to protected resources.

Outputs data are easily readable. A perturbation that modify only a part of the result facilitates cryptanalysis. Dusart, Letourneux and Vivolo [4] show this on AES by modifying only a quarter of the result (4 bytes out of 16).

The first step of our wheighting method (that is described in section V) is about adding a weight to latches containing these three data types.

IV. Related perturbations

Latches are connected by logical cones. If a perturbation on a latch B (cf. figure 1) changes the circuit behaviour, it is likely that perturbations on latches of its propagation path (A_i and C_i) can induce a modification of the circuit behaviour.

Faults on these latches are called *related perturbations* and are divided in two groups : *anterior perturbations* and *posterior perturbations*.

A. Anterior perturbations

A fault injected on the input latches of the logical cone of B can have a consequence on the value of B. This latches are written A_i and these perturbations are called *anterior perturbations* because B is on the propagation path of the A_i.

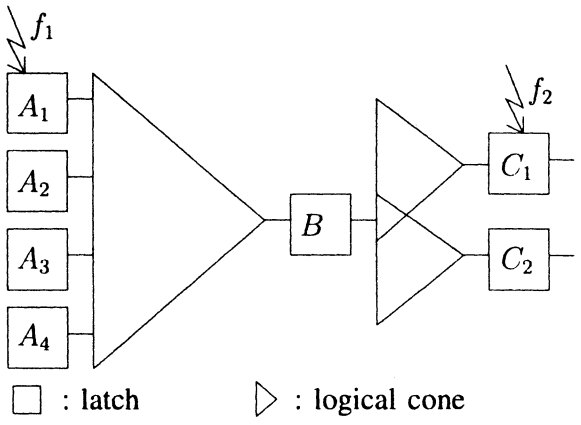

Fig. 1. Related perturbations (f_1 : anterior, f_2 : posterior)

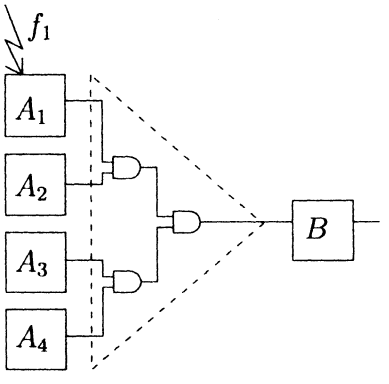

Fig. 2. Example of anterior perturbation

On the case of figure 2, a value change of one of the A_i latches is equivalent to a fault on the B latch.

If B contains some secret data, an adressing fault is an anterior perturbation, as the behaviour is modified even if the latch containing the secret data was not directly touched by the fault.

if B contains some control data, an anterior perturbation will modify the circuit behaviour if the test result is changed. It is also possible to have a circuit deadlock which is acceptable if it does not supply a wrong result.

An injected fault near the outputs will provide a partially wrong result that can lead to a differential analysis. That is why outputs data are considered as critical : anterior perturbations can leak informations.

Thus, the weight of a latch B can induce a cascade phenomenon on the weight of the previous latches on the propagation path (A_i).

B. Posterior perturbations

In the same way, perturbing output latches of the logical cones whose B is an input will probably change the behaviour of a circuit part. These latches are written C_i and these perturbations are called *posterior perturbations* because the C_i are in the propagation path of B.

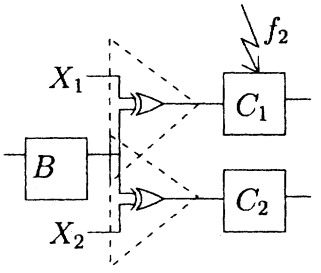

Fig. 3. Example of posterior perturbation

Figure 3 shows that a value change of one of the C_i latches is equivalent to a fault on the B latch for a part of the circuit (the one depending on C_1).

For example, perturbing a latch containing a temporary result depending on secret data can reveal informations. Works of Yen and Joye [14] show that injection of a safe error (i.e. error that does not lead to a wrong result) on a temporary result during exponentiation can provide secret informations.

A posterior perturbation on control data can completely change the behaviour of a circuit part. This can lead to an unexpected state and/or an illegal state according to the security specifications of the system.

With posterior perturbations, a weight on a latch B can induce a weight on the latches of the logical cones depending on it (i.e. the C_i).

C. Combining weights

Weights will decrease while going up or down the propagation paths. They will be combined in order to reveal potentially interesting injection points.

Attacks on AES [3][4][5] are based on some faults that are near the outputs and the secret round keys : these faults are anterior perturbations for outputs data and posterior perturbations for secret data.

V. Weight determination method

In the first step, the user provides the initial couple list {latch, weight} for the latches dealing with secret, control or outputs data. This is the list of all B latches and their associated weights, as shown on figure 1,

From these informations and the circuit netlist, the second step is to compute the weight for related latches (A_i and C_i). This couples {latch, weight} are then added to the list. This second step is iterated enough to obtain a stable list (i.e. that is not modified between two iterations).

