Multiple and Reproducible Fault Models on Micro-Controller using Electromagnetic Fault Injection

Vanthanh Khuat, Oualid Trabelsi, Laurent Sauvage, Jean-Luc Danger

Multiple and Reproducible Fault Models on Micro-controller using Electromagnetic Fault Injection

Vanthanh Khuat*†, Oualid Trabelsi*, Laurent Sauvage* and Jean-Luc Danger*
*LTCI, Télécom Paris, Institut polytechnique de Paris, France
Email: {khuat, oualid.trabelsi, laurent.sauvage, jean-luc.danger}@telecom-paris.fr
†Faculty of Information Technology, Le Quy Don Technical University, Hanoi, Vietnam
Email: van-thanh.khuat@lqdtu.edu.vn

***Abstract*—In this paper, we present a method to obtain multiple and reproducible fault models on a 32-bit Micro-controller (MCU) using Electromagnetic Fault Injection (EMFI). By using different Pulse Width (PW), this method allows to obtain either a replay or skip of instructions fault model with a fault rate up to 100%. Specifically, a replay of an instruction block is obtained with the PW of 1.5 nano second (ns), whereas a skip of an instruction block is observed with the PW of 7.0 ns. With these types of fault model, an adversary may be able to retrieve secret information, as cryptographic key, by using efficient attacks. The study is carried out by enabling or disabling the cache. The only difference is that the resulting faulty block is either 32 bits when the cache is disabled or 64 bits when the cache is enabled. The impact of the Pulse Amplitude (PA) has been analyzed, and the fault model has been characterized at bit level. These results demonstrate the efficiency and the flexibility of the EMFI which should be considered for designing robust MCU.**

***Index Terms*—Electromagnetic fault injection, Fault models, Characterization, Micro-controller.**

I. Introduction

In recent years, with the development of the Internet of Things (IoT), a large number of embedded systems have been deployed for sensing, collecting data, and connecting to the server. Their architecture relies on MCUs which process plenty of valuable information such as password, account number, identity, critical data, etc. These objects are the target of many attack types. As they are physically accessible, the physical attack becomes possible in addition to cyber attacks. One of the most powerful threats is the Fault Injection (FI).

FI is an active side-channel attack in which the attacker induces faults to the target to further exploit them in order to extract secret information by differential analysis (fault vs no-fault). Many FI techniques have been developed, such as clock or voltage tampering [1], [2], EMFI [3], [12], optical fault injection [7], [13]. The attacks can be classified into invasive, semi-invasive, and non-invasive attacks. EMFI is a non-invasive technique, in which the electromagnetic (EM) pulse is used to divert the program from its proper functioning. A probe is specifically designed for conducting EM waves to be coupled into the device and disturbs the operation inside it. There are several advantages of EMFI. First, because the Electromagnetic (EM) pulse can be confined in a small space, it only affects the device locally as compared to voltage or clock tampering which has a global impact on the device. Second, though the space resolution of the EMFI is not as high as that of the Laser Fault Injection (LFI) [6], in EMFI, due to the nature of EM pulse, it is possible to attack the MCU without depackaging it. In addition, because the front-side of the MCU is protected with several metal layers which can absorb and reflect the light, normally LFI must be performed from back-side of the MCU. However, most of the available works using EMFI were able to attack the devices from its front-side, which is beneficial in many practical applications.

The EMFI has been proven to be a very effective tool for FI in MCUs including 8-bit MCU [9] and 32-bit MCU [10]. At logical level, the fault includes bit-set, bit-reset, bit-flip and no sampling. At software level, the fault models which characterizes the main scheme of the Fault Injection Attacks (FIA) are: instruction replay, instruction skip, instruction replacement, and register fault. They are also classified into single and multiple instruction(s) faults. In single instruction fault model, EM impacts only one instruction. Effective countermeasure has been proposed against it [11]. In multiple instructions fault model, two or more instructions are impacted by EMFI. This fault model is more complex to obtain, and so as the countermeasure to thwart it.

