Integrated Evaluation Platform for Secured Devices

Pascal Manet, Jean-Baptiste Rigaud, Julien Francq, Marc Jeambrun, Bruno Robisson, Jérome Quartana, Selma Laabidi, Assia Tria

Integrated Evaluation Platform for Secured Devices

Pascal Manet, Jean-Baptiste Rigaud, Julien Francq, Marc Jeambrun,
Bruno Robisson, Jérôme Quartana, Selma Laabidi and Assia Tria
SESAM Laboratory (joint R&D team CEA-LETI/EMSE),
Centre Microélectronique de Provence
Avenue des Anémones, 13541 Gardanne, France
Email: {pascal.manet, bruno.robisson, assia.tria}@cea.fr, {rigaud, quartana, francq, jeambrun, laabidi}@emse.fr

***Abstract*— In this paper, we describe the structure of a FPGA smart card emulator. The aim of such an emulator is to improve the behaviour of the whole architecture when faults occur. Within this card, an embedded Advanced Encryption Standard (AES) protected against DFA is inserted as well as a fault injection block. We also present the microprocessor core which controls the whole card.**

Keywords. smart card, faults, DFA, countermeasure, hardware, FPGA, MIPS, AES.

I. Introduction

Considering the evaluation of the safety of secured objects with maximum relevance and to protect them with the greatest efficiency, BTRS project[1] (”Briques Technologiques pour le Renforcement de la Sécurité”) aims at having a platform to test a whole set of secured components. Safety being a permanent race between the attacker and the defender, the innovation in evaluation as in protection are the two axes of the project. The prime objective of the project thus consists in pre-empting the technological elements which will make it possible to the actors to build the most sharpened benches of evaluation.

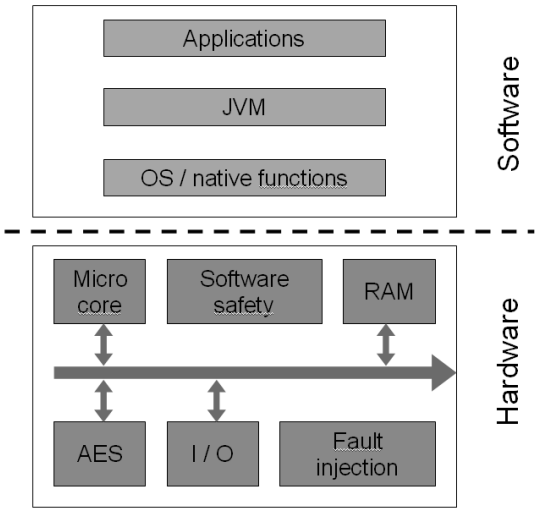

Fig. 1. Structure of the whole project

Within this project, a cryptosystem strengthened towards existing attacks has to be developped. We chose AES which replaced the DES algorithm as the standard for symmetric secret key encryption ([Na01]). Each physical implementation of AES must be secured, against side channel and fault-based cryptanalysis.

The injection of fault may enable an attacker to recover the secret information stored in the device by modifying its behaviour [SA02]. To test the robustness of the countermeasure we propose an injection fault block that will be developped to simulate a fault attack on the hardware device.

The evaluation of security on a smartcard deals with hardware and software. It is important to be able to test each level of attack and to strengthen the whole card because hardware attacks can modify the software behaviour.

In order to evaluate the behaviour of a entire card, this fault injection block may be expanded to simulate an attack anywhere. To control this fault injection and the whole system, a microprocessor core is developped.

The evaluation of security on a smartcard deals with hardware and software. It is important to be able to test each level of attack and to strengthen the whole card because hardware attacks can modify the software behaviour.

This paper is organized as follows. In Section II, we give a short overview of the main attacks and their associated countermeasures. The next section describes the whole card and the choices we made. In Section IV, we present our current work : our strengthened AES, the injection fault block and the miniMIPS with their implementation on a FPGA. We conclude in section V.

