J. Cryptol. (2011) 24: 247–268
DOI: 10.1007/s00145-010-9083-9

Journal of
CRYPTOLOGY

Glitch and Laser Fault Attacks onto a Secure AES Implementation on a SRAM-Based FPGA

G. Canivet
TIMA Laboratory (Grenoble INP, UJF, CNRS), 46 av. Félix Viallet, 38031 Grenoble, France
Gaetan.Canivet@imag.fr
and
CESTI/CEA-LETI, Minatec, 17 Avenue des Martyrs, 38054 Grenoble Cedex 9, France

P. Maistri and R. Leveugle
TIMA Laboratory (Grenoble INP, UJF, CNRS), 46 av. Félix Viallet, 38031 Grenoble, France
Paolo.Maistri@imag.fr; Regis.Leveugle@imag.fr

J. Clédière
CESTI/CEA-LETI, Minatec, 17 Avenue des Martyrs, 38054 Grenoble Cedex 9, France
Jessy.Clediere@cea.fr

F. Valette
DGA/CELAR, 35171 Bruz Cedex, France
Frederic.Valette@dga.defense.gouv.fr

M. Renaudin
Tiempo, 110 Rue Blaise Pascal, 38330 Montbonnot Saint Martin, France
Marc.Renaudin@tiempo-ic.com

Received 1 September 2009
Online publication 26 October 2010

Abstract. Programmable devices are an interesting alternative when implementing embedded systems on a low-volume scale. In particular, the affordability and the versatility of SRAM-based FPGAs make them attractive with respect to ASIC implementations. FPGAs have thus been used extensively and successfully in many fields, such as implementing cryptographic accelerators. Hardware implementations, however, must be protected against malicious attacks, e.g. those based on fault injections. Protections have been usually evaluated on ASICs, but FPGAs can be vulnerable as well. This work presents thus fault injection attacks against a secured AES architecture implemented on a SRAM-based FPGA. The errors are injected during the computation by means of voltage glitches and laser attacks. To our knowledge, this is one of the first works dealing with dynamic laser fault injections. We show that fault attacks on SRAM-based FPGAs may behave differently with respect to attacks against ASIC, and they need therefore to be addressed by specific countermeasures, that are also discussed in this paper. In addition, we discuss the different effects obtained by the two types of attacks.

Key words. AES, SRAM-based FPGA, Power glitch, Laser fault injections, DDR.

1. Introduction

In our modern society, embedded devices often contain confidential or critical information that needs to be protected from unauthorized use. Secure devices are often made in ASICs, but these kinds of devices are limited to high production volumes. Another solution consists of using programmable devices like SRAM-based FPGAs, thanks to their low cost and high flexibility. In recent times, FPGAs have been reported many times as a valuable mean for cryptographic implementations.

The security of the implemented cryptosystem can be compromised by cryptanalytic attacks aiming at specific weaknesses of the algorithms and protocols that are used. However, thanks to the fact that most cryptographic standards must now stand a large scrutiny effort before public acceptance, this possibility is usually unlikely. On the other hand, attacks against the actual implementations can take advantage of the characteristic of the device itself, such as the power consumption [1], electromagnetic emission [2–4], or susceptibility to computation errors [5].

Fault-based attacks [6], in particular, can be very powerful. Usually, differential cryptanalysis can recover the key only from block ciphers with a reduced number of rounds; power analysis attacks can break a regular implementation after a few thousands encryptions, depending on the implementation and on the noise level of the measurements. Fault attacks, on the other hand, can break an implementation just after a few encryptions [7]. They exploit the same principles of differential cryptanalysis: in fact, they can attack full-sized ciphers by injecting the differential error in the latest rounds of the encryption process. Even if most works target transient faults, research has shown that even permanent faults can be exploited to mount an attack [8].

Faults can be injected in different ways, but usually the attacker aims at controlling the fault injection as much as possible. For this reason, the most used techniques are based on power glitches, or laser injections. They allow fine tuning of the injection time and, in the case of lasers, also a good precision in choosing the attack target.

Most effort, so far, has been dedicated to attacking ASIC implementations, due to their large diffusion and to the relative ease of altering the behavior of the system in a non-permanent way. In [9], for instance, the authors have attacked an AES implementation on a smart card: errors are injected by progressively reducing the voltage supplied to the core, until timing errors are introduced. The same approach has been applied successfully to FPGA implementations [10]; on the other hand WDDL proved to be robust against such fault injection technique [11]. Laser attacks may be much more powerful but at a higher cost. On the other hand, it is well known that optical-based attacks are possible even with much cheaper equipment [12], and EM-based fault injections are starting to be considered as well [13].

To our knowledge, there are few works dealing with fault attacks against SRAM-based FPGA implementations. This is probably due to the fact that such devices may be much more sensitive to perturbations than ASICs. This leads to different error models in presence of fault attacks, which make cryptanalysis more difficult. Nonetheless, attacks may still be possible. So far, previous works have mainly addressed the characterization of the behavior of a programmable board in harsh environments, in particular for spatial applications [14]. A few works have dealt with the analysis of the global effects of a laser-based attack, either with multiple shots and error accumulations [15] or single shots [16]. No paper presents attacks against a cryptographic design implemented on

FPGA, except [17] that introduced the dynamic fault injection during an encryption with the Data Encryption Standard; however, no analysis of the ciphering error is done.

This paper is one of the first works describing glitch- and laser-based attacks against a secured cryptographic implementation onto a SRAM-based FPGA. Unlike glitch attacks, we show that laser fault injections pose a serious threat to FPGA implementations, since they may alter the configuration of the FPGA in a semi-permanent way. Many error detection schemes developed against fault attacks are designed considering only transient faults, which is the predominant model in the literature. However, when the laser modifies the configuration of the device it may also change the function that is computed; due to the SRAM technology, the modification remains until the device is reconfigured. We hence propose an improved version of our error detecting implementation, which is able to sustain also laser-based fault attacks on the FPGA.

The paper is organized as follows. The next section briefly resumes the basics of the Advanced Encryption Standard and describes the architecture we attacked. The architecture was first validated against emulated fault injections: the results are described in Sect. 3. Then, we implemented and attacked an actual FPGA implementation: the board is described in Sect. 4; Sects. 5 and 6 present power-based and laser-based fault injections. Given the experimental results, we addressed the specific flaws and improved the protection scheme, which was again validated with both emulated and laser-based fault injections: this is presented in Sect. 7. Finally, Sect. 8 concludes this paper.

2. Architecture

2.1. *The Algorithm*

The architecture attacked in this paper implements the Advanced Encryption Standard (AES). AES [18] is a symmetric block cipher, standardized by NIST in 2001 as a substitute to the old and flaky DES cipher. It is based on a substitution-permutation network and it can encrypt (or decrypt) 128-bit inputs by using 128-, 192-, or 256-bit keys.

The cipher has an iterative structure and the number of rounds depends on the length of the key: 10 rounds for 128-bit keys, whereas 12 and 14 rounds are used for 192- and 256-bit keys, respectively. Each round is made of four different operations:

1. *SubBytes*. It is a non-linear byte substitution based on the multiplicative inversion in binary finite fields. It can be implemented as a lookup table (in ROM or synthesized as a multilevel combinational network) or as a functional block over composite fields following the algebraic definition. The latter implementation is smaller and it allows being easily pipelined, thus higher frequencies can be easily reached at a small cost.
2. *ShiftRows*. This operation is a row-wise permutation: rows are rotated by 0, 1, 2, or 3 bytes, depending on the row index.
3. *MixColumns*. It is a multiplicative scaling of each column with fixed coefficients in a binary polynomial field.
4. *AddRoundKey*. The modulo-2 addition with a round-dependent key.

The operations are generally byte-oriented, although some operate on wider elements. For sake of symmetry, which simplifies the implementation of architectures supporting both encryption and decryption, the first round is preceded by an initial key addition; conversely, the last round lacks the *MixColumns* operation.