Most of the available works on EMFI focused on how to achieve a fault model with a high fault rate. To our best knowledge, no studies have reported on obtaining multiple fault models with a perfect fault rate. In this paper, we report on the observation of multiple and reproducible fault models on a 32-bit MCU, in which for a same targeted block of instructions, the adversary can choose the fault model between multiple instructions replay and multiple instructions skip both with a perfect fault rate. The main contributions of this paper are:

- investigating the impact of the PW on the fault induced by EM pulse to obtain two fault models on multiple instructions;
- determining the injection time to achieve a fault reproducibility of 100%;

- investigating the impact of the PA on the skip and replay fault models induced by EM pulse, respectively;
- characterizing the fault model at bit level.

The rest of the paper is organized as follows. Section II describes the experiment setup and target. Section III discusses the two fault models of replay and skip of block instructions achieved with EMFI by using different PW. Section IV discusses the impact of the PA on the fault rate of each observed fault model. Section V characterizes the fault on buffer content at bit level. Section VI provides the main conclusions and perspectives.

II. Experimental setup

Fig. 1 shows the experimental setup used for conducting EMFI on SAMD21G18A MCU which consists of an oscilloscope, a pulse generator, an amplifier, a probe, a computer, and the device under test. The pulse generator is the Keysight Pulse generator 81160A, which is capable of generating a pulse as short as 1.5 ns and a rising time of 1.0 ns. Its output

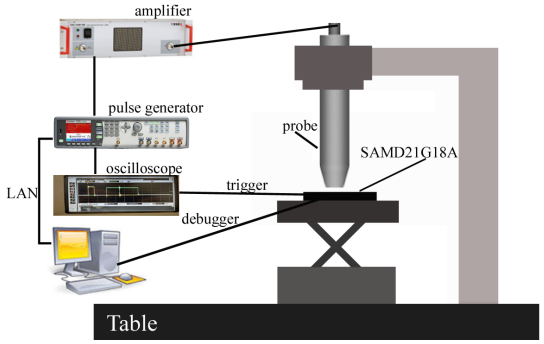

Figure 1. Schematic of EMFI experimental setup

is redirected to the CBA 400M-260 Power Amplifier, which delivers the pulse to a handmade probe. The later is built with four loop turns of a 150 µm wire, around a ferrite core designed as a circular truncated cone, with a top diameter of 1.5 mm and a bottom diameter of 0.8 mm. The oscilloscope was used to monitor the trigger related to the test code execution time, as well as a second trigger signal generated from the target, and redirected it to the pulse generator.

Our target is the 32-bit MCU SAMD21G18A [8], embedding an ARM Cortex-M0+ core (2-stage pipeline), that implements the ARMV6 thumb instruction set. For all the tests, the MCU was configured to work at 12 MHz, with zero wait states to ensure no delay during the data read operation from the Flash memory. The MCU is equipped with an 8 lines of 64-bit cache, and the data transfer is performed via 32-bit AHB and APB buses. The EMFI induced faults were studied with both cache enabled and disabled. The MCU debug was performed through the Atmel-ICE Debugger which allows collecting all the needed registers and memory data.

A manual scan of all the chip package was performed to find the sensitive positions. The probe was then fixed at an optimal position that provides the highest fault rate. All the registers were initialized to a known value at the beginning of all the tests to ensure the fault traceability. One test iteration follows three main steps: (1) the target is reset and all systems registers are initialized; (2) the trigger for the pulse generator is set, and the test code is executed; (3) all the registers value are collected as the program reaches the configured breakpoint, or when an interrupt routine is performed.

During the campaign, 100 tests were performed for each configuration of fault injection parameters. Before each test, we collected the value of all the registers to confirm the that the program functions correctly in the normal condition and used it as the reference to detect the fault after each EMFI. More details on experiment setup and the method for applying the EMFI can be found in [12].

Our analysis focused mainly on the observed faults through the variation of setup parameters, such as the injection time, the PW, and the PA.

