Experimental Analysis of the Electromagnetic Instruction Skip Fault Model and Consequences for Software Countermeasures*

Jean-Max Dutertre[a,*], Alexandre Menu[a], Olivier Potin[a], Jean-Baptiste Rigaud[a] and Jean-Luc Danger[b]

[a]Mines Saint-Etienne, CEA-Tech, Centre CMP, F - 13541 Gardanne France
[b]LTCI, Télécom Paris, IP Paris, France

ARTICLE INFO

Keywords:
Hardware attacks
EM fault injection
Fault model

ABSTRACT

Microcontrollers storing valuable data or using security functions are vulnerable to fault injection attacks. Among the various types of faults, instruction skips induced at runtime proved to be effective against identification routines or encryption algorithms. Until recently, most research works assessed a fault model that consists in a single instruction skip, i.e. the ability to prevent one chosen instruction in a program from being executed. We question this fault model for EM fault injection on experimental basis and report the possibility to induce several consecutive instructions skips. Such an extended fault model proved to be effective against a duplication-based software countermeasure as our experiments revealed.

1. Introduction

Hardware attacks take advantage of the physical implementation of an electronic device to overcome its security features. To this purpose, either passive side channel techniques or active fault injection techniques can be used. The former is out of the scope of this article. Fault injection techniques aims to force the device out of its specification by altering its environmental conditions [2] (e.g. its voltage, temperature, frequency, etc.). An attacker that successfully induces an erroneous behavior refers to it as an injected fault. These faults can then be used as an attack primitive to extract a cryptographic key [4] or provide an unauthorized access to some of the target functionalities [11]. The set of properties of a fault induced by an electromagnetic (EM) perturbation (or by any other fault injection means) is referred to as a fault model (FM). It is often linked to a given attack scheme and expressed as an ability to meet requirements in terms of synchronization with the target activity and extension of the induced fault (e.g. the FM of the well-known Piret fault attack (FA) [25] requires to fault one byte of the AES algorithm calculations before its last MixColumn transformation).

Among fault injection (FI) techniques, laser FI and EM fault injection (EMFI) achieve the best performances [12, 6, 9]. Laser is probably the most expensive FI means, however it makes it possible to inject faults with high accuracy even at advanced technology nodes [12]. It is accurate both in terms of timing (faults may be injected with laser pulses as short as a few picoseconds [18]) and in terms of location (its effect is mainly limited to the logic gates located within its spot size which may be as low as a few micrometers [6]). EMFI on the other hand does not require a direct access to the die, the clock, or the power supply of the target, while enabling an attacker to fault micro-architectural features [27]. As an operational setup can be build for less than a thousand dollars [8], EMFI is often considered a good compromise between price and locality.

In this work, we report our analysis of the EM-induced instruction skip FM. This FM relates to how a given instruction of a microcontroller program may be skipped (i.e. not executed) at runtime. Several works already described this FM and assessed the possibility of EM-induced single instruction skips [3, 21, 23, 22]. The assessment of FMs on experimental grounds is of high interest regarding how FIs countermeasures (CMs) are designed and tailored. As a matter of example, the authors of [22] discussed two CMs based on instruction redundancy designed on the assumption that an attacker is only able to induce single instruction skips. Would this assumption be proved wrong, their CMs would be vulnerable.

Our experiments extend further this FM by reporting the feasibility of inducing several successive instruction skips by EM perturbation.

Our contributions are as follows:

- We achieve to skip, with a perfect reproducibility, several consecutive fetch operation with a single EM perturbation, controling both the timing and the number of the affected instructions,
- We highlight the influence on the observed fault model of the width of the voltage pulse inducing the EM perturbation,
- We demonstrate practical applications on an 8-bit microcontroller, bypassing several software CM. We were indeed able to defeat an instruction duplication-based CM (similar to that described in [22]).

This article is organized as follows. Section 2 discusses the state-of-the-art of instruction skips and introduces the

*This research has been partially supported by the European Commission under H2020 SPARTA (Grant Agreement 830892).

*Corresponding author
dutertre@emse.fr (J. Dutertre); alexandre.menu@emse.fr (A. Menu); olivier.potin@emse.fr (O. Potin); rigaud@emse.fr (J. Rigaud); jean-luc.danger@telecom-paris.fr (J. Danger)
ORCID(s): 0000-0002-2251-7815 (J. Dutertre)

aim of our work. Our experimental setup and settings are described in section 3. Section 4 reports the obtained results. Section 5 describes how the observed extended FM is able to defeat a software countermeasure based on instruction duplication. Then, the obtained FM is discussed in section 6. The last section concludes the paper.

2. The EM-induced instruction skip fault model

Our first research objective was to reproduce EM-induced instruction skips on a microcontroller and to study the main characteristics of its FM: accuracy (i.e. ability to choose the skipped instruction), extent (i.e. number of skipped instructions), success rate, etc. Our aim was also to assess if the multiple instructions skip fault model was achievable on a simple target (one that has no cache or complex pipeline) or whether it relies only on specific micro-architectural features. This question is of interest, as CMs based on a too narrow FM may reveal vulnerabilities at test time.

2.1. Fault model definition

A fault model refers indisctintly either to the abstract properties of a fault or to the main properties of a FA scheme. These are often expressed in terms of extension (e.g. bit, byte, nibble) and synchronization with the execution of the program. However, a fault and a FA scheme are most conveniently described at two different level of abstraction, i.e. the physical layer and the logical layer. The authors of [12] follow the former approach and describe laser-induced bit-set and bit-reset faults at the transistor and gate level. However, this approach is extremely difficult and time consuming if the attacker has a limited insight on the device internals. The authors of [25] follow the latter approach and describe a mono-byte FM to corrupt the AES algorithm calculations before its last MixColumn transformation. However, this approach faces practical limitations in understanding the underlying fault mechanism because of the complex interaction between the physical and the logical layer of an electronic device.

In our approach, we considered the instruction set architecture (ISA) which is a natural interface between the physical and the logical layer. Previous works highlighted that the characterization of FM at the level of the ISA covers a wide range of practical faults [22, 1]. In this article, we followed this approach and analyzed the behavior of a microcontroller target at the ISA level in order to assess and study EM-induced instruction skips.

2.2. Instruction skip fault model

An instruction skip is a fault that results in skipping, meaning not executing, one instruction of a program at runtime. There is to date very few explanations of how an instruction skip is induced at gate level, with the notable exception of [1]. It describes how progressively increasing the stress applied by a clock glitch to a microcontroller induces an increasing number of bit-reset faults into the opcode of an instruction. It results in (1) instruction modification at low stress or (2) in turning the instruction into an actual `no operation` instruction (`nop`) at high stress.