The decryption process is obviously made of the inverse operations in reverse order. An additional process is dedicated to generate the key material (i.e., all the round keys) from the initial secret key. This process is often referred to as the key scheduler, and it is based on some operations used also in the encryption process, which allows sharing some functional blocks. For further details, the reader is invited to refer to the original specifications [18].

2.2. *The Implementation*

The architecture implemented and attacked in this paper is equipped with an error detection mechanism based on temporal redundancy. The basic principle is based on repeating the computation twice (or more) and then comparing the results. If any of the outcomes is different from the others, it is conservative to state that at least one error has occurred. This is a well-known approach and it allows detecting all transient faults that do not modify the state of the circuit in a permanent way.

Temporal redundancy can be applied at different levels and with different techniques. For instance, it can be easily adopted at application level, forcing the application to run twice in order to compare the results. At the hardware level, two interesting applications of detection schemes based on temporal redundancy have been proposed: pipeline [19] and DDR [20] redundancy. The architecture implemented in this paper uses the latter. The DDR technique is based on the exploitation of both clock edges to perform computation and to control the memory elements. A suitable subset of the register space is partitioned in order to identify those flip-flops that will store data on the rising edge of clock, and those switching on the falling edge. One register from the "positive" data path and one from the "negative" can be coupled together to form a single DDR register: the combinational logic implementing the functional operations can be thus shared through the connection to the DDR register.

Under proper conditions, this approach allows processing twice the data for each clock cycle, thus halving the computation time (in terms of clock cycles). The spare time can be used to reissue the same computation and then check the results. However, redundancy requires that selected DDR registers are also duplicated in order to store and compare both the primary and the backup computations. Not all the registers need to be duplicated: intermediate results that are in the pipeline still need to be stored into a DDR register (since two values are computed for each clock cycle), but they do not need to be saved for later verification and hence the backup register is not needed.

The architecture which has been protected with the DDR scheme is made of three functional blocks:

- the main data path, which is based on an implementation presented in [21] and equipped with four substitution boxes and 16 functional blocks implementing the linear operations and containing the state registers;
- the key scheduler, which shares the substitution boxes with the encryption data path;
- and the controller, which uses two finite state machines (FSM), one for each clock edge, synchronized with each other.

Only the main data path works in DDR mode, since most fault attack models target the temporary state of the encryption. The registers are partitioned and grouped into

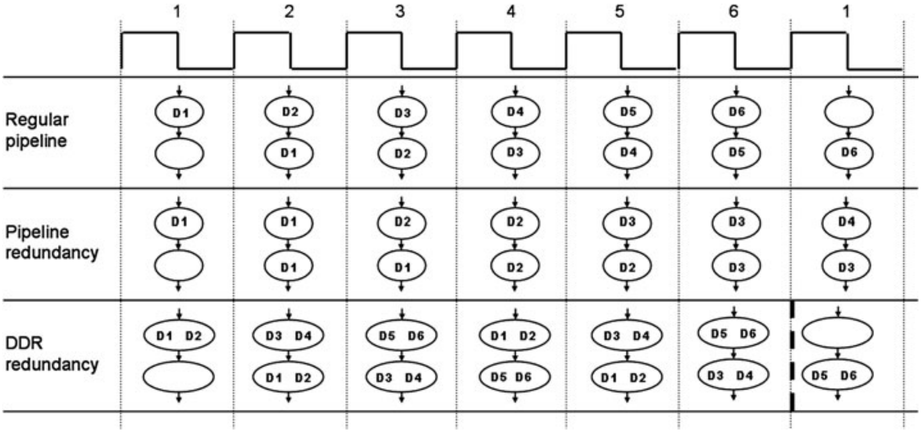


Fig. 1. Comparison of different techniques of temporal redundancy. The symbol *Dx* identifies subsequent data tokens processed through the pipeline.

DDR registers; then, each DDR register storing the internal state is accompanied by an auxiliary backup element. There is no need for backup registers in the S-Boxes, since they process only temporary values. The key scheduler does not use the DDR template in the implemented version and it is not protected against faults; adding protection in this part of the circuit would be straightforward but it was not necessary for this study. The controller, on the other hand, is protected with dedicated techniques: validation of the state encoding, verification of the state transitions for both FSMs, duplication of critical elements (e.g., the iteration counter) [20].

The DDR scheme is exemplified and compared to the regular pipeline and to the pipeline redundancy scheme in Fig. 1. Although pipeline redundancy can be adopted in an existing design flow more easily, it is more vulnerable to faults lasting longer than one clock cycle. If a fault lasts two cycles, in fact, it may affect both the main computation and its repetition, and there will be no way to detect that an error has occurred. DDR redundancy, on the other hand, computes the same operations in cycles that are far apart: an attacker must thus be able to alter both the primary and the secondary execution in the same way in order to inject an undetected error. In principle, such approach based on distancing the computations might be applied also to the implementations based on pipeline redundancy: in this case, however, the interventions made to adapt the design would void the main advantage of the pipeline approach, which is its easy integration and adaptability to the traditional design flow.

We will show the detection capabilities of the DDR technique in the following sections.

3. Emulated Fault Attacks

The robustness of the implemented design against fault attacks has been initially evaluated through emulated fault injections. This approach gives significant advantages over simulated faults: although it requires a synthesizable description of the system, on the other hand it ensures that the model is quite realistic, and moreover the hardware acceleration gives very large speedup of the experimental campaigns. We used a dedicated tool to instrument the netlist of the architecture: this automated approach allowed simple control over the flip-flops chosen as a target for fault injections. The hardware acceleration speeded up the experimental campaign significantly, which was the major benefit from the emulated injections. The tool allowed targeting only upset faults, which means that we were able to alter the content of any flip-flop in the architecture at a specific clock cycle; on the other hand, we could not inject transient faults (SET) into the combinational logic. This is not a major issue: this work is aimed at fault attacks (few bit errors concentrated in a single register) and not at analyzing the dynamics of a SET propagation tree.

Our setup is based on a Virtex-II Pro board in order to take advantage of the embedded processor. Thus, we can distribute the injection tasks at several levels: from the hardware blocks, to the embedded processor, and to the host PC used to configure the board itself. This approach allows great flexibility and it allows partitioning the computation and communication tasks as it is most convenient. To enable fault emulation and access inner memory cells, the original circuit was instrumented: thus, selected memory elements of the design could be fully controlled and observed during the computation process. Unlike scan chains, the computation did not have to be stopped in order to interact with the internal state of the circuit.

The AES algorithm is very regular. This means that the relative error propagation pattern does not depend on the absolute location of the fault. This observation is fundamental, because it allows reducing the size of the test campaign without affecting the validity of the results significantly. In particular, in the reported experiments, only one substitution box (S-Box) and one linear computing element were instrumented; then, every possible error value was verified for each target, for each computation cycle, and for errors lasting up to nine clock cycles. We performed a large number of emulated fault injections: as term of reference, the number of faults injected into the S-Box was about 10^9.

To characterize the effects of fault injections, we classified the outcome of the experiments into five different classes. These classes are:

1. *No effect*: the internal state of the circuit was not modified at all. Class 1 means that the injection was unsuccessful and it can be applied only to real experiments (reported hereafter), since during the emulation campaign an error is always injected.
2. *Silent errors*: the state of the circuit (either configuration or data) is modified, but the result is correct and no error detected. This may occur when altering logic or values that are not used at that stage of the computation process.
3. *False positives*: there is actually an alteration, and an alarm is raised, but the result is nonetheless correct.
4. *Detected errors*: the fault led to an unexpected result and it was successfully detected.
5. *Undetected errors*: no error is detected, but the cipher is not the expected value. This is the most dangerous situation, since it may lead to a security breach.