III. Impact of the PW on the EMFI-induced fault models

To understand the impact of the PW to the fault obtained by EMFI, pulses with different PW, from 1.5 ns to 9.0 ns with an increment of 0.5 ns, were used to induce fault on the MCU, while all the other parameters such as the probe position and the PA were fixed. The faults were induced in MCU with cache disabled and cache enabled, respectively.

A. Cache disabled

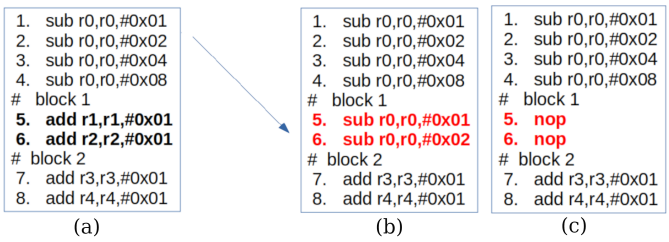

Figure 2. Demonstrations of the replay and skip instructions with cache disabled, (a) testcode, (b) replay of two instructions, (c) skip of two instructions

The main part of our test code, which consists of eight instructions, is shown in Fig. 2(a). For convenience, the instructions are denoted as (i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8); block1 is (i_5, i_6); block2 is (i_7, i_8). As the cache is disabled, 32-bit data corresponding to two 16-bit instructions are loaded from the Flash every two clock cycles. EM pulse was used to target the loads of block1 and block2. It should be pointed out that to differentiate the replay and skip fault model, (i_1, i_2, i_3, i_4) in the test code must not be (`nop,nop,nop,nop`), because the replay and skip of a block of `nop` instructions are equivalent. Here, for convenience of post processing, instructions: `sub r0,r0,#value` (with `value` being `0x01`, `0x02`, `0x04`, `0x08`) were used for (i_1, i_2, i_3, i_4); and (i_5, i_6, i_7, i_8) are simply the operations to `add 0x01` to a register.

The faults are classified into the following types:

- **replay** $i_1 i_2$: (i_5, i_6) being replaced by (i_1, i_2);

- **skip** $i_a i_b$: (i_a, i_b) being replaced by instructions equivalent to (`nop`, `nop`), with a, b being 5, 6, 7, 8;
- **other**: register fault, system fault, instruction replacement, up to three instructions skip, etc.

The effects of replay of two instructions and skip of two instructions are shown in Fig. 2(b) and (c). It is worth noting that, in the replay of two instructions, (i_5, i_6) are replaced by (i_1, i_2), not (i_3, i_4) which are right above them. While in the skip of two instructions, (i_5, i_6) are replaced by two instructions equivalent to (`nop,nop`).

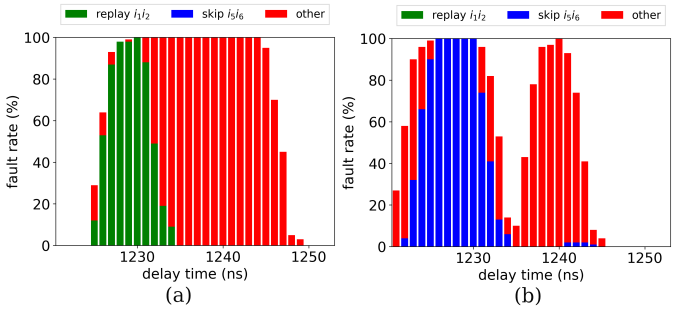

Figure 3. Two different fault models induced by EMFI on block1 with a perfect fault rate (a) Replay of two instructions with PW of 1.5 ns, (b) Skip of two instructions with PW of 7.0 ns

The result shows that the PW has a decisive impact on the fault type or fault model induced by EMFI. As the PW changes, the fault rate for each type of fault changes accordingly. And it is very interesting that at almost the same injection periods, two multiple instructions fault models namely replay and skip of two instructions are achieved, both with a perfect fault rate of 100%. With the PW of 1.5 ns, the faults are mainly replay of two instructions and system fault, and the replay fault rate is up to 100% as shown in Fig. 3(a). While for the PW from 2.0 ns to 9.0 ns, no replay of instructions is observed, the faults are mainly instruction replacement, instruction skip, and other fault. Specifically, with the PW of 7.0 ns, skip of two instructions with fault rate up to 100% is obtained as shown in Fig. 3(b).