II. Attacks and Existing Countermeasures

Concerning hardware dedicated attacks on a crypto-processor, we studied AES. This algorithm is conjectured mathematically safe. Cryptanalytic attacks such as Square, Boomerang or Impossible Differentials defeat only reduced version of AES. On the contrary, side-channel analysis and fault-based cryptanalysis can be used to attack software or hardware implementations.

A. Side-channel attacks

Among the existing and developping SCA attacks, DPA is the most known and used.

[1]Powered by CIMPACA, in association with GEMPLUS and SPS

1) Differential Power Analysis: Differential power analysis was first presented in 1999 by Kocher et al. in [KJJ99]. It is a statistical analysis of power consumption traces taken from a cryptographic device as one of its inputs and determines the validity of a guess made on the cipher key. This attack relies on the assumption that a correlation exists between the device operation and the power consumed by the device while performing that operation. DPA is a powerful attack because it is non-invasive, it requires only an oscilloscope and computation means. Furthermore, it is independent of the algorithm implementation. The most common DPA attacks have been performed on the Data Encryption Standard (DES). A DPA attack on DES can be done on a plain or ciphertext with the associated power traces.
The principle is to make hypothesis on a bit value. Then, a statistical approach enables the attacker to choose between the two values. This attack works because there is a correlation between the physical measurements taken at different points during the computation and the internal state of the processing device. As AES has replaced DES, such an attack on AES can also be found in the litterature [OGOP04].

2) Countermeasures: There are three major types of countermeasures presented in the litterature.

a) Make internal results unavailable: This first approach prevents the prediction of intermediate results by using the duplication method ([GP99]). In [CJRR99], a masking method that splits into k parts each intermediate result of the cryptographic algorithm is presented. This increases the difficulty of obtaining the key through DPA exponentially with k. Messerges shows in [Mes00] that this method does not protect against higher order analysis.

b) Perturbate measurements: This can be done by introducing noise in the power measurements [KJJ99]. The noise added is usually uncorrelated to the data. This countermeasure forces the attacker to obtain more power traces to clean them. Messerges et al. [MDS02] present several techniques to improve the quality of the power trace. An other way is to add a temporal jitter to remove the spike if the correct key is guessed ([ABDM00]). With a learning step to understand how this desynchronization works, one can use classic DPA. Clavier, Coron and Dabbous ([CCD01]) propose the insertion of unused instruction to add information that is not correlated with data. However, they still propose a way to perform the attack.

c) Reduce signature: Shamir in [Sha00] proposed to add capacitors to isolate the target. As these capacitors are outside the cryptographic hardware, it is possible to cut them and perform DPA. Other techniques consist in reducing the current signature locally, that is directly on paths which transport the secret data. These approaches tend to balance the current consumption at the logical level (by using, for example, asynchronous logic and/or dual rail encoding [TMC$^+$02], [SMB05] and [BRR$^+$04]), at electrical level (by using differential logic, for example [TAV02] and [GHM$^+$04]) and even at layout level (by using ad hoc technics [TV04] and [GHMP05]).

B. Fault attacks

Fault-based cryptanalysis is a method that takes advantage of the misfunctionning of a device which may occur during the computation of a ciphertext [SA02]. Even before the standardization of AES, many different attacks based on faulty executions were proposed [BDL97], [BS97], [YJ00].

1) Differential Fault Analysis: Differential fault analysis (DFA), uses a set of correct and faulty ciphertexts to recover the key. The first DFA attack was proposed in 1997 by Biham and Shamir on DES [BS97] .

In [Gir05], Giraud proposed a method to retrieve bytewise the secret key of an AES by assuming that the attacker is able to switch only one bit between the last MixColumns and SubBytes. He also presented an attack which is more realistic. He assumes that several faulty bits can be injected in a single byte. This last method proceeds in three steps: first, attacking KeySchedule at the beginning of the last round (K9), then at the beginning of the ninth round (K8) and finally, the data path at the beginning of round 9.