The detailed report on the campaign results can be found in [20]. Here we want to highlight the behavior of the DDR design against transient faults in the user memory

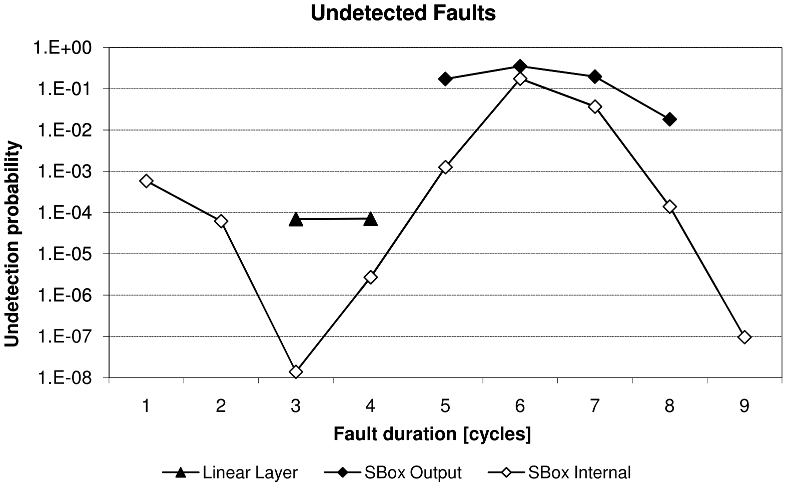


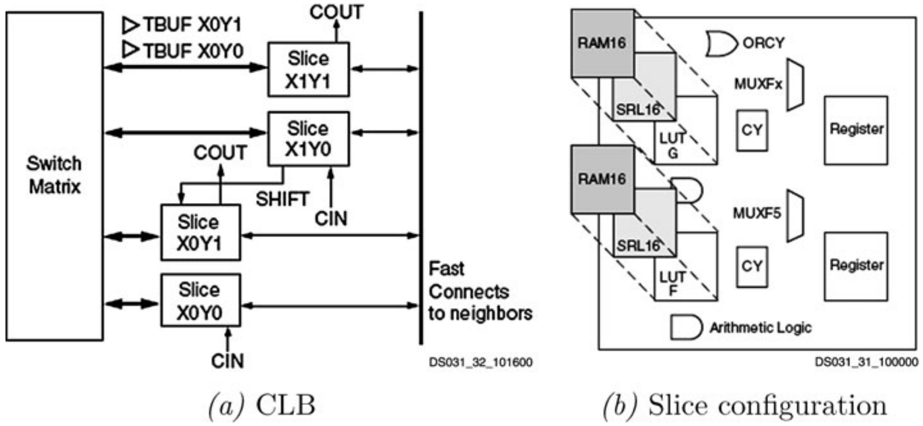
Fig. 2. Undetected transient faults with emulated injections [20].

elements of an ASIC implementation, which are the typical model for fault attacks. As already anticipated, the results from the emulated campaigns may belong only to classes 2 to 5: since we can control finely the content of each register, an error was actually introduced for all the experiments. This may not hold for the real experimental campaign, where we need to deal also with the success rate of the injection technique.

This protection scheme has a very good coverage rate for short or very short faults targeting the flip-flops (see Fig. 2): when the fault is shorter than a complete round computation (i.e., the regular and the backup computation), then errors are detected with a probability of 99.9%. As the length of the fault increases, it is more likely that it will not be detected by the protection scheme, which is based on temporal redundancy, since both computations will be probably affected by the same error. The confirmation comes from the fact that the highest probability of an undetected error is for faults lasting six clock cycles, which is exactly the duration of a round. Such a fault duration is however quite unlikely when attacking an ASIC.

4. The FPGA Board

The FPGA used in this paper is a Xilinx Virtex-II XC2V1000, fabricated on a 0.15 μm CMOS 8-layer metal process. The FPGA is embedded in an 896-pin flip-chip fine-pitch package. Physically, there are 720 Kbit block RAMs distributed on four columns with multipliers and 432 available I/Os placed on all the surrounding of the chip. The device is configured by downloading a configuration file (bitstream) that contains all the configuration information. All user programmable features inside the Virtex-II device are controlled by memory cells that are volatile and must be configured on power-up. These memory cells are known as the configuration memory and define the Look-Up Table (LUT) equations, signal routing, Input/Output Blocks (IOBs) voltage standards, and all other aspects of the user design.



(*a*) CLB (*b*) Slice configuration

Fig. 3. Virtex-II Elements [22].

Virtex-II configuration memory is arranged in vertical frames that are one bit wide. The length of the frame depends on the size of the device. For the XC2V1000 used during our campaigns, the bitstream is composed of 1104 frames of 106 32-bit words. Configuration frames are grouped into six column types that correspond to physical device resources. Each Virtex-II device has the same configuration column types: IOBs, Input/Output Interconnects (IOIs), Configurable Logic Blocks (CLBs), Global Clocks (GCLKs), BlockRam (BRAM) and BlockRam Interconnects.

The major elements of the Virtex-II independently of the device type are CLBs, which are composed of four identical slices (logical element) and of interconnections (Fig. 3(a)).

As shown in Fig. 3(b), each slice is composed of a sequential part (Flip Flops) and a logical part (LUTs and multiplexers). Each 4-input function generator is programmable as a 4-input LUT, 16 bits of distributed SelectRAM memory, or a 16-bit shift register element.

Each Virtex-II device can be represented as an array of logic blocks surrounded by switch matrices. Long lines are bidirectional wires distributing the signals across the device; Hex lines route signals to every third and sixth block in all directions; finally, double lines route signals to every first or second block in all directions.

The device used has three kinds of power supplies: the core (1.5 volts), the input–output, and the auxiliary (3.3 V). The input–output supply allows defining the voltage standard used by the design (LVTTL, LVCMOS, LVDS), whereas the auxiliary supply is used for example to power the JTAG module.

5. Power Glitch Attacks

5.1. *Experimental Setup*

During the fault injections by means of power glitches, the voltage of the power supply is increased by the attacker for a brief instant. For our campaigns, we used several

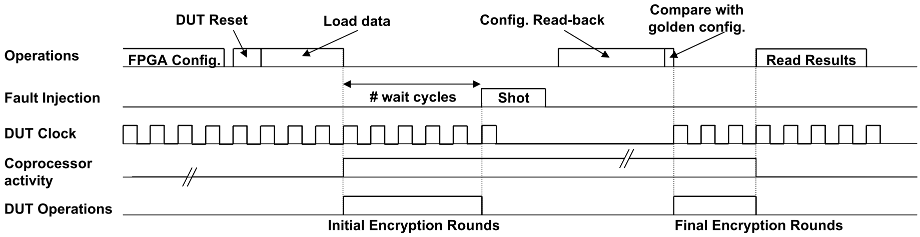


Fig. 4. Timing diagram of the test sequence used for each spatial position.

values for the width and the amplitude of the pulse. The first series of experiments consisted in applying overshoots between 5 and 80 volts on the rising edge of the clock. In order to reduce the time required by the whole experimental campaign, we decided to split the whole campaign into 2 sessions with different voltage ranges. The first session envisioned overshoots ranging from 5 to 45 volts: we will refer to this campaign as S1-1; the other session ranged from 45 to 80 volts (referred to as S1-2). A third set of experiments was also conducted, where the overshoots range again between 45 and 80 volts (as for campaign S1-2), but are applied on falling edge of the clock (this will be called S2). The voltage incremental step is the same for all campaigns and set to 5 volts. For all the campaigns, the power glitches were applied for durations ranging from 10 ns to 100 ns by steps of 10 ns. We targeted the cycles 49 to 60, corresponding to the two last ciphering rounds.

For these campaigns a specific electronic board was developed, with separated power supplies (core, IOs, and auxiliary). Thanks to this approach, it was possible to apply the overshoots only to the chosen power supply lines: in our experiments, the power glitches were applied only to the core voltage. If glitches had been applied to other power lines (auxiliary and/or IOs), then the effects would have occurred only during the download of the bitstream or during the data transmission (key, data, go ciphering ...), without modifying the data during the encryption process.