It can be seen from Fig. 3 that the adversary is able to choose between the two fault models of replay and skip of two instructions both with a perfect fault rate by only changing the PW. This makes the attack more flexible and threatening.

The fault on block2 is different from the fault on block1 in several ways. For one thing, no replay of instructions is observed with the PW of 1.5 ns as shown in Fig. 4(a). For the other, the distributions of the faults are also different. However, with the PW of 7.0 ns, the skip of two instructions with the fault rate up to 100% is still achieved as shown in Fig. 4(b).

Further investigation shows that the fault behavior above repeats every four clock cycles for each block of two instructions. Because the SAMD21G18A implements ARM Cortex-M0+, which has a 32-bit bus, with two stages pipeline, and every two clock cycles the processor fetches 32 bits data (2×16-bit instructions). Therefore, we conclude that there exists two 32-bit buffers: buffer1 and buffer2 at the Flash

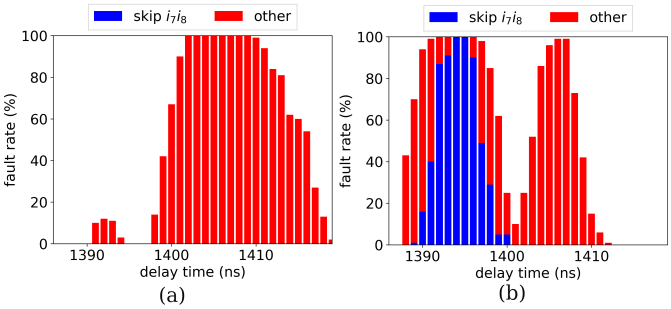

Figure 4. Fault induced by EMFI on block2 (a) 1.5 ns, (b) Skip of two instructions with a perfect fault rate as the PW of 7.0 ns was used.

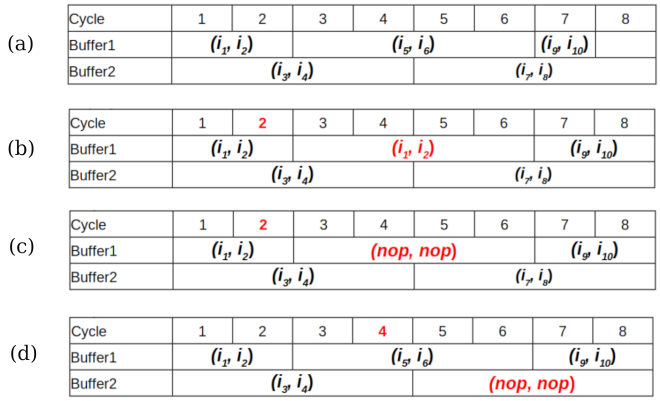

Figure 5. Hypothesis on EMFI-induced different fault models on SAMD21G18A with cache disabled, (a) normal execution instruction loading process, (b) EMFI-induced replay of two instructions on buffer1, (c) EMFI-induced skip of two instructions on buffer1, (d) EMFI-induced skip of two instructions on buffer2

interface, and each buffer is updated with new data every four clock cycles as shown in Fig. 5(a).

In the first case, during the clock cycle 2, the content of buffer1 is (i_1, i_2), and is supposed to be updated with (i_5, i_6), however as disturbed by the short EM pulse, the buffer1 fails to update; as the result, (i_1, i_2) are replayed as shown in Fig. 5(b). While subjected to the EM pulse with a longer PW, the content of buffer is updated with the corrupted data, resulting into instructions modification. In case, the modified instructions are not recognized by the Processor Core (PC), they are turned into instructions equivalent to `nop` instructions, resulting into skip of instructions as shown in Fig. 5(c). The fault behavior can be explained by considering the sampling fault model [4], [5]. According to which the fault happens as the EM pulse is injected during the sampling windows, affecting one or more inputs of the D flip-flop (DFF) including (set, reset, clock, D). The different fault models observed here certainly correspond at physical level to a sampling fault for the replay, and bit modification for the skip. The exact physical phenomenon depending on the PW needs to be deeply analyzed in future works.