An instruction modification has the same effect as an instruction skip if the modified instruction has no effect on the context of the program. While this modification are the most frequent in practice [7], we focused our experiments toward achieving instruction skips by turning the target instructions into `nop` instructions.

Several works studied the EM-induced instruction skip FM. Most of them assessed *single instruction* skips, in the same 8-bit microcontroller as the one studied in this work [3], or in a 32-bit microcontroller [22].

Several works also studied the laser-induced instruction skip FM. In [5], the authors obtained single instruction skips with high accuracy and high success rate. Still on the same target, [17] reports single instruction skips based on resetting one or two bits of the targeted instruction opcode. The authors of [28] induced single instruction skips on a more complex 32-bit cortex-M3 microcontroller. They were able to inject two single instruction skips distant from 58 ms to defeat a protected CRT-RSA algorithm.

While the instruction skip FM alone does not encompass the complexity of practical faults [7], skipping multiple instructions was, until recently, a theoretical consideration.

2.3. Multiple instructions skip fault model

The state-of-the-art in laser-induced instruction skip reports that the number of successive instructions that could be skipped was not limited in the same 8-bit microcontroller as the one studied in this work [13]. Moreover, the injection can be syncronized with the program execution and the duration of the laser pulse is linearly correlated with the number of consecutive instructions which are skipped. This is a strong FM as entire sections of code can be selectively *erased* by an attacker. To the best of our knowledge, the only works reporting several successive instructions skips with other injection techniques are [27], [29] and [14, 15]. The authors of [27] assessed four successive EM-induced skips of instructions stored in the instruction cache of an ARMv7 microcontroller. The authors of [29] succeeded in faulting instructions stored in the pipeline of a FPGA implementation of the LEON7 core with a clock glitch. More recently, [14, 15] reports EM-induced skips of up to six consecutive instructions with a low success rate in a modern RISC-V microcontroller. However, they did not provide an understanding of the micro-architectural effects.

Based on previous experiments and taking into account the state-of-the-art we described, we focused our experiments toward achieving instruction skips by turning the targeted instructions into `nop` instructions. Our experiments were carried out with the same 8-bit microcontroller studied by [1, 3, 5, 17], so that our results can be easily compared. Moreover, this microcontroller has a very simple 2-stage pipelined architecture and no caches, which is easy to analyse in the absence of unwanted architectural effects. The results of this research should give a better insight on the instruction skip

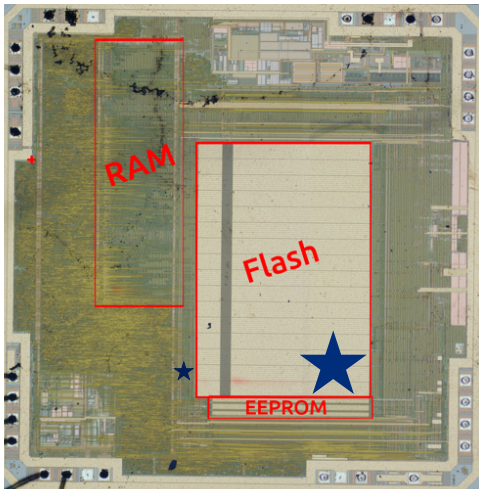


Figure 1: Frontside view of the ATmega328P test chip - Flash, RAM and EEPROM memories highlighted in red; fault sensitive areas for laser and EM injection highlighted resp. by a small and a greater deep blue stars.

FM when induced through EMFI and also provide a comparison basis with laser injection.

3. Experimental settings

3.1. EM injection bench

The EM pulse injection setup we used is similar to the one described in [9], it consists in the following elements:

- a voltage pulse generator,
- an injection probe,
- a XYZ positioning table,
- an oscilloscope,
- a target (described in subsection 3.2),
- a control PC interfacing the different elements of the bench and running the test series.

The EM disturbance that induces a fault is generated thanks to the voltage pulse generator: it delivers a square voltage pulse with a transition time of 2 ns (for the first edge of the pulse, the second edge has a transition time of 5 ns), a maximum amplitude of $\pm$400 V, and a width in the 6 - 100 ns range. The voltage pulse edges are converted into current variations in a coil wrapped arround the tip of a handcrafted injection probe. The coil is made of six turns of copper wire around a ferrite core which diameter is $\sim 500\mu$m. The swift current variation induces an EM perturbation at the root cause of the injected faults. A trigger signal generated by the device under test synchronizes the voltage pulse with the operations of the microcontroller target (the corresponding jitter was measured below 0.5 ns).

```
1  # Store 0x39 to 0x30 in RAM at address Z
2  # Initialize r16 to r25 at 0x55
3  # Set synchronization trigger
4  nop # 400 ns
5  # Set core trigger
6  ld r16,Z+      ld r16,Z+
7  ld r17,Z+      ld r17,Z+
8  ld r18,Z+      ld r18,Z+
9  ld r19,Z+      nop
10 ld r20,Z+      ld r20,Z+
11 ld r21,Z+      ld r21,Z+
12 ld r22,Z+      ld r22,Z+
13 ld r23,Z+      ld r23,Z+
14 ld r24,Z+      ld r24,Z+
15 ld r25,Z+      ld r25,Z+
16 # Clear core trigger
17 nop # 700 ns
18 # Clear synchronization trigger
19 # read back r16 to r25
```

Listing 1: Test code - Instruction skip analysis.

3.2. Test chip

We chose a simple target for the purpose of being able to analyse easily its responses to fault injection: an 8-bit non-secure ATmega328P microcontroller designed in the CMOS 0.35 μm old technology. It has 2 kB RAM, 32 kB Flash and 1 kB EEPROM memories; a Harvard architecture with a 2-stage fetch-execute pipeline. It runs at 16 MHz and has 32 general purpose registers (registers r16 to r25 were used during our experiments). Figure 1 gives a front view of the test chip with its Flash, RAM and EEPROM memories highlighted in red. The laser sensitive area that makes it possible to induce an instruction skip is highlighted by a small deep blue star (slightly outside the Flash memory at its left), it is in the order of a few micrometers [13]. The EM sensitive area, which is convenient for inducing instruction skips, is larger (see experimental results in section 4), it is in the order of a hundred of micrometers. It is located in the bottom-right part of the Flash memory and highlighted by a greater deep blue star in Figure 1. As the strength of the magnetic field generated by the coil strongly decreases with the distance from the injection probe, it was placed close to the target's silicon die ($\sim$1 mm). To this end, the package of the target device was open on its top by chemical etching.