Faults were injected during the ciphering following the methodology depicted in Fig. 4. The FPGA configuration is first sent through a serial interface (JTAG), followed by data and commands that are needed by the system (e.g., reset, secret key and plain text, the instant of fault injection, and the start commands). According to the commands and parameters that have been received, a motherboard sends all the proper signals to a daughter board with the Device Under Test (DUT), including the clock and the data. Once the encryption has started, the motherboard counts the rising edges of the clock in order to stop the DUT clock at the right instant. The fault is injected at the last rising (or falling) edge before stopping the clock. Since the clock is stopped, we can tune the duration of the fault injection according to our needs: in fact, the DUT is not working and its computation is suspended. The fault can be thus injected at a single specific cycle and with the chosen strength of the power pulse. After the fault injection, the configuration is also read back in order to allow an accurate analysis of the elements that were modified by the attack. After this step, the clock is re-started to complete the encryp-

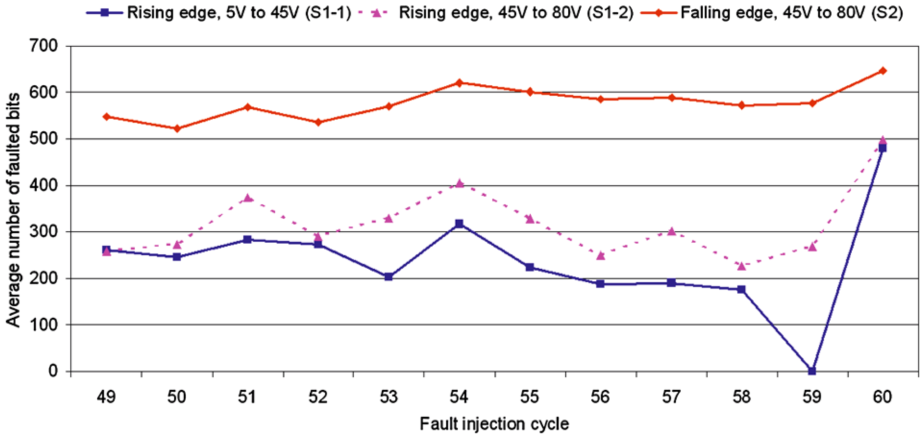


Fig. 5. Average number of faulted bits onto the configuration using power glitches faults injections for the different campaigns.

tion process. Finally, the different registers (encrypted result and error) can be read for subsequent analysis.

5.2. *Characterization of the Effects on the FPGA Configuration*

Previous articles have shown the modifications of the FPGA configuration due to laser faults injections [15–17], but no paper dealt with fault injections using power glitches. This paper allows filling this lack.

In order to know the effects of fault injections in the configuration or in the user flip-flops, it is necessary to use a mask file defined by the compilation tool ISE-9.2i. The mask file has the same length than the read-back file (i.e., the configuration read from the FPGA itself): it allows bit-by-bit comparison and it defines configuration and user flip-flop bits.

During the different campaigns (S1 and S2), no configuration bit of the design was modified, only the content of user flip-flops. This means that the injected errors can be accurately modeled as transient bit-flips. The same results were observed independently of the overshoot parameters.

Figure 5 shows the average number of modified bits as a function of the clock edge (rising or falling) and of the overshoot amplitude. These average numbers are computed from the total number of modified bits for each injection cycle and normalized with respect to the number of faulted configurations. We always observe the same trend for faults injected during the rising edge, independently of the voltage range, whereas the average number depends on the range: the higher the voltage, the more bits are modified.

Comparing the results obtained for both edges in the same voltage range (45 to 80 volts), we note that the average number of modified bits during the falling edge is twice than for the rising edge. A possible reason for these results may be the difference in the physical layout. Flip-flops are made of two latches (Fig. 6): the first one (Latch 1) is transparent when the clock is at one level and the second (Latch 2) is transparent during the other level of the clock. Latch 1 is connected only to Latch 2, while Latch 2 may be

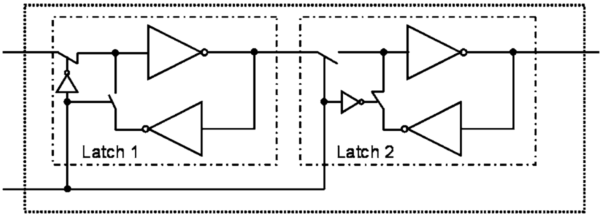


Fig. 6. Flip-flop schematic based on two latches.

Table 1. Global results obtained with overshoots included between 5 volts and 80 volts. *No. = Number, Perc = Percentage.*

Over-voltage range	Rising edge 5 V to 45 V		Rising edge 45 V to 80 V		Falling edge 45 V to 80 V	
Class	No.	Perc	No.	Perc	No.	Perc
No effect	1027	95.1%	809	84.3%	268	27.9%
Silent error	0	0.0%	0	0.0%	12	1.2%
False positive	16	1.5%	55	9.3%	60	6.2%
Detected error	37	3.4%	89	5.7%	157	16.4%
Undetected error	0	0.0%	7	0.7%	463	48.2%
Total	1080		960		960	

connected to several elements and signal drivers. This may result in design differences making one latch more sensitive to voltage spikes than the other.

5.3. *Global Results*

The majority of fault injections using power glitches on the rising edge have no effect onto the configuration and onto the ciphering (line *No effect* of Table 1). We observe in this table that a fault injection into the configuration always leads to a ciphering error (0% of silent error).

When the overshoot is between 5 and 45 volts, the proportion of error detection among the altered executions is 100%, including about 30% of false positives; no undetected errors at all are obtained. For glitch amplitudes between 45 and 80 volts, the proportion of errors is more significant (more than 15%) but most faults lead to false positives (57% of the corrupted executions). Some undetected errors were also obtained (4% of the errors, or 0.7% of the attacks) showing the possibility to bypass the countermeasure. In the next section, we will analyze these global results with respect to the injection cycle in order to identify which cycle is the most critical.

During the fault injection campaigns on the rising edge, more than 84% of the overshoots had no visible effect, while this proportion goes down to 28% when attacking on the falling-edge (Table 1). This result may be explained by the sensitivity of the flip-flop as mentioned in the previous section. The proportion of attacks with detected errors is more important for the falling edge campaign (16% for S2 against 5% for S1), but the proportion of undetected errors is also noticeably increased (48%). However, each corrupted outcome was always set to zeroes, which corresponds to a reset of the different registers (UART, crypto-processor). The configuration was not altered; further encryp-

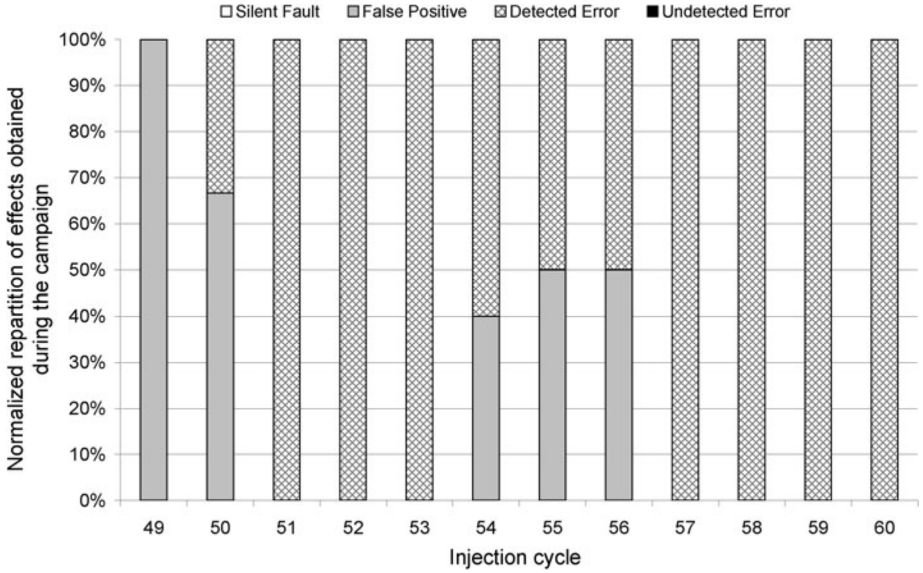


Fig. 7. Percentage of classes obtained for each injection cycle for power glitches between 5 volts and 45 volts.

tions would hence therefore give the correct results. Since no information on computed data is leaked, these errors are therefore not exploitable to mount a fault analysis attack.