The same principle is applied for the buffer2. However, for buffer2, the replay is not observed, though skip of two

instructions is still observed. Fig. 5(d) explains the skip of two instructions induced by EM pulse on buffer2. This is maybe because the difference in position makes it not as sensitive to the pulse as the buffer1. This also makes the distribution of the fault observed with a longer EM pulse on the two buffers different from each other.

B. Cache enabled

The faults induced by EMFI as the cache is enabled are classified into the following types:

- **replay** $i_1i_2i_3i_4$: (i_5, i_6, i_7, i_8) being replaced by (i_1, i_2, i_3, i_4)
- **skip** $i_5i_6i_7i_8$: (i_5, i_6, i_7, i_8) being replaced by (`nop`, `nop`, `nop`, `nop`)
- **other**: register fault, system fault, instruction replacement, up to three instructions skip, etc.

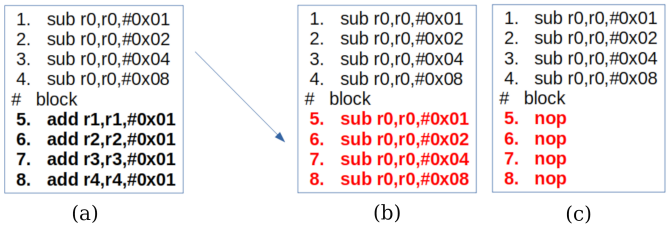

Figure 6. Demonstrations of replay and skip instructions with cache enabled, (a) test code, (b) replay of four instructions, (c) skip of four instructions

The test code used was the same as in section III-A. However it is worth mentioning that, as the cache is enabled, four instructions are buffered every four clock cycles, therefore the block is used to specify (i_5, i_6, i_7, i_8) as shown in Fig. 6(a). The two effects of replay of four instructions and skip of four instructions are depicted in Fig. 6(b) and Fig. 6(c), respectively. Here, for the replay fault, (i_5, i_6, i_7, i_8) are replaced by the (i_1, i_2, i_3, i_4). And for the skip fault, (i_5, i_6, i_7, i_8) are replaced by (`nop`, `nop`, `nop`, `nop`).

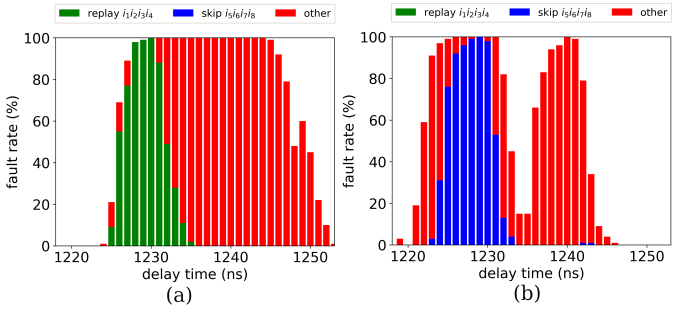

Figure 7. Cache enabled: impact of the PW on the fault model induced by EMFI on the block (a) 1.5 ns, (b) (f) 7.0 ns

The PW also has a decisive role on the obtained fault model. As the PW changes the fault rate for each fault model changes accordingly. It is noticed that the replay four instructions is observed with the PW of 1.5 ns as shown in Fig. 7(a), and the skip of four instructions is observed with the PW of 7.0 ns as shown in Fig. 7(b). For both of the fault models, the fault rate is up to 100%.

As the cache is enabled, the adversary can chose which fault model for the whole block of four instructions by using the