Chen and Yen [CY03] presented an attack on the KeySchedule which uses the same fault propagation properties as the second attack of Giraud. It attacks only the KeySchedule. Three faulty executions are required: at the beginning of the last round then twice at the beginning of the ninth one. At last, the whole last Round Key can be retrieved: eleven bytes with the attack and the last five with an exhaustive search.

Dusart, Letourneux and Vivolo [DLV03] present an attack that uses the properties of MixColumns. The four bytes are linked by a relation that can be used to make hypothesis on the last Round Key. The whole last Round Key is retrieved in four steps. Each step needs its own faulty executions.

Piret and Quisquater [PQ03] present a very powerful attack based on the same idea as the attack described by Dusart and al. Only two pairs of good and faulty ciphertexts enable the attacker to find the whole 128-bit key. The fault has to occur on one or more bits of one byte between the MixColumns of rounds 7 and 8.

2) Countermeasures: For DFA-type attacks, various countermeasures have been presented. Chen and Yen([CY03]), proposed to counteract their own attack. Among their three countermeasures, they use twice the fact that the attacks need several executions of the KeySchedule. So, for example, computing only once the KeySchedule is an efficient solution. It should be noted that this technique is also effective against Giraud's second attack.

Bertoni and al. [BBKP02], [BBK$^+$02] present a very area-convenient countermeasure based on the use of a 4x4 parity bit matrix and a 4x4 error bit matrix which enables to stop the device or produce a wrong result if an odd number of faulty bits is purposedly induced. However, if this number is even, the Piret and Quisquater's method will enable to find the key. As it needs only two pairs of correct and faulty texts, one can't rely upon statistics to protect the circuit.

In a recent paper [MSY05], Malkin, Standaert and Yung show that countermeasures against DFA-type attacks which enable a rather good fault coverage present a cost close to duplication: Karpovsky and al. [KKT04] use a robust non-linear code and Karry and al. [KWMK02] decrypt on the

fly each round to ensure that none of them are faulty. This is sligthly better than what was proposed by Mitra and Mc-Cluskey [MM00], where efficient detection implementations would exceed the cost of duplication.

III. What our bench will be

Within the BTRS project, one objective is to study the behavior of made-safe objects, including smart cards, when they are attacked. To understand how faults operate and thus develop adapted countermeasures, we have to be able to control with a high precision how to inject them.

This precision level can be reached if we manage to run our simulation on a hardware platform. Indeed we could be able to inject faults on each desired bit.

The adopted solution is built on FPGA devices. A hardware emulation platform is being implemented to copy a smart card as close as possible. To be efficient, this system has to make all the following elementary components work together: microprocessor core, memories, cryptography block, serial communication protocol.

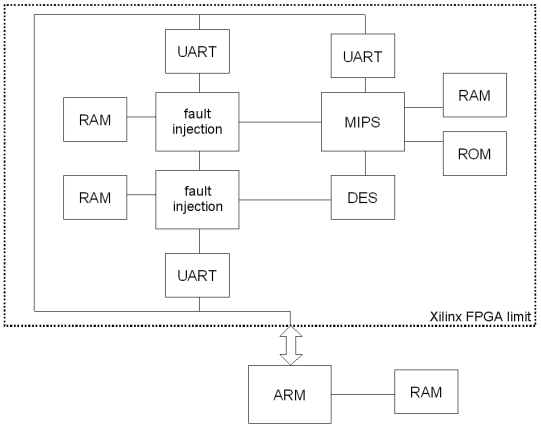

Fig. 2. Structure of the whole card

The question could be: why is it pertinent to inject simulated fault in FPGA while the software simulations are correct and the design rules are respected ? We only try to have real simulation results when a transient fault occurs on what software consider as a register that will not exist on a chip. Concerning AES for example, software simulated faults occur at every step of the round computation, but only one register hardwarely exists instead of four. Therefore, even if software simulation can take into account this type of event, hardware validation seems important.

This type of emulator also enables to test software behaviour in case of attack on the hardware that executes it.

The ability to run a large amount of faulty simulation on a hardware dedicated platform in a reduced time enables to validate the countermeasure in term of concept as well as in term of implementation.