To conclude, these results demonstrate the efficiency of the implemented countermeasure against transient faults when injected during the rising edge. We have also demonstrated the difference of sensitivity of flip-flops with respect to the injection edge. Finally, although more errors go undetected when injecting on the falling edge, the wrong encryption results cannot be exploited by the attacker.

5.4. *Results with Respect to the Injection Cycle*

In the previous section, we have shown that injecting faults during the clock falling edge is not effective to recover the key because each outcome was zeroed. So, in this section, we will present only the results obtained for the rising edge.

In Fig. 7, we observe the repartition of the classes obtained at each injection cycle for overshoots between 5 and 45 volts. We observe that some cycles are more prone to some classes; for example, if the fault is injected during the two cycles at the beginning of the round, then the fault mainly leads to false positives. Nevertheless the countermeasure is efficient because the detection alarm is raised for each fault leading to ciphering errors.

The same results were observed for glitch amplitudes from 45 to 80 volts (Fig. 8). We have previously mentioned the possibility to obtain undetected errors. This occurs only if the glitch is injected during the last operation of the encryption process. However, this kind of error should be not exploitable because it affects only the key addition and/or the register writings. The attacker may try to distinguish different bit transitions (0-to-1 and 1-to-0) by monitoring and analyzing the power consumption or the EM emissions:

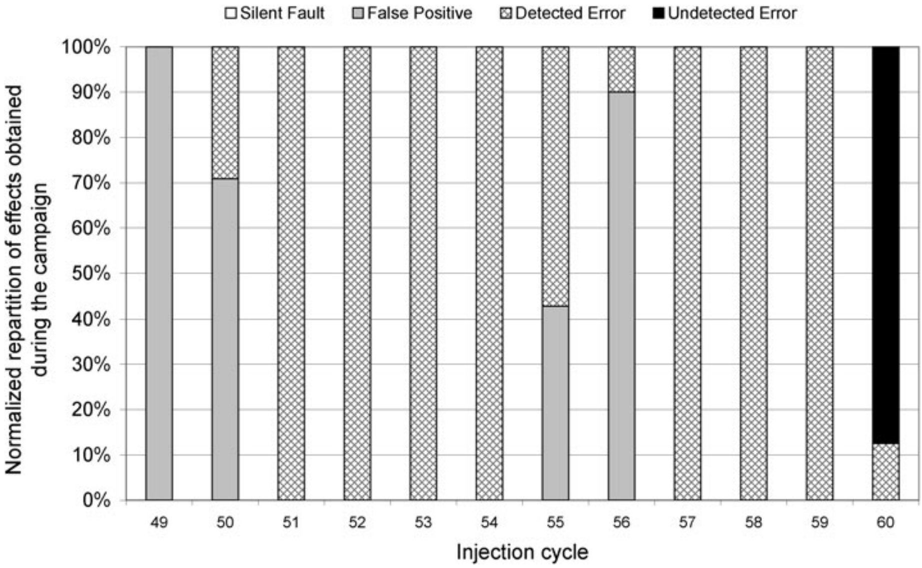


Fig. 8. Percentage of classes obtained for each injection cycle for power glitches between 45 volts and 80 volts.

however, such an approach would be disturbed by all the noise generated by the power spike used to alter the computation process.

6. Laser Attacks

6.1. *Setup*

During the laser-based experiments, the device was configured with a Universal Asynchronous Receiver Transmitter (UART) and the secure AES crypto-processor presented in Sect. 2. Some place-and-route (PAR) constraints were added to reduce the experiment duration. The target device measures approximately 1 cm^2; since the displacement step used for the campaign was 20 µm, studying the entire device would have taken more than 6 months without constraints. Therefore, we constrained the placement and routing of one column of the state: one S-Box, the linear function, and one DDR register. Only these three elements are studied because they are representative of a large part of the design. The results can be thus considered valid, with proper considerations, for the whole encryption data path. The three blocks were placed at the top left corner of the device, while the rest of the design (crypto-processor and UART) were placed at the bottom (Fig. 9). Thus, only the desired elements have been modified during the attack.

The device used is Flip-Chip encapsulated and it was attacked backside through the substrate (Fig. 10). The initial XC2V1000 device has a width of 10.6 mm, a length of 9.7 mm, and a thickness of 790 µm. Following a classical attack procedure, a mechanical process was employed to thin the die until a residual thickness of 30 µm. This is usually done to ensure a good optical transmission of the light in the active layers of the device. Without this thinning, the laser light does not affect the active layers and no errors

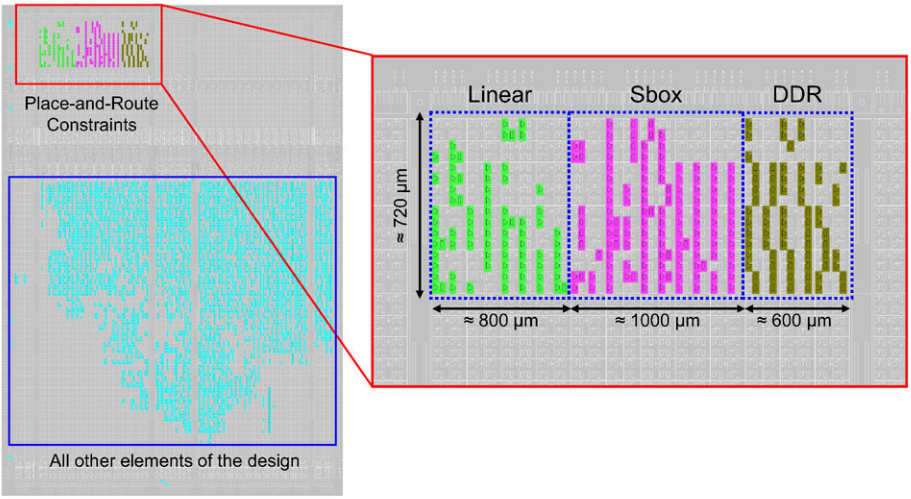


Fig. 9. Place-and-Route Constraint used during the experiments.

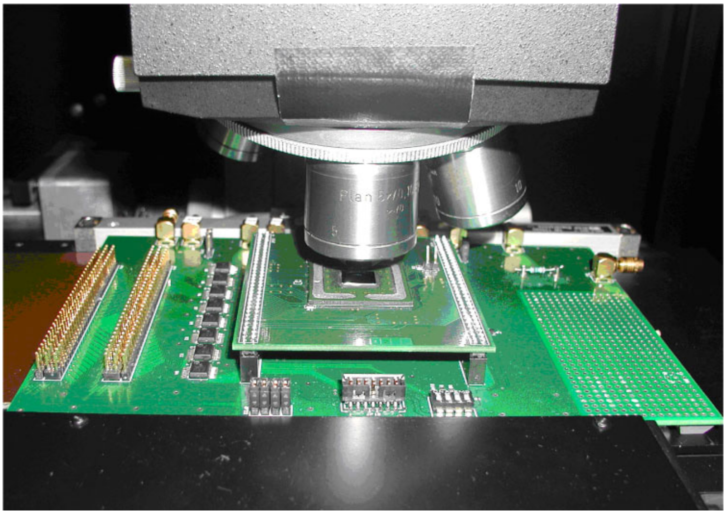


Fig. 10. Laser fault injection equipment (inner view).

are generated. The laser wavelength is about 900 nm in order to have a good laser penetration depth; the beam had a power of a few Watts and a 20 µm spot diameter. Figure 10 shows the test board with the device under the laser equipment.