3.3. Test codes

We studied the effect of EM-induced faults on dedicated test codes mostly written in assembly langage (see Listing 1). Our intent was to induce and analyze instruction skips by examining their effect.

For each test series, we used two trigger signals, both produced by the test chip (as denoted in Listing 1), for synchronization purposes:

- a synchronization trigger signal to accommodate for the latency of our EM injection setup (about $\sim$ 300 ns),
- a core trigger signal to synchronize the actual EM perturbation (thanks to an image of the transmitted perturbation, see Fig. 2 as an example) with the part of the assembly code of interest.

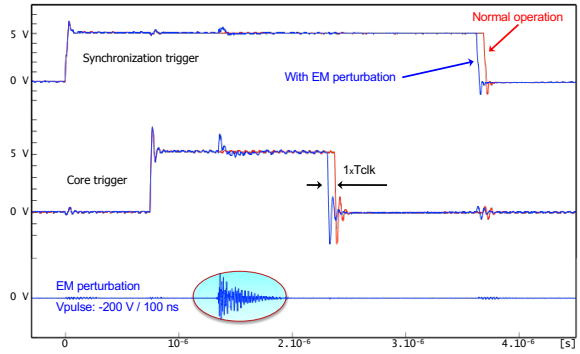


Figure 2: EM-induced single instruction skip: effect on execution time, trigger waveforms modification due to EM injection (from top to bottom, synchronization trigger, core trigger, and approximated image of the EM perturbation).

Table 1
Registers r16 to r25 read back values, for a fault free execution (top) and for an instruction skip (bottom).

Register	16	17	18	19	20	21	22	23	24	25
Fault free	0x39	0x38	0x37	0x36	0x35	0x34	0x33	0x32	0x31	0x30

Register	16	17	18	19	20	21	22	23	24	25
Faulted	0x39	0x38	0x37	0x55	0x36	0x35	0x34	0x33	0x32	0x31

Listing 1 provides a description of the test code we used to tune our settings in order to induce instruction skips. The core part of the test code (delimited by the core trigger signal: lines 5 and 16 in Listing 1) is a series of ten `ld rX,Z+` instructions. Each one corresponds to a load in a destination register `rX` of a byte value stored in RAM memory at address `Z` with a post increment of `Z`. Prior to that, the ten destination registers, `r16` to `r25`, are initialized at `0x55` and an array of ten byte values `0x39` to `0x30` are stored in RAM with `Z` storing the address of its first element. Registers `r16` to `r25` are read back after the synchronization trigger has been cleared. The two top signals in Figure 2 are the synchronization and core triggers, the bottom blue signal is an approximated image of the EM perturbation (obtained by wrapping the wire end of a coaxial line around the wire conducting the voltage pulse to the injection probe). The top part of Table 1 displays the values read back from `r16` to `r25` for a fault-free execution.

As an example, the right column of the test code core part in listing 1 (lines 6 to 15) displays the effect of an EM perturbation turning the `ld` instruction of line 9 into a `nop` instruction. The effect of such an EM-induced instruction skip is highlighted in the bottom part of table 1: the initialization value `0x55` is read back from `r19` (in red), and because an increment of address `Z` is missing, all the values read back from `r20` to `r25` are shifted (in black).

4. Experimental results

4.1. Finding the Points-of-Interest

EMFI has a local effect [9, 30] that may be classified in between the strongly local effect of laser FI [13] and the global effect of other FI means (such as clock and power glitches). As a result, the position of the injection probe w.r.t. the target has a significant effect on the FI process: a fault may be injected or not, different FMs may be obtained [3, 9, 21]. Our research objective was to induce instruction skips similar to that described in [13] for laser FI. This implied to test a large amount of settings: injection probe location, synchronization with the test code, and voltage pulse parameters. Because this search space has several dimensions, this process could be very time consuming, especially regarding the XY position of the injection probe (we refer the reader to [20] for an illustration of the probe location effect; for the other parameters, previous work helped us to converge in a matter of a few hours). It took approximately a week to find the right settings allowing to induce an instruction skip (the injection probe location is shown in Figure 1).

Figure 2 exemplifies the kind of instruction skip that can be observed. It displays the trigger signals of the test code (two top waveforms) and an image of the EM perturbation (third waveform). The trigger signals are both drawn for a fault-free execution (in red, denoted as 'normal operation') and for an EM-injection that induces an instruction skip of the `ld` instruction into register `r19` (in blue). Because execution of a `ld` instruction takes two clock periods contrary to that of a `nop` instruction which takes one clock period, each consecutive instructions skip shall correspond to a reduction of the test code execution time of one clock period. This phenomenon is displayed in Figure 2. The test code execution time is shortened as well as the duration of the trigger signals: the faulted execution triggers in blue last one clock period less than the fault free execution. This experiment was performed with a voltage pulse amplitude set to -200 V and a width of 100 ns (the values read back from the registers were actually those mentioned in the bottom part of Table 1). For these settings a 100 % success rate[1] was achieved (calculated from 1,000 tests).

It assessed the ability of EMFI to induce an instruction skip in a running microcontroller. These settings (in particular the injection probe location) served as a basis for the experiments we carried out to analyze further the EM instruction skip FM (as reported in the following subsections).

4.2. EM-induced instruction skip, test of accuracy

In terms of accuracy, we tested whether the single instruction skip fault model was still valid while targeting the `ld` instruction of the other test registers. Our aim was to assess an attacker ability to target an arbitrary instruction in a program. To do so, we sweeped the time delay between the EM-perturbation and the synchronization trigger signal to span the whole test code. Figure 3 reports the obtained results. It displays the skipped registers (positions 1 to 10 correspond to registers `r19` to `r25`) as a function of the delay. It reveals that an attacker is able to inject EM-induced single instruction skips into a running microcontroller with high timing accuracy. For each instruction, we were indeed

[1] it is calculated, for a given settings, as the ratio of the number of instruction skips to the number of experiments.