IV. Current work

Our current work, before the global design of a test card, has three components. We first develop a countermeausre that improve the security of AES against DFA attempting not to have too bad performances in case of DPA attack, then we develop a fault-injection block to test the reliability of our countermeasures, finally we work on a MIPS core to control the whole card.

A. Secured AES

We used the opencores' website VHDL sources (www.opencores.org) as a seed to model our AES circuit. This implementation first converts the data and key, which are each made of four 32-bit words, to two 128-bit registers. Then, each round is computed within a clock cycle; KeySchedule is computed on the fly. At last, the 128-bit ciphertext is converted to four 32-bit words. Fig. 3 presents the top-level architecture of our AES.

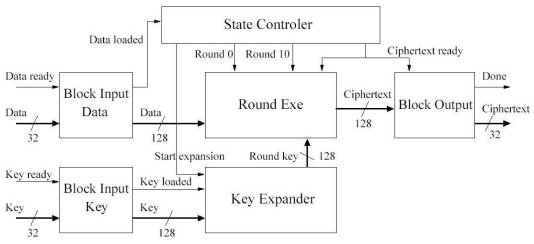

Fig. 3. AES architecture

1) Description: The general idea behind the proposed countermeasure is to enable the device to detect the error with a high fault coverage and then to perform modifications inside the chip to prevent any existing attack. The error detection information stay inside the AES computation without interaction with any state machine or controller.

If an error occurs, the circuit performs self-defence modifications so that the DFA-techniques could not be applied. More precisely, the aim of this countermeasure is to spread the error in a way that differs from the "normal" AES faulty execution. In our AES, any error which impacts only one byte of a state is spread over at least six other bytes of the state. Note that the proposed countermeasure will obviously not prevent a SEA-type attack.

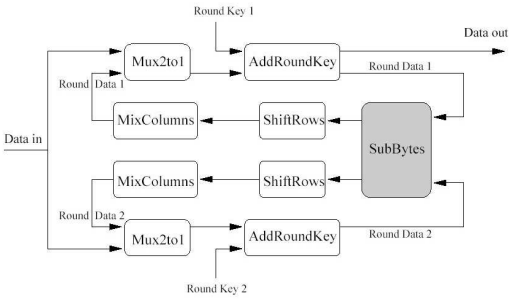

Fig. 4. Structure of the robust RoundExe

As shown in Fig. 4, the data path and KeySchedule are duplicated so that the encryption is performed twice in parallel.

The countermeasure takes place in the block "SubBytes". It is decomposed in four steps. Fig. 5 shows its architecture.

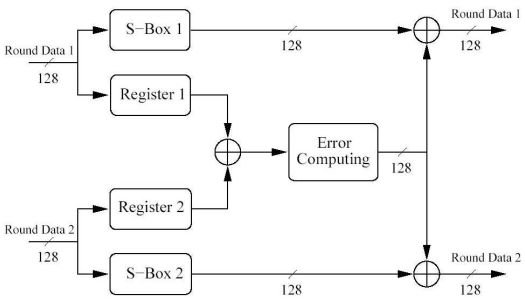

Fig. 5. Robust SubBytes

First, duplicated data states Round Data 1 and Round Data 2 are respectively latched in Register 1 and Register 2. These states are compared by using a bitwise XOR. The result is used as inputs for the Error Computing Block (Fig. 6).

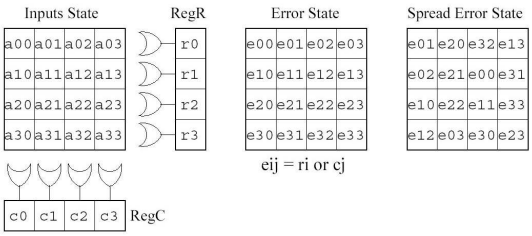

Fig. 6. Error Computing

Second, the 4 bytes of each row and each column of this comparaison are added together by using a bytewise OR. We obtain 2 registers of 4 bytes (RegC, RegR). We swap the bits of all RegC bytes. The most significant bit becomes the least one etc.