For these experiments, the attacked area (Fig. 9) approximately measures 500 µm by 600 µm. A systematic scan of the selected zone was done using an X-Y table.

Faults have been injected during the ninth ciphering round to seek exploitable results and a simple cryptanalysis step. Earlier injection would be difficult to analyze; on the other hand, injecting later may not guarantee to give exploitable results. Since the crypto-processor uses six clock cycles per round [20] and the main state machine is active on the rising edge, we injected the faults at the rising edge only from the 49th to 54th clock cycles.

The test sequence used during the laser experiments is similar to the one presented in Sect. 5.1. Unlike the approach in [17], the fault can be injected during a single specific cycle. Thanks to this methodology, the pulse duration is unconstrained.

6.2. *Characterization of Laser Effects*

It must be first noticed the high sensitivity of the CLB tiles [16]. The average number of modified bits per shot can be quite high and clearly depends on the laser spot size. The actual number also depends on the initial configuration, since the probability to flip a ‘1’ is greater than the probability to flip a ‘0’.

As previously mentioned, a CLB tile is composed of two different parts: the internal logic and the interconnections. For the logical part, the most sensitive parts are the LUT contents and the internal multiplexers. For the interconnection part, the effect of the fault depends on its initial state and on the fact that each connection may be defined by one to three bits in the configuration. When there was no connection, then a new one may or may not be actually created: in most cases, even if the configuration was modified, there were no actual consequences on the connection. This is due to the fact that multiple bits must be properly modified and this is not so likely. If a connection was already set, then the injected fault may lead to different patterns: the connection is modified, there is no effect, the connection is suppressed, or further connections are added. These two latter situations are the most likely.

6.3. *Global Results*

During the campaign, more than 1400 laser shots have been performed for each clock cycle during the ninth encryption round. The results are shown in Table 2 and classified according to the categories presented in Sect. 3. A significant number (about 13%) of the laser shots were actually ineffective, since the configuration was not modified. This result can be explained either by the value of the configuration bit or by the shot position. In [15] and [23], it was shown that the sensitivity of the bit depends on its initial value. According to [15], the probability to flip a one is about 2.5 times higher than the probability to flip a zero which is the configuration default value. An explanation concerning this result is given in [23]. The structure of the configuration memory cell is not similar to a classical SRAM cell; the 6-transistor structure is replaced by a 5-transistor memory element with a sixth transistor used to implement the power-up-reset, creating a dissymmetry between the input and output inverters. This analysis confirms that the probability to flip a bit initially at ‘1’ is higher than to flip a ‘0’.

When the FPGA configuration is modified, more than 38% of the laser shots, although modifying the configuration, did not produce any tangible effect (silent errors). About 50% of the experiments led to ciphering errors: in particular, 29% were detected errors, while a troublesome 18% went undetected. These results demonstrate the insufficient efficiency of the implemented countermeasure against laser-based fault injections into a SRAM-based FPGA.

Table 2. Global results of laser injections during the ninth ciphering round.

Class	Number	Percentage
No effect	1155	13.4%
Silent error	3302	38.4%
False positive	44	0.5%
Detected error	2549	29.6%
Undetected error	1557	18.1%
Total	8610	

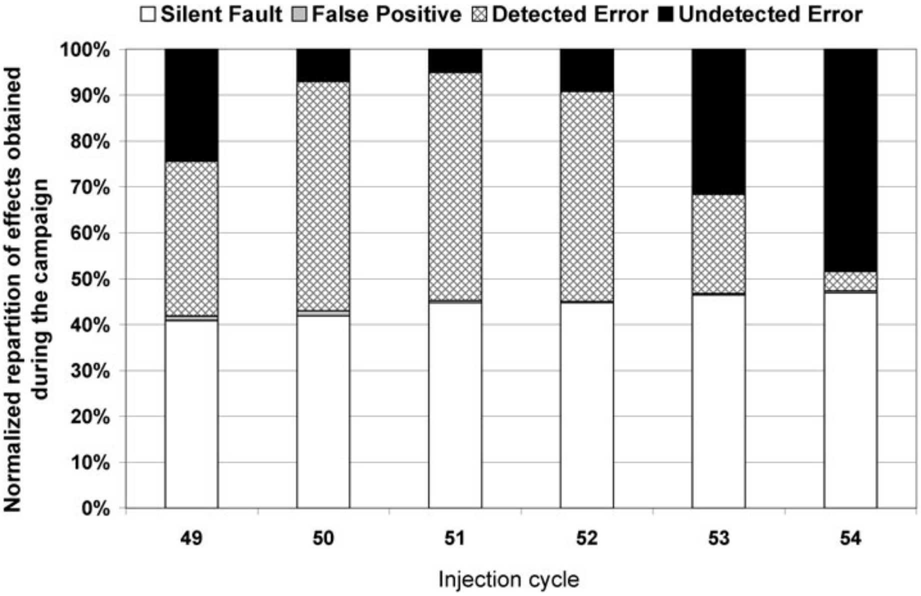


Fig. 11. Outcome of the experiments when the configuration is actually modified.

6.4. *Results with Respect to the Injection Cycle*

We have shown in the previous section that different outcomes can be expected from a laser shot. An analysis of these effects with respect to the injection cycle is presented in Fig. 11, only when the configuration is modified. The number of silent faults is close to 39% independently of the clock cycle. When a fault injection leads to an unexpected ciphering value, the proportion of detected errors depends on the injection cycle.

If the fault occurs during the main AES computation (clock cycles 49 to 51), the detection probability is high, while if the error occurs during the second copy of the round (cycles 52 to 54), then the probability of undetected ciphering faults increases. The FPGA used here is a SRAM-based device so a configuration error remains until a new configuration is loaded. If the fault occurs in the last clock cycle of the ninth round, then the configuration error is present during the whole computation of the last round (round 10, composed of both the main and verification cycles). Because the AES architecture uses the same S-boxes for the main and the verification cycle, a fault will modify both computations and the error will not be detected, in spite of the temporal redundancy.

Although the fault model used in the two experiments is quite different (long transient faults in the emulation campaigns, remanent faults occurring during the encryption in the laser experiments), we can find some common points. In particular, we can clearly identify in both campaigns the main vulnerability of an approach based on temporal redundancy. When the fault is able to affect both computations, then it will be hardly detected. This occurs when the transient fault lasts six clock cycles (i.e., an entire round and its repetition), or when the remanent fault caused by the laser is created at the beginning of the last round. Finally, we can also see that the last computation cycles are also quite vulnerable. All these flaws need to be addressed with further interventions.

7. Securing Against Faults in the Configuration SRAM

7.1. *Further Hardening the Design*

The detection scheme based on the DDR computation template is well suited to protect against transient faults, as all mechanisms based on temporal redundancy. This assumption was demonstrated both with emulated fault injections (Sect. 3) and fault injections by power glitches (Sect. 5). As seen in the previous section, however, laser-based fault injections on SRAM-based FPGAs are able to alter the configuration of the device. Since these modifications will remain just until the next reconfiguration, they can be described as remanent faults.

Remanent faults may be difficult to detect when using an approach based on temporal redundancy, because they may alter all the process repetitions in the same way. Thus, a different technique must be used as a complement. For instance, critical components may be duplicated and different process repetitions may use different resources. This solution is effective, but also highly expensive, since it may require a large area overhead: for the considered architecture, this would require duplicating all the substitution boxes, which are already the largest blocks of the architecture. An approach based on partial duplication may be used [24], but the overhead would be about 25%, which is already non negligible.

Similar results, however, can be obtained without directly resorting to hardware redundancy. An alternative approach is based on allocating resource usage differently each time the process is recomputed. In our design, the main and the auxiliary copy of data share the non-linear combinational logic (i.e., the S-Boxes) and four functional units are available in the design working in parallel. Hence, we propose as an additional countermeasure to rotate data when the computation is repeated on the auxiliary copy; thus, each value is processed by two different functional units. If a remanent fault occurs, then it will be detected easily by comparing the different outputs.