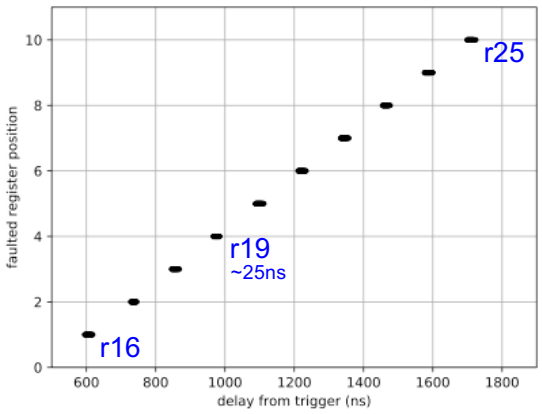


Figure 3: EM-induced single instruction skip: ability to choose the skipped instruction - Faulted registers as a function of EM injection time (ns) for a −200 V voltage amplitude and 100 ns pulse width.

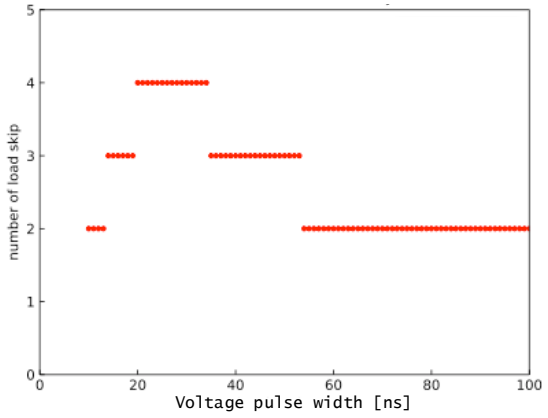


Figure 5: EM-induced instruction skip: effect of the voltage pulse width - Number of EM-induced instruction skips as a function of the voltage pulse width (ns).

able to find an injection timing leading to a 100 % success rate (calculated for 100 experiments for each).

4.3. EM-induced multiple instructions skip

Some experimental results (not reported here) suggested that several consecutive instructions may be skipped simultaneously. In order to verify this assumption, we again carried out our experiments with a voltage pulse amplitude increased to -250 V. The corresponding results are displayed in Figure 4. It shows that this increase makes it possible to skip two consecutive instructions with a still high timing accuracy (i.e. the ability to choose the two instructions that are skipped). As an illustration (denoted in blue in Figure 4), the two successive load instructions into `r19` and `r20` are skipped for a trigger to shot delay of 980 ns. A further increase of the voltage pulse amplitude to −400 V did not change the number of skips (similar results were obtained using positive voltage pulses).

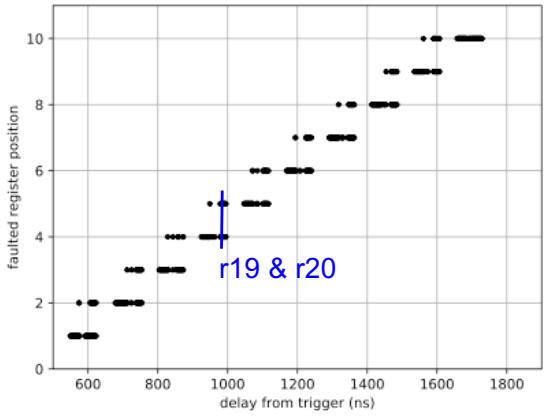


Figure 4: EM-induced instruction skip: ability to skip two consecutive instructions - Faulted registers as a function of EM injection time (ns) (−250 V voltage amplitude and 100 ns pulse width).

Through further testing, we discovered that tuning the width of the voltage pulse (from 10 ns to 100 ns) had an effect on the number of achievable successive skips. The use of different values of the voltage pulse width led to different number of skips as reported in Figure 5. For the same settings (location, voltage amplitude, and delay) but different pulse widths, up to four successive instructions skips were obtained (for a width between 20 ns and 35 ns). The tuning of the width also made it possible to choose precisely the number of skips: 2, 3 or 4. With other settings (location and voltage amplitude), we ascertained the ability to increase to six the number of successive instructions skipped. This ability to skip up to six instructions in a row strengthens the threat associated to the EMFI fault model. It is acknowledged in section 5 which describes how an instruction duplication-based software countermeasure may be defeated.

4.4. EM-induced multiple but separate instruction skips

Voltage pulse generators generally have a repetition rate in the kHz range[2]. Hence, there will be a latency in the microsecond range between two instruction skips, which is a long time at the scale of a microcontroller program. It may be used for designing a software countermeasure based on the assumption that an attacker would not be able to skip two instructions that are placed sufficiently close in a program (i.e. their execution times being in a time interval smaller than one microsecond) but not too close in order to avoid the risk of multiple instructions skip as described in subsection 4.3. It is a potential limitation of the EMFI potency (contrarily to laser FI that do not suffer from this limitation [13]).

However, some voltage pulse generators have a double pulse generation capability: they are able to output two voltage pulses with a latency in the nanosecond range. It is the case for our voltage pulse generator which is able to output

[2]Our pulse generator has a 2 kHz repetition rate, other generators from established manufacturers may be slightly faster with repetition rates of 20 kHz [19] or 1 MHz [26].

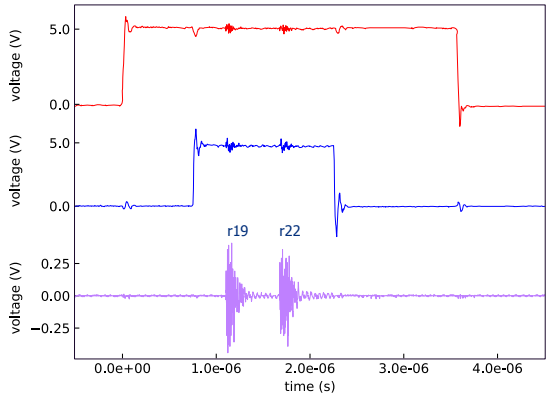


Figure 6: EM-induced instruction skips: inducing two separate instruction skips using two voltage pulses, `ld` instructions into `r19` and `r22` were skipped (from top to bottom, synchronization trigger, core trigger, and approximated images of the EM perturbations).

Table 2
Skip of two instructions: delay (in ns) between the two EM perturbations and reference of the second skipped instruction expressed as the `load` instruction destination register (r19 to r25).

Delay [ns]	225	350	475	600	725	850	975
Register #	19	20	21	22	23	24	25

a second voltage pulse with a minimal delay of 150 ns after the first one. The characteristics of the second voltage pulse (duration and voltage amplitude, see subsection 3.1) can be set independently from that of the first pulse. It makes it possible to skip two separate instructions, with a low latency, by exposing a target to two close EM perturbations.