Third, we combine, two by two with one OR gate, each byte of RegC with each byte of RegR in order to obtain the Error state (Fig. 6). This state is scrambled so that each byte has a new position in the state.

At last, this Spread Error State is added to both S-Boxes output data. These data give the outputs of the SubBytes block.

2) Fault coverage: The detection method has a 100% fault coverage on single and multiple event upset occurring on one byte. Assuming we consider a multiple byte attack, the row and column sums may be zero and the circuit would not spread this kind of error. The probability to have one sum equal to zero is 2^{-8} with a random error hypothesis. As all sums have to be zero, the probability of a non-spread error is 2^{-64} in the case of a multiple-byte attack. Even in such a case there is no existing attack with multiple faulty bytes.

The fact that the error is spread on other bytes prevents the attacker from finding the key. The link of the data path is broken and one can't compute the key by using DFA-techniques. Giraud's and Yen and Chen's attacks will not be able to retrieve the link to the error injected in the KeySchedule. Dusart and al.'s as well as Piret and Quisquater's attacks use the link provided by the MixColumns and it is also modified.

Concerning DPA resistance, the second data-path processes inverted data. A 0 on one side always corresponds to a 1 on the other one. We try to equilibrate the data path at a gate level. We used modified SubBytes and MixColumns, and $xnor$ instead of xor to add the inverted roundkey.

B. Fault injection

To test the proposed countermeasures and others, we have to inject faults within the AES.

The "fault generator" can inject three different types of fault chosen by the user: 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). The choice of these faults is not harmless. The "bit-flip" model is the most used in the litterature. The "stuck-at" models are very convenient, due to the fact they are representative of numerous kinds of physical failures ([ABB+04]). It can be used to simulate a ligth 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.

1) Description of the "fault generator": Obviously, the insertion of this block must have a limited impact on the execution time of our initial system.

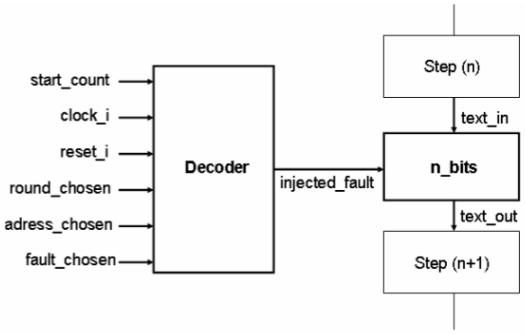

Fig. 7. Structure of the "fault generator"

Such a block enables to choose the address of corrupted bits, the kind of faults and the moment of injection. The "fault generator" is built with two blocks called "n_bits" and "decoder" (see Fig. 7).

The result computed by the decoder, called "injected_fault", is sent to one of the inputs of the block "n_bits". 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 Fig. 8). 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.

Other works on the subject may be found for example in [ALV05].

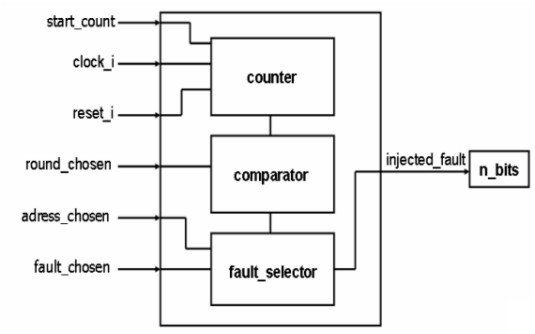

Fig. 8. Structure of the "decoder"

2) Fault injection on AES: In order to inject some faults at the right round and at the right transformation of the AES, 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 KeySchedule). 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 Fig. 9).