During the second computation, each row is permutated before entering the substitution boxes and the order is restored later. Due to the structure of AES, the overhead is quite limited: only a few multiplexers are used to manage the permutation at the input of non-linear layer, while the proper alignment is restored at almost no cost by using the existing logic that implements the ShiftRows operation. Additionally, the key scheduler was protected as well, in particular when the shared logic is used: the key is sent to the

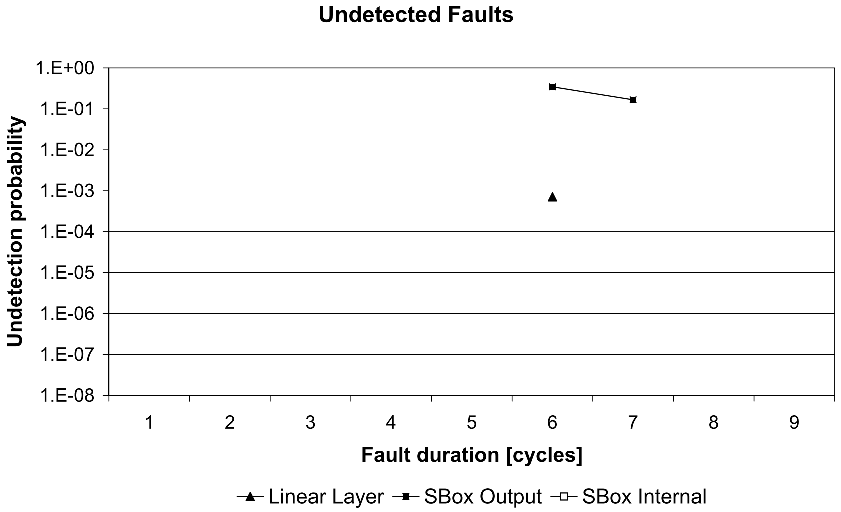


Fig. 12. Undetected transient faults with emulated injections (new countermeasure).

S-Boxes twice, by exploiting the cycles when they are not in use by the encryption data path, and the output is verified. Since the key word is never permutated, this allows a complementary check of the non-linear logic.

It is interesting to note that the new countermeasures do not increase the area used on the FPGA: a slightly larger number of flip-flops are used, due to the additional registers in the key scheduler, but on the other hand less slices are actually required, thanks to optimized resource usage. Without the PAR constraints that were used to facilitate the laser injection campaign, the new implementation uses 54 additional flip-flops (2023 against 1969), but 41 less slices (1699 against 1740).

7.2. *Emulation Results*

Before performing the laser injection campaigns, the new countermeasures were validated with emulated fault injections. The same campaigns were run as in Sect. 3; the results are shown in Fig. 12.

It can be seen that the number of undetected faults is drastically reduced. The new countermeasures help detecting almost all possible transient faults. In particular, only a very small number of faults into the state register are not detected: namely, those which inject the same value in both the main and the auxiliary copy for an entire encryption round. Likely, the faults injected in the substitution boxes are all suppressed, except some of those lasting a complete round.

7.3. *Laser Results*

For this new countermeasure, we use place-and-route constraints that are comparable to the previous version; the same zone is also studied. However, the netlists are slightly different, which must be taken into consideration when comparing the results.

Table 3. Global results of fault injections during the ninth ciphering round (new countermeasure).

Class	Number	Percentage
No effect	4698	54.6%
Silent error	1548	18.0%
False positive	89	1.0%
Detected error	2263	26.3%
Undetected error	11	0.1%
Total	8610	

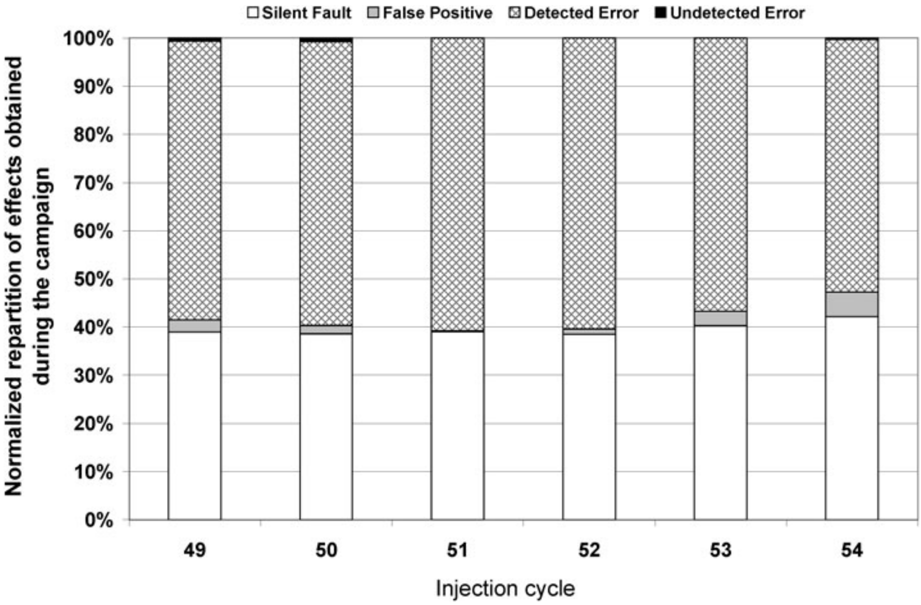


Fig. 13. Outcome of the experiments when the configuration is actually modified (new countermeasure).

Unlike the previous experiments, most laser shots had no effect at all onto the configuration (54.6% for this campaign against 13.4% previously; see Table 3). This result can be explained by the implemented design, but also by the experimental setup which can easily change between different experiments (laser focalization, circuit and ambient temperature). Due to large differences in the fault injection results, it is quite difficult to precisely compare the new design with the previous one.

Nevertheless, we can observe that only 11 laser shots lead to undetected errors and they represent only 0.1% of the whole campaign. Additionally, the number of detected errors increases, which shows that most errors that were undetected are now properly identified. The new architecture is definitely more secure, since less than 0.3% of injected errors are undetected (against 20.9% of the previous version).

This is independent of the injection cycle as shown in Fig. 13. The percentage of silent faults is close to 40% for both implemented designs, while the ratio of undetected faults is considerably different. When the injected fault induced a corrupted encryption result, the error is detected in 99% of the cases.

7.4. *Exploitation of the Attack*

Since the undetected faults are very few, it is interesting to study whether the errors are exploitable to recover the secret key. In a real case, this step would be hard for the attacker: rewinding the execution to understand the effects of the faults can be easily accomplished by already knowing the secret key, but it would be quite difficult in a black box approach.

A few different patterns were recognizable in the 11 faulty outputs. It was seen that the same or similar errors were injected by a pair of injections, which were related by injection coordinates and timing. As expected, adjacent locations gave the same error when the shots were only slightly delayed in time.

In six experiments, we were able to suppress the key addition for a single byte. This can be easily exploitable by an attacker, who may recover a byte of the last round key just by comparing the final outputs. This result was a consequence of the fine area optimizations performed on the design, which increased the criticality of some functional blocks. This vulnerability can be easily addressed by increasing the redundancy in the key addition logic: the overhead will be very limited when compared to the entire architecture.

A few other faults suppressed the computation of the MixColumns operation. This vulnerability can be quite serious, since it removes completely the diffusion layer of AES. If the fault affects the architecture from the beginning of the encryption, then each byte of the result depends on only one byte of the input. Hence, the secret key may be found by incremental brute search one byte at a time. Again, this is due to the quite aggressive optimizations chosen when designing the linear functional blocks. Again, this vulnerability can be addressed by hardening the vulnerable functions.

It must be observed that all the control signals were encoded in dual rail for improved security. However, due to characteristics of the FPGA and the difficulty to control the synthesis process, this was not enough against the laser-based injections, since connections and routing are still controlled by few bits. It is thus mandatory, in a real case application, to verify that all the countermeasures are properly implemented during the design flow.