We carried out these tests on the test code given in Listing 1. Both voltage pulses had the same duration and voltage amplitude settings: 10 ns and 150 V respectively. The first EM perturbation aimed at the `ld r17, Z+` instruction (line 7 of Listing 1) which was successfully skipped. The second EM perturbation was outputted with a delay (measured from the onset of the first perturbation) varying between 150 ns (its minimal achievable value) and 1,000 ns with a time increment of 5 ns (the duration of a `ld` instruction). Table 2 reports timing delays that made it possible to skip the `ld rx, Z+` instructions (line 9 to 15 of Listing 1) into registers `r19` to `r25` with a 100 % success rate. These delays are spaced out by a 125 ns time interval which is logically equal to the 125 ns execution time of a `ld` instruction (two clock periods).

The first EM perturbation induced a skip of the `ld r17, Z+` instruction; due to the minimal 150 ns delay between the two voltage pulses it was not possible to skip the following `ld r18, Z+` instruction. However, the next following instruction (a `ld` into `r19`) was successfully skipped for a 225 ns delay. These results underline that EMFI is able to target simultaneously separate but close instructions. Figure 6 is an illustration of this property for a time delay of 600 ns between the two EM perturbations: the `ld` instructions into registers `r19` and `r22` were skipped. In addition, because both voltage pulses (at the root cause of the EM perturbations) can be set independently with their full duration and amplitude parameter ranges, it is possible to induce two series of several successive instruction skips (as reported in subsection 4.3) while not disturbing the instructions executed in between. This provides a potential attacker with a lot of flexibility.

4.5. PIN bypass with EM perturbation

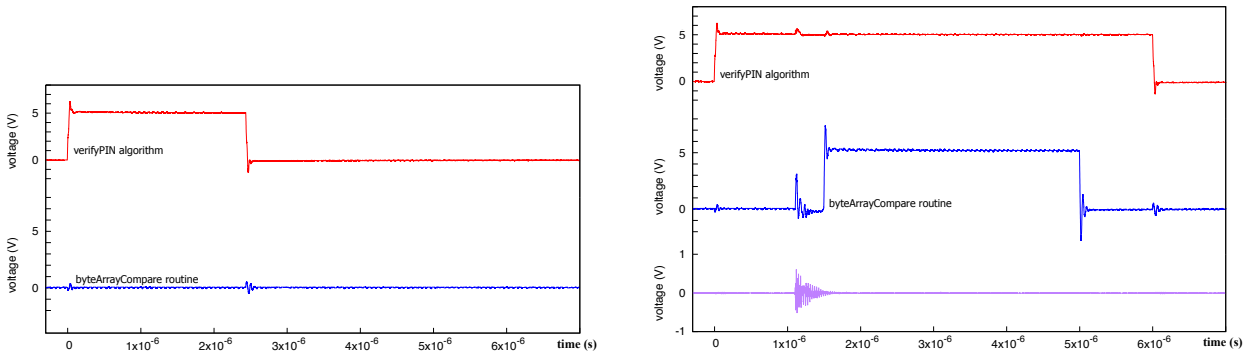
In order to improve our assessment of the threat raised by EM-induced instruction skips, we targeted a 4-digit PIN verification algorithm (denoted `verifyPIN` hereafter, and described in [11]). It is protected against brute force attacks (i.e. attacks that consist in testing all existing PIN codes) thanks to a PIN trial counter (denoted by `g_ptc` hereafter). The value of this counter is tested before calling the PIN verification routine: `byteArrayCompare` (this code routine is in charge of comparing the user PIN to a reference PIN). `g_ptc` is decremented and memorized before each verification. It is initialized at 3 (and restored to this value after a successful identification). If `g_ptc` reaches zero, a conditional `if` statement denies the entry into the `byteArrayCompare` routine. As a result, the verification systematically fails. A part of the (simplified) pseudo-code of this algorithm is shown in Listing 2, where `BOOL_TRUE` and `BOOL_FALSE` are respectively the true and false boolean values, and `g_auth` a variable indicating the user status (`g_auth` set to `BOOL_TRUE` indicates that the user has been successfully authenticated). The `g_auth` and `g_ptc` variables are updated if the verification routine `byteArrayCompare` succeeds. This routine takes two arguments: `g_userPin` and `g_cardPin` (the addresses of the user pin and of the secret pin respectively).

```
1  BOOL g_auth = BOOL_FALSE;
2  ...
3  g_ptc--;
4  if(g_ptc > 0){
5      if(byteArrayCompare(...) == 1){
6          g_ptc = 3;
7          g_auth = BOOL_TRUE;
8      }
9  }
```

Listing 2: C code of the `verifyPIN` algorithm.

[11] describes several instruction skip attacks that may result in a successful PIN bypass: most of them targeting one or a few successive code instructions. We chose to implement that which consists in skipping the verification of the trial counter (line 4 in Listing 2), allowing us to perform a bruteforce attack on the four digits of the secret PIN code.

Figure 7 displays the trigger signals highlighting the execution of the `verifyPIN` algorithm in red, and of the `byteArray-Compare` routine in blue. When `g_ptc` equals zero (Figure 7.a), the access to the `byteArrayCompare` routine is denied (the corresponding trigger stays low). Figure 7.b reports a successful EMFI attack (an image of the EM perturbation is de-



(a) PIN trial counter at zero.

(b) EM-forced entry into the byteArrayCompare routine.

Figure 7: Illustration of a PIN brute force bypass through EM-induced instruction skips.

picted in purple) that forces the entry into the `byteArrayCompare` routine: the PIN code entered by the attacker is checked against the reference PIN despite the exhaustion of the PIN counter. We were able to find settings that achieve a 100% success rate and then to brute force the `verifyPIN` algorithm.

5. Test of the extended EM fault model on a duplication-based software countermeasure

The extended EMFI fault model we ascertained in section 4 questions the relevance of countermeasures based on the assumption that EMFI attacks are limited to single instruction skips. This section describes the software countermeasure using instruction redundancy we used to harden a PIN code verification algorithm against EMFI. It also reports its efficiency when tested against multiple instructions skip EMFI.

5.1. Instruction duplication-based software countermeasure

Moro et al. [23, 22] introduced a software countermeasure against EMFI based on the assumption that an attacker can only induce a single instruction skip. It consists in duplicating the sensitive instructions of a program. Applying a software countermeasure to a program requires that its functionality remains identical when it is under attack and when it runs undisturbed. This requirement is fulfilled when duplicating idempotent instructions (i.e. instructions that can be executed several times without changing the result obtained by its first execution). Comparison and branch instructions are idempotent and can be duplicated easily, which is of interest to hardened the execution flow of a program against fault injection attacks. The hardening of a non-idempotent instruction requires to substitute an equivalent idempotent instruction or (if possible) to rewrite part of the code with idempotent instructions [23, 22]. This software countermeasure comes at a price, it involves an overhead both in execution time and in code size that are more than doubled.