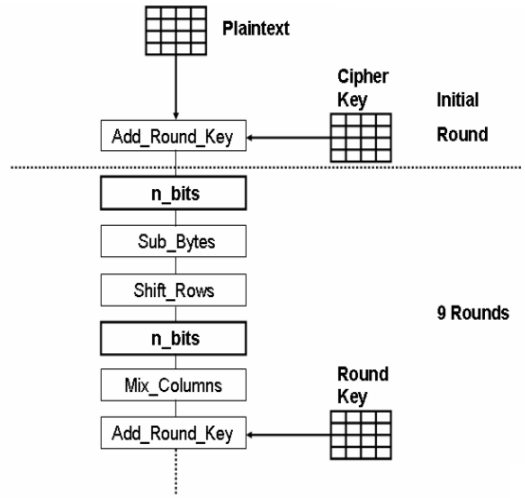

Fig. 9. Insertion of the block "n_bits" in AES

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 KeySchedule 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. 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 state 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.

C. Microprocessor core

The first challenge was to find a reliable and debugged model for the microprocessor core that fits our needs. We chose to work with miniMIPS core [2]. It consists in an open source model of a 32 bits RISC microprocessor based on MIPS-I instruction set. The version we use was provided by TIMA Laboratory[3].

Using the adapted design flow tools (ModelSim for simulation and ISE for synthesis on FPGA), the miniMIPS model was ported on the VirtexE. We took advantage of the basic I/O components on the card (push-button, switches, LEDs) to design a demonstrator. Once inputs are defined by the eight switches on the card, the user uses the push-button to apply a reset signal on the miniMIPS. Then instructions trigger the microprocessor to perform an addition of the two four-bit inputs. To do this, 32-bit instructions extracted from the MIPS instruction set are pre-programmed (using VHDL code) on the FPGA and sent sequentially to the miniMIPS.

Memory components are currently being added and will communicate with the miniMIPS core. The program stored in the ROM will be executed by the miniMIPS and data will be managed in read/write mode in the RAM. Adding the secured AES code, we will be able, using previous communication protocol, to run cryptographic simulation campaigns and to retrieve results.

Then the complete platform on the FPGA will be the subject of a complete validation process. After that, it will be ready to accept the fault injection block described above and test the robustness of our countermeasures.

D. Implementation

Given the FPGA cards available for the project, the whole smart card emulation will be implemented on a XCV2000E+ Logic Module from Xilinx, $Virtex - E^{\mathrm{TM}}$ family. This FPGA was embedded on a Integrator/LM XCV600E+ from ARM ltd. This card was chosen also for its properties : it can be easily connected on a larger platform to extend the number of peripherals it can be linked to. So ARM processors, communication ports, ethernet ports can be reached by that way. The design was made 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.

1) AES: We kept the opencores architecture of the AES but we used our own transformation components. The AES component contains a State Controler, Key Expander and Round Execution and also input and output interfaces. Key Expander is modified in order to perform twice the calculation. Round Execution computes two data paths in parallel using ShiftRows, MixColumns and AddRoundKey and a modified SubBytes component that inputs and outputs the two data paths' states.

Each round is computed within a clock cycle. The S-Boxes are implemented by dual-port RAM blocks. We added registers

[2]based on a students project from ENSERG school

[3]http://tima.imag.fr

Derived from their MDR project developped on an ALTERA FPGA card

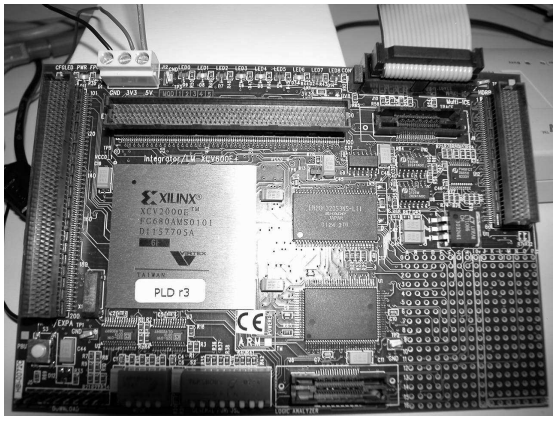

Fig. 10. View of the demonstrator card

to use these RAM inputs and compute the countermeasure. The input data are combined to create the error state.