8. Conclusion

SRAM-based FPGAs are becoming common for low-volume markets, thanks to their affordability and versatility. Cryptographic accelerators are one of the possible application fields: on the other hand, we must consider vulnerabilities coming from side-channel analysis or fault attacks.

In this work, we have implemented an AES architecture secured against fault attacks; we have then conducted extensive fault injection campaigns by means of power glitches and laser-based attacks. To our knowledge, this is the first time that a secured cryptographic implementation on FPGA has been attacked with laser-based injections during runtime. We have shown that, although the implemented countermeasure had proved itself very resistant against transient faults, FPGA implementations may also be vulnerable to remanent faults, i.e. faults that modify the configuration of the device.

This additional source of weakness, which is due to the specific technology of the considered class of FPGAs, has been addressed with further dedicated and inexpensive protections. A new experimental campaign proved it to be very efficient: only a very limited percentage (0.1%) of injected faults remained undetected. The undetected errors affected the functionality of the device: the ciphering algorithm was significantly altered and some operations were completely suppressed. This revealed a few exploitable vulnerabilities in the linear functional blocks of the architecture. However, due to the simplicity and the small size of these blocks, effective countermeasures can be envisioned, at a limited cost with respect to the entire design. Additionally, the designer may choose to extend the additional proposed countermeasure (implemented here only for the substitution boxes) to the whole encryption data path to improve robustness against permanent faults at a reasonable cost.

This work shows that secure implementations on reconfigurable devices must face a large variety of threats. One of the major contributions is that remanent faults may be an effective mean to get the secret key: the fact that the function computed by the circuit may be changed allows altering the computation, such as by making the device skip some ciphering steps. In one of our experiments, for instance, the diffusion function was suppressed: running further encryptions on such a faulty device would easily compromise the confidential information. It must be said, however, that a thorough analysis without any knowledge on the circuit or on the key (i.e., a black box approach) would be quite difficult. On the other hand, the attacker may find an easy breach by performing some selected encryptions, which would be used to gather useful information.

These threats are, however, difficult to counteract by relying solely on architectural countermeasures, which might always be circumvented by changing the configuration logic. It would be therefore advisable to adopt more robust mechanisms in order to assure the dependability of programmable circuits when used in sensitive contexts.

References

[1] S.B. Örs, E. Oswald, B. Preneel, Power-analysis attacks on an FPGA—First experimental results, in *Proceedings of the 13th International Conference on Field-Programmable Logic and Applications* (2003), pp. 35–50

[2] K. Gandolfi, C. Mourtel, F. Olivier, Electromagnetic analysis: concrete results, in *The Proceedings of Cryptographic Hardware and Embedded Systems* (CHES 2001). Lecture Notes in Computer Science, vol. 2162 (Springer, Berlin, 2001), pp. 251–261

[3] D. Agrawal, B. Archambeault, J.R. Rao, P. Rohatgi, The EM Side-channel(s), in *The Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2002)*. Lecture Notes in Computer Science, vol. 2523 (Springer, Berlin, 2003), pp. 29–45

[4] D. Carluccio, K. Lemke, C. Paar, Electromagnetic side channel analysis of a contactless smart card: First results, in *The Workshop on RFID and Lightweight Crypto* (RFIDSec05), Graz, Austria, July 13–15 (2005)

[5] D. Boneh, R. DeMillo, R. Lipton, On the importance of eliminating errors in cryptographic computations. *J. Cryptol.* **14**, 101–119 (2001)

[6] H. Bar El, H. Choukri, D. Naccache, M. Tunstall, C. Whelan, The sorcerer's Apprentice guide to fault attacks. *Proc. IEEE* **94**(2), 370–382 (2006)

[7] G. Piret, J.-J. Quisquater, A differential fault attack technique against SPN structures, with application to the AES and Khazad, in *Proc. Fifth Int'l Workshop Cryptographic Hardware and Embedded Systems* (CHES '03), vol. 2779 (2003), pp. 77–88

[8] S.-M. Yen, S. Moon, J.-C. Ha, Hardware fault attack on RSA with CRT revisited, in *Proceedings of the Information Security and Cryptology—ICISC 2002*. Lecture Notes in Computer Science, vol. 2587 (Springer, Berlin, 2003), pp. 374–388

[9] N. Selmane, S. Guilley, J.-L. Danger, Practical setup time violation attacks on AES, in *Proceedings of the Seventh European Dependable Computing Conference* (EDCC 2008), May (2008), pp. 91–96

[10] S. Bhasin, J.-L. Danger, S. Guilley, N. Selmane, Security evaluation of different AES implementations against practical setup time violation attacks in FPGAs, in *Proc. of the IEEE International Workshop on Hardware-Oriented Security and Trust* (HOST 2009) (IEEE CS, Los Alamitos, 2009), pp. 15–21

[11] N. Selmane, S. Bhasin, S. Guilley, T. Graba, J.-L. Danger, WDDL is protected against setup time violation attacks, in *Proceedings of the 6th International Workshop on Fault Diagnosis and Tolerance in Cryptography* (FDTC 2009) (IEEE Computer Society, Los Alamitos, 2009), pp. 73–83

[12] S.P. Skorobogatov, R.J. Anderson, Optical fault induction attacks, in *Cryptographic Hardware and Embedded Systems—CHES 2002, 4th International Workshop*. Lecture Notes in Computer Science, vol. 2523 (Springer, Berlin, 2003), pp. 2–12

[13] J.-M. Schmidt, M. Hutter, Optical and EM fault-attacks on CRT-based RSA: concrete results, in *The Proceedings of the Austrochip 2007* (Springer, Berlin, 2007), pp. 61–67. ISBN:978-3-902465-87-0

[14] D.H. Habing, The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. *IEEE Trans. Nucl. Sci.* **39**, 1647–1653 (1992)

[15] V. Maingot, J.B. Ferron, R. Leveugle, V. Pouget, A. Douin, Configuration errors analysis in SRAM-based FPGAs: software tool and practical results. *Microelectron. Reliab.* **47**(9–11), 1836–1840 (2007)

[16] G. Canivet, J. Clédière, J.B. Ferron, F. Valette, M. Renaudin, R. Leveugle, Detailed analyses of single laser shot effects in the configuration of a Virtex-II FPGA, in *International On-Line Testing Symposium* (IOLTS'08) (2008), pp. 289–294

[17] V. Pouget, A. Douin, G. Foucard, P. Peronnard, D. Lewis, P. Fouillat, R. Velazco, Dynamic testing of an SRAM-based FPGA by time-resolved laser fault injection, in *International On-Line Testing Symposium* (IOLTS'08) (2008), pp. 295–201

[18] National Institute of Standards and Technology (NIST), FIPS-197: Advanced Encryption Standard, Nov. 2001

[19] K. Wu, R. Karri, Idle cycles based concurrent error detection of rc6 encryption, in *Proceedings of the 16th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems* (DFT '01) (2001), pp. 200–205

[20] P. Maistri, R. Leveugle, Double-data-rate computation as a countermeasure against fault analysis. *IEEE Trans. Comput.* **57**(11), 1528–1539 (2008)

[21] N. Pramstaller, S. Mangard, S. Dominikus, J. Wolkerstorfer, Efficient AES implementations on ASICs and FPGAs, in *Proceedings of the Fourth International Conference on the Advanced Encryption Standard* (AES '04) (Springer, Berlin, 2004), pp. 98–112

[22] Xilinx, Virtex-II Platform FPGAs: Functional Description, Data Sheet DS031, module 2 of 4, November 5, 2007

[23] G. Canivet, R. Leveugle, J. Clédière, F. Valette, M. Renaudin, Characterization of effective laser spots during attacks in the configuration of a Virtex-II FPGA, in *VLSI Test Symposium* (VTS'09) (Springer, Berlin, 2009), pp. 327–332

[24] G. Di Natale, M. Doulcier, M.-L. Flottes, B. Rouzeyre, A reliable architecture for parallel implementations of the advanced encryption standard. *J. Electron. Test.* **25**(4–5) (2009)