Instruction duplication provides an effective hardening against the single instruction skip fault model: the code functionality is left unchanged by any single instruction skip because its duplicated twin is still executed. As mentioned previously, it is well suited for hardening the conditional branches of a program (e.g. as the PIN code trial counter test bypassed through EMFI reported in figure 7). We applied this countermeasure methodology to our PIN verification algorithm and tested its efficiency against the extended EMFI fault model we reported in section 4.

5.2. PIN verification algorithm hardened against single instruction skip attacks

Listing 3 provides the assembler language version of the PIN code verification algorithm (denoted `verifyPINasm()`); several instructions were omitted for concision (they are marked by left-aligned comments, while tabulated comments refer to the following block of instructions).

This `verifyPINasm()` routine is the basis we used to write a duplication-based hardened code: `verifyPINhardened()`. For the sake of concision, we provide in Listing 4 three subparts of this code corresponding to three attack paths an attacker may use. Hereafter, we refer to these attack paths by:

1. Attack 1: the brute force attack reported in subsection 4.5, it consist in skipping the conditional branch instruction `brge label_end_verifyPINasm` of line 9 in Listing 3.
2. Attack 2: forcing authentication with an incorrect PIN code by skipping the `brne .+6` instruction of line 24 in Listing 3. As a result, authentication is granted to the attacker (`ldi r16, 0xAA`) despite a negative comparison of the user entered PIN against the reference PIN.
3. Attack 3: implementing a brute force attack by skipping the `sts (g_ptc), r20` instruction of line 31 in Listing 3 which is in charge of storing the decremented PIN trial counter after an incorrect PIN code has been tested.

The first hardened code part (lines 3 to 6 of Listing 4) consists in duplicating the test of `g_ptc` (stored in registers `r20` and `r21`) and the corresponding branch instructions (taken when `g_ptc` equals zero). It provides a countermeasure against a brute force attack (attack 1) due to the single skip of any of these instructions.

```
1  # Set synchronization trigger
2  # Save registers content (push)
3  ldi r16, 0x55        # init status (r16) at BOOL_FALSE (0x55)
4  sts (g_auth),r16       # set g_auth. at BOOL_FALSE
5  ldi r17, 0x55        # init diff (r17) at BOOL_FALSE
6    # test g_ptc, if smaller or equal to 0 then end verifyPIN
7  lds r20, (g_ptc)
8  cp r1, r20
9  brge label_end_verifyPINasm
10 # Set core trigger
11 movw r24,r30
12 adiw r24,4
13   # Comparison loop
14 label_comp_loop:
15 ld r18,Z+
16 ld r19,X+
17 cpse r18,r19
18 ldi r17,0xAA        # Set diff to TRUE if characters differ
19   # test end of loop
20 cp r30, r24
21 cpc r31, r25
22 brne label_comp_loop
23   # Test diff, if FALSE then set status at TRUE else stay FALSE
24 cpi r17, 0x55
25 brne  .+6
26 ldi r16, 0xAA        # Set status to TRUE
27 ldi r20, 0x03        # Set g_ptc to 0x03
28 rjmp .+4
29 cpse r1, r20        # Test g_ptc = 0 to avoid g_ptc = FF
30 subi r20, 0x01        # g_ptc--
31 sts (g_ptc), r20        # Store g_ptc
32 sts (g_auth),r16        # Store authentication variable
33 # Clear core trigger
34 label_end_verifyPINasm:
35 # Restore registers content (pop)
36 # Clear synchronization trigger
```

Listing 3: `verifyPINasm()` routine.

```
1  ...
2    # test of g_ptc, hardening against attack 1
3  cp r1, r20
4  brge label_end_verifyPINhardened
5  cp r1, r21
6  brge label_end_verifyPINhardened
7   ...
8    # Test diff, if FALSE do not branch and set STATUS at TRUE
9    # Hardening against attack 2
10 cpi r17, 0x55
11 brne  .+8
12 cpi r17, 0x55
13 brne  .+4
14 ...
15    # Store g_ptc, hardening against attack 3
16 sts (g_ptc), r20
17 sts (g_ptc), r20
18 ...
```

Listing 4: Subparts of the `verifyPINhardened()` routine using a countermeasure based on instruction duplication [23].

The second hardened code part (lines 10 to 13 of Listing 4) provides a hardening against attack 2 by duplicating the test of the `diff` variable (`r17`) and the associated branch instructions.

The third hardened code part (lines 16 and 17 of Listing 4) consists in a simple duplication of the store instruction of `g_ptc`, this is a countermeasure against attack 3.

This third countermeasure uses two identical duplicated instructions that are adjacent. It is immune to a single instruction skip, however it is ineffective against a double instruction skip similar to those we report in subsection 4.3. The two other countermeasures combines an interleaving of the different duplicated instructions. This provides the same protection against a single instruction skip and also provides immunity against double instruction skips. A successful attack would require to skip three successive instructions (e.g. those of lines 4 to 6 in Listing 4) which may be considered more difficult to achieve than a double instruction skip.

5.3. Attacking an instruction duplication-based software CM, experimental results

The extended EMFI fault model, that makes it possible to skip several successive instructions, was then experimentally tested against our duplication-based hardened PIN verification algorithm. We used the same EMFI test bench (described in subsection 3.1) but a different voltage generator. It has a tunable pulse duration range of 10-200 ns and a maximum voltage amplitude of ±200 V.

With a voltage pulse width of 100 ns and a 100 V amplitude (well suited for inducing single instruction skips), we were able to implement attacks 1, 2, and 3, with a 100 % success rate on the unprotected `verifyPINasm()` code (for delays set respectively to 1210 ns, 4270 ns, and 4530 ns). Using the same voltage pulse width and amplitude, we were unable to succeed in implementing the same three attacks on the hardened `verifyPINhardened()` code. This proves the efficiency of instruction duplication against single instruction skips. However, by using pulse settings producing multiple skips, we were able to carry out the attacks successfully as reported in the following subsections.