We first implemented a non-robust AES on the 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 familly FPGA. This basic element is composed of a four variable logic function generator, a carry logic and a memory element. The unprotected AES also uses ten dual-port RAM blocks: eight for the sixteen data-path S-Boxes and two for the four of the KeySchedule.

We tried then to only duplicate the data and key paths without comparison. It had a size of 1600 slices, used twenty RAM blocks and still worked at 60 MHz.

The robust version of our AES oversizes the duplication by a 20% factor (1910 slices) and decreases the speed from 60 to 50 MHz. The number of RAM blocks used is unchanged.

2) Fault-generator: After synthesis, the design takes 1150 more slices, 800 more slice flip flops and 1500 more 4-input LUTs than the unprotected AES. Area was not a critical criterion and represents less than 10% of the total amount of available space on the device. The speed requirement is matched, only two gates were added in the critical path and the required clock frequency is unchanged.

3) Microprocessor core: Some resulting figures are listed hereafter: the design takes 1% of the total amount of available slice flip flops (494 out of 38400) and 4% of 4-input LUTS (1650 out of 38400). It uses an equivalent count of 15094 gates. Moreover, with the chosen synthesis effort, the maximum frequency is 36 MHz.

V. Conclusion

The BTRS project is on the long run to increase the level of security of robust block. To study the behavior of objects that have to be safe, including smart cards, under fault attacks, we need to understand how these attacks work. With a hardware dedicated platform, we can then develop adapted countermeasures.

The adopted solution is built on FPGA devices. A hardware simulation platform is being implemented to emulate a smart card. This system includes a microprocessor core, AES cryptography block and a fault injection block, but memories and serial communication protocol are to be developped too.

Concerning AES, we present a countermeasure which defeats all known DFA attacks on AES. This countermeasure consists in detecting the error by duplicating the key and data paths and spreading an error all over the computation process in order to break the link between the pairs of correct and faulty ciphertexts and the secret key. We also note that the detection information is kept within the data path and used to corrupt it. This countermeasure is designed in order to produce a limited DPA signature. To test this countermeasure, a fault injection block is currently under developpement as well as a microprocessor core to control the whole system.

References

[Na01] National Institute of Standards and Technology (NIST). Announcing the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication, n. 197, November 26, 2001. .

[ABB+04] Florence Azaïs, Serge Bernard, Yves Bertrand, Marie-Lise Flottes, Serge Pravossoudovitch, Christian Landrault, Patrick Girard Michel Renovell, Laurent Latorre, and Bruno Rouzeyre. *Test de Circuits et de Systèmes Intégrés*. Hermes, 2004.

[ABDM00] Mehdi-Laurent Akkar, Régis Bevan, Paul Dischamp, and Didier Moyart. Power analysis, what is now possible... In *ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 489–502, London, UK, 2000. Springer-Verlag.

[ALV05] Lorena Anghel, Régis Leveugle, and Pierre Vanhauwaert. Evaluation of set and seu effects at multiple abstraction levels. In *IEEE International On-Line Testing Symposium (IOLT)*, 2005.

[BBK+02] Guido Bertoni, Luca Breveglieri, Israel Koren, Paolo Maistri, and Vincenzo Piuri. On the propagation of faults and their detection in a hardware implementation of the Advanced Encryption Standard. In *13th IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP 2002)*, pages 303–312. IEEE Computer Society, 2002.

[BBKP02] Guido Bertoni, Luca Breveglieri, Israel Koren, and Vincenzo Piuri. Fault detection in the Advanced Encryption Standard. In *Proceedings of MPCS 2002*, Ischia, Italy, 2002.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

[BRR+04] G. Bouesse, Marc Renaudin, Bruno Robisson, Edith Beigné, Pierre-Yvan Liardet, S. Prevosto, and J. Sonzogni. DPA on quasi delay insensitive asynchronous circuits: Concrete. results. In *XIX Conference on Designof Circuits and Integrated Systems*, 2004.

[BS97] 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.