Finally, it provides as a result a list of couples {latch, weight} that guides the user's choice of fault injection to simulate for the circuit validation. The computation is more formally described on the algorithm 1.

This method is generic as there is no hypothesis on the type of the considered circuit. Moreover, contrary to the first step that needs user intervention, the second step can be entirely automatic.

Algorithm 1 Latch weighting algorithm

Require: $circuit \leftarrow$ circuit description
Require: $new\_list \leftarrow$ initial set of couples $(critical\_latch, weight)$
Require: $old\_list \leftarrow \{\}$
Ensure: final list of weights
  while $new\_list \neq old\_list$ **do**
    $old\_list \leftarrow new\_list$
    for all $latch$ in $circuit$ **do**
      $tmp\_weight \leftarrow Compute\_weight(latch, circuit, old\_list)$
      if $tmp\_weight > 0$ **then**
        $new\_list \leftarrow new\_list \cup \{(latch, tmp\_weight)\}$
      end if
    end for
  end while
  return new_list

VI. Conclusion & future works

In this work, we propose the base of a new method for circuit validation under fault attacks. This generic and semi-automatic method targets critical injection points to make the simulation time acceptable.

We currently work on inital weight valuation of latches (values of B) regarding to their functionality and on the influence of B latches weights over the ones of A_i and C_i in the case of our simplified model described in figure 1. Thereafter, we will work on a multiple model (interconnections of several simplified models) whose weights will be deduced from several weights of the simplified models.

The perspectives of this work will be to make hypothesis on values and computation formulas, and then to experimentally test these choices with the tool we develop nowadays. The results of this study will show the different factors of functional weight.

Next, we have to combine functional weight and structural weight that takes in account circuit design near the injection points considered.

References

[1] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," *Lecture Notes in Computer Science*, vol. 1233, pp. 37–51, 1997.

[2] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, ser. Lecture Notes in Computer Science, B. S. K. Jr., Ed., vol. 1294. Springer, 1997, pp. 513–525.

[3] C. Giraud, "Dfa on aes," in *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, ser. Lecture Notes in Computer Science, H. Dobbertin, V. Rijmen, and A. Sowa, Eds., vol. 3373. Springer, 2004, pp. 27–41.

[4] 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/.

[5] G. Piret and J.-J. Quisquater, "A differential fault attack technique against spn structures, with application to the aes and khazad," in *Cryptographic Hardware and Embedded Systems – CHES 2003*, ser. Lecture Notes in Computer Science, C. D. Walter, Çetin Kaya Koç, and C. Paar, Eds., vol. 2779. Springer, 2003, pp. 77–88.

[6] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. 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*, 1994, pp. 66–75. [Online]. Available : citeseer.ist.psu.edu/jenn94fault.html

[7] V. Sieh, O. Tschäche, and F. Balbach, "Verify : Evaluation of reliability using vhdl-models with embedded fault descriptions," in *FTCS '97 : Proceedings of the 27th International Symposium on Fault-Tolerant Computing (FTCS '97)*. Washington, DC, USA : IEEE Computer Society, 1997, pp. 32–36.

[8] H. R. Zarandi, S. G. Miremadi, and A. 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*. IEEE Computer Society, 2003, pp. 485–492.

[9] A. R. Ejlali, G. Miremadi, H. R. Zarandi, G. Asadi, and S. B. Sarmadi, "A hybrid fault injection approach based on simulation and emulation co-operation," in *DSN-2003 : Proceedings of the 2003 International Conference on Dependable Systems and Networks*. Washington, DC, USA : IEEE Computer Society, 2003, pp. 479–488.

[10] K. K. Goswami, R. K. Iyer, and L. Young, "Depend : A simulation-based environment for system level dependability analysis," *IEEE Trans. Comput.*, vol. 46, no. 1, pp. 60–74, 1997.

[11] J. Güthoff and V. Sieh, "Combining software-implemented and simulation-based fault injection into a single fault injection method," in *FTCS '95 : Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing*. Washington, DC, USA : IEEE Computer Society, 1995, pp. 196–206.

[12] D. Smith, B. Johnson, J. I. Profeta, and D. Bozzolo, "A method to determine equivalent fault classes for permanent and transient faults." Washington, DC, USA : IEEE Computer Society, 1995, pp. 418–424.

[13] W. Wang, K. S. Trivedi, B. V. Shah, and J. A. P. III, "The impact of fault expansion on the interval estimate for fault detection coverage," in *FTCS '94 : Proceedings of the 24th International Symposium on Fault-Tolerant Computing (FTCS '94)*. Austin, TX, USA : IEEE Computer Society, 1994, pp. 330–337.

[14] S.-M. Yen and M. Joye, "Checking before output may not be enough against fault-based cryptanalysis," *IEEE Trans. Comput.*, vol. 49, no. 9, pp. 967–970, 2000.