5.3.1. *Implementing attack 1 with multiple instructions skip*

The voltage pulse width and amplitude were set to 20 ns and 200 V to induce a multiple instructions skip with a single EM perturbation. During our experiments, we were able to force repeatedly the entry into the PIN comparison loop of `verifyPINhardened()` despite a `g_ptc` counter equal to zero. However, we were unable to find an EMFI settings that would assure a 100 % repetition rate (it was rather in the 40-60 % success rate depending on the test series and settings). Nonetheless, such a 100 % success rate was achieved when considering a set of several tests performed with a delay varying between 1450 ns and 1510 ns with a time step of 10 ns. This is a suitable strategy to perform a brute force attack (it involves more tests but still guarantees the success of the first attack).

5.3.2. *Implementing attack 2 and 3 with multiple instruction skips*

Attack 2 and 3 proved even more difficult to succeed in. Despite the fact that we were indeed several times successful in performing these attacks (for delays in the 4650-4710 ns and 4840-4970 ns ranges respectively), we were unable to

find settings associated to a significant success rate. We also found no significant difference in the success rates of attacks 2 and 3 despite the fact that involved respectively the skip of 3 and 2 successive instructions.

Considering attacks 2 and 3 that both require that the PIN trial counter is not exhausted (we regularly reset `g_ptc` to 3 during our experiments for convenience purposes), the implementation of both attacks shall be considered unsuccessful. On the other hand, the countermeasure shall be considered broken (regarding attack 2 and 3) because several attacks were successful (in addition, progress in EMFI settings may provide a significant success rate in the future).

6. Discussion

6.1. EM-induced instruction skip fault model

The experimental results reported in sections 4 and 5 demonstrate that a very high accuracy is achievable with EM-induced instruction skips: we were able to choose and skip a single instruction of a test sequence with a 100 % success rate. Moreover, we were able to increase its extent to skip six successive instructions which adds to its strengh (unlike [27, 29], the ability to skip consecutive instructions was not linked to the micro-architecture of the target). From an attacker perspective, this attack does not require any insight on the underlying micro-architecture, which makes it easily reproducible. This FM has been successfully applied to brute force a PIN verification algorithm. We demonstrate on a 8-bit microcrontroller that several consecutive fetch operations can be skipped with a single EM pulse, extending the current FMs.

This fault model shares similarities with laser-induced instruction skips in terms of accuracy and repeatability under the same synchronization requirements. However, the number of consecutive instructions which can be skipped is limited with the EM injection technique, whereas an arbitrary number of instruction can be blinded with laser injection [13]. This difference might be understood as a difference in the nature of the stress induced by laser and EM injection techniques. The former can be precisely tuned with the laser power and pulse duration, whereas the latter only occurs during the rising edge of the EM pulse. While software CMs against EM-injection may take advantage of this temporal limitation, the number of consecutive instructions which can be skipped might be higher than the number assessed in this article.

Finally, we underline the importance of synchronization of the EM injection with the target code to achieve the high success rate we report. We also refer the reader to [13] for a discussion on how synchronization may be obtained.

6.2. Discussion of the EM-induced fault injection mechanism

Several physical mechanisms have been proposed to describe the faults induced by an EM pulse [9, 23, 24]. The authors of [9] and [23] highlighted the similarities between EM induced faults in an IC and faults that were obtained with the violation of timing constraints. A model of the glitch induced by an EM pulse on a power-ground network was later proposed in [10]. The authors of [24] suggested that glitching the set and reset circuitry corrupts the operation of D-flip-flop memory cells, thus explaining the specificity of the timing of EM injections. In the same time, the authors of [16] measured and analyzed the effets of an EM pulse on the clock circuitry of a FPGA. The diversity of suggested hypothesis reflects the complexity of the analysis of EM physical effects.

In this article, we observed a non-linear correlation between the width of the EM pulse and the number of skipped instructions. This result shares some similarities with [31], where damped oscillation were obtained by glitching a FPGA power supply with a voltage pulse. The authors observed constructive and destructive interferences for various duration of the pulse. Moreover, positive and negative polarities of the voltage pulse were shown to have the same effects, as this is the case with our experiments. These elements support the hypothesis that we observed violation of timing constraints in our experiments (similarly to the timing constraint violations observed in [31]). In addition [15] reports that EMFI sensitivity is higher at lower supply voltage or higher frequency of the target, which increases the propagation delays of its digital gates (a result we also observed on other targets). This is consistent with a fault injection mechanism related to timing constraint violations [31].

7. Conclusion

This research work assesses on experimental basis an extended fault model for EM-induced instruction skips. The main characteristics of this fault model are:

- its accuracy, or ability to choose the skipped instruction with a 100 % success rate provided that a precise synchronization is obtained,
- its extension, or ability to skip up to six successive instructions (without the help of any micro-architectural effect),
- its flexibility, with the ability to skip several separate instructions in a small time interval (using a voltage pulse generator able to output two close voltages pulses).

We also highlighted the influence of the width of the voltage pulse, which has been underestimated until now, in the properties of EM-induced faults.

Simply put, EMFI may offer an attacker the ability to erase a small part (or two small parts) made of several instructions of a microcontroller firmware at runtime. This extended capability was ascertained successfully against a software countermeasure based on instructions duplication.

Though, this EM fault model is not as threatening as the laser FM that offers the ability to erase several chosen parts of arbitrary length of a running program [13]. It nonetheless extends the threat model that shall be considered when securing microcontrollers against FAs as EMFI is more common and affordable than laser FI.

Acknowledgment

This research has been partially supported by the European Commission under H2020 SPARTA (Grant Ag. 830892).

References

[1] Balasch, J., Gierlichs, B., Verbauwhede, I., 2011. An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs, in: Fault Diagnosis and Tolerance in Cryptography.

[2] Barenghi, A., Breveglieri, L., Koren, I., Naccache, D., 2012. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. Proceedings of the IEEE 100, 3056 – 3076.

[3] Beckers, A., Balasch, J., Gierlichs, B., Verbauwhede, I., Osuka, S., Kinugawa, M., Fujimoto, D., Hayashi, Y., 2019. Characterization of EM faults on ATmega328P, in: International Symposium on Electromagnetic Compatibility, IEEE.

[4] Boneh, D., DeMillo, R.A., Lipton, R.J., 1997. On the importance of checking cryptographic protocols for faults, in: Advances in Cryptology, International Conference on the Theory and Application of Cryptographic Techniques.

[5] Breier, J., Jap, D., Chen, C.N., 2015. Laser profiling for the back-side fault attacks: With a practical laser skip instruction attack on AES, in: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, New York, NY, USA.