[CCD01] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. *Lecture Notes in Computer Science*, 1965:252–??, 2001.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 398–412, London, UK, 1999. Springer-Verlag.

[CY03] Chien-Ning Chen and Sung-Ming Yen. Differential fault analysis on AES key schedule and some countermeasures. In R. Safavi-Naini and J. Seberry, editors, *Information Security and Privacy — ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2003.

[DLV03] Pierre Dusart, Gilles Letourneux, and Olivier Vivilo. Differential fault analysis on A.E.S. In J. Zhou, M. Yung, and Y. Han, editors, *Applied Cryptography and Network Security – ACNS 2003*, volume 2846 of *Lecture Notes in Computer Science*, pages 293–306. Springer, 2003.

[GHM+04] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, Renaud Pacalet, and Jean Provost. Cmos structures suitable for secured hardware. In *DATE*, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[GHMP05] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, and Renaud Pacalet. The "backend duplication" method. In *CHES*, pages 383–397, 2005.

[Gir05] Christophe Giraud. DFA on AES. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *Advanced Encryption Standard – AES*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2005.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In *Cryptographic Hardware and Embedded Systems – CHES 1999*, pages 158–172, London, UK, 1999. Springer-Verlag.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 388–397, London, UK, 1999. Springer-Verlag.

[KKT04] Mark G. Karpovsky, Konrad J. Kulikowski, and Alexander Taubin. Robust protection against fault injection attacks on smart cards implementing the Advanced Encryption Standard. In *2004 International Conference on Dependable Systems and Networks (DSN 2004)*, pages 93–101. IEEE Computer Society, 2004.

[KWMK02] Ramesh Karri, Kaijie Wu, Piyush Mishra, and Yongkook Kim. Concurrent error detection scheme for fault-based side-channel cryptanalysis of symmetric block ciphers. *IEEE Transactions on Computer-Aided Design*, 21(12):1509–1517, 2002.

[MDS02] Thomas S. Messerges, Ezzat A. Dabbish, and Robert H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.*, 51(5):541–552, 2002.

[Mes00] Thomas S. Messerges. Using second-order power analysis to attack dpa resistant software. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, pages 238–251, London, UK, 2000. Springer-Verlag.

[MM00] Subhasish Mitra and Edward J. McCluskey. Which concurrent error detection scheme to choose. In *IEEE International Test Conference 2000*, Lecture Notes in Computer Science, pages 985–994. IEEE Computer Society, 2000.

[MSY05] Tal G. Malkin, François-Xavier Standaert, and Moti Yung. A comparative cost/security analysis of fault attack countermeasures. In *Second Workshop on Fault Detection and Tolerance in Cryptography (FDTC 2005)*, pages 109–123, Edinburgh, UK, September 2, 2005.

[OGOP04] Siddika Berna Ors, Frank Gürkaynak, Elisabeth Oswald, and Bart Preneel. Power-analysis attack on an asic aes implementation. In *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2*, page 546, Washington, DC, USA, 2004. IEEE Computer Society.

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

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 2002.

[Sha00] Adi Shamir. Protecting smart cards from passive power analysis with detached power supplies. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, pages 71–77, London, UK, 2000. Springer-Verlag.

[SMB05] Design and analysis of dual-rail circuits for security applications. *IEEE Trans. Comput.*, 54(4):449–460, 2005. Danil Sokolov and Julian Murphy and Alexander Bystrov and Alex Yakovlev.

[TAV02] K. Tiri, M. Akmal, and I. Verbauwhede. A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards. In *29 European Solid-State Circuits Conference (ESSCIRC 2002)*, 2002.

[TMC+02] George Taylor, Simon Moore, Paul Cunningham, Ross Anderson, and Robert Mullins. Improving smart card security using self-timed circuits. *async*, 00:211, 2002.

[TV04] K. Tiri and I. Verbauwhede. Place and route for secure standard. In *CARDIS'04*, 2004.

[YJ00] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Transactions on Computers*, 49(9):967–970, 2000.