[6] Buchner, S., Miller, F., Pouget, V., McMorrow, D., 2013. Pulsed-laser testing for single-event effects investigations. Nuclear Science, IEEE Transactions on 60, 1852–1875.

[7] Cojocar, L., Papagiannopoulos, K., Timmers, N., 2017. Instruction duplication: Leaky and not too fault-tolerant!, in: Eisenbarth, T., Teglia, Y. (Eds.), Smart Card Research and Advanced Applications, Springer. pp. 160–179.

[8] Cui, A., Housley, R., 2017. BADFET: Defeating modern secure boot using second-order pulsed electromagnetic fault injection, in: USENIX Workshop on Offensive Technologies (WOOT 17), USENIX Association, Vancouver, BC. URL: https://www.usenix.org/conference/woot17/workshop-program/presentation/cui.

[9] Dehbaoui, A., Dutertre, J.M., Robisson, B., Tria, A., 2012. Electromagnetic transient faults injection on a hardware and a software implementations of AES, in: Bertoni, G., Gierlichs, B. (Eds.), Workshop on Fault Diagnosis and Tolerance in Cryptography, IEEE Computer Society, Leuven, Belgium. pp. 7–15.

[10] Dumont, M., Maurine, P., Lisart, M., 2019. Electromagnetic fault injection: how faults occur, in: 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC).

[11] Dureuil, L., Petiot, G., Potet, M.L., Le, T.H., Crohen, A., de Choudens, P., 2016. FISSC: A fault injection and simulation secure collection, in: International Conference on Computer Safety, Reliability, and Security.

[12] Dutertre, J.M., Beroulle, V., Candelier, P., De Castro, S., Faber, L.B., Flottes, M.L., Gendrier, P., Hély, D., Leveugle, R., Maistri, P., Di Natale, G., Papadimitriou, A., Rouzeyre, B., 2018. Laser fault injection at the CMOS 28 nm technology node: an analysis of the fault model,, in: 2018 Workshop on Fault Diagnosis and Tolerance in Cryptography.

[13] Dutertre, J.M., Riom, T., Potin, O., Rigaud, J.B., 2019. Experimental analysis of the laser-induced instruction skip fault model, in: Askarov, A., Hansen, R.R., Rafnsson, W. (Eds.), The 24th Nordic Conference on Secure IT Systems, Nordsec 2019, Springer International Publishing, Cham. pp. 221–237.

[14] Elmohr, M., 2020. Embedded Systems Security: On EM Fault Injection on RISC-V and BR/TBR PUF Design on FPGA. Master's thesis. University of Waterloo. Waterloo, Canada.

[15] Elmohr, M.A., Liao, H., Gebotys, C.H., 2020. Em fault injection on arm and risc-v, in: 2020 21st International Symposium on Quality Electronic Design (ISQED), pp. 206–212.

[16] Ghodrati, M., Yuce, B., Gujar, S., Deshpande, C., Nazhandali, L., Schaumont, P., 2018. Inducing local timing fault through EM injection, in: Proceedings - Design Automation Conference, ACM, San Francisco, CA, USA. pp. 142:1–142:6.

[17] Kumar, D., Beckers, A., Balasch, J., Gierlichs, B., Verbauwhede, I., 2018. An in-depth and black-box characterization of the effects of laser pulses on ATmega328P, in: Smart Card Research and Advanced Applications.

[18] Lacruche, M., Borrel, N., Champeix, C., Roscian, C., Sarafianos, A., Rigaud, J.B., Dutertre, J.M., Kussener, E., 2015. Laser fault injection into SRAM cells: Picosecond versus nanosecond pulses, in: On-Line Testing Symposium.

[19] Länger, . IC EM pulse injection langer pulse. URL: https://www.langer-emv.de/en/product/ic-side-channel-analysis/94/ici-03-1-eft-set-ic-em-pulse-injection-langer-pulse/1166.

[20] Menu, A., Bhasin, S., Dutertre, J., Rigaud, J., Danger, J., 2019. Precise spatio-temporal electromagnetic fault injections on data transfers, in: 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2019, Atlanta, GA, USA, August 24, 2019, pp. 1–8.

[21] Moro, N., Dehbaoui, A., Heydemann, K., Robisson, B., Encrenaz, E., 2013. Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller, in: Fischer, W., Schmidt, J.M. (Eds.), Workshop on Fault Diagnosis and Tolerance in Cryptography, IEEE Computer Society, Los Alamitos, California, USA. pp. 77–88.

[22] Moro, N., Heydemann, K., Dehbaoui, A., Robisson, B., Encrenaz, E., 2014a. Experimental evaluation of two software countermeasures against fault attacks, in: Hardware-Oriented Security and Trust.

[23] Moro, N., Heydemann, K., Encrenaz, E., Robisson, B., 2014b. Formal verification of a software countermeasure against instruction skip attacks. Journal of Cryptographic Engineering 4, 145–156.

[24] Ordas, S., Guillaume-Sage, L., Maurine, P., 2017. Electromagnetic fault injection: the curse of flip-flops. Journal of Cryptographic Engineering 7, 183–197.

[25] Piret, G., Quisquater, J.J., 2003. A differential fault attack technique against SPN structures, with application to the AES and Khazad, in: Cryptographic Hardware and Embedded Systems, Berlin, Heidelberg.

[26] Riscure, . EM-FI transient probe. URL: https://getquote.riscure.com/picdb/filedb/3792/EM-FI%20datasheet.pdf.

[27] Rivière, L., Najm, Z., Rauzy, P., Danger, J.L., Bringer, J., 2015. High precision fault attacks on the instruction cache of ARMv7-M architectures, in: 2015 IEEE International Symposium on Hardware Oriented Security and Trust.

[28] Trichina, E., Korkikyan, R., 2010. Multi fault laser attacks on protected CRT-RSA, in: Fault Diagnosis and Tolerance in Cryptography.

[29] Yuce, B., Ghalaty, N.F., Santapuri, H., Deshpande, C., Patrick, C., Schaumont, P., 2016. Software fault resistance is futile: Effective single-glitch attacks, in: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography.

[30] Zussa, L., Dehbaoui, A., Tobich, K., Dutertre, J.M., Maurine, P., Guillaume-Sage, L., Clédiere, J., Tria, A., 2014a. Efficiency of a glitch detector against electromagnetic fault injection, in: Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, pp. 1–6.

[31] Zussa, L., Dutertre, J.M., Clédiere, J., Robisson, B., 2014b. Analysis of the fault injection mechanism related to negative and positive power supply glitches using an on-chip voltmeter, in: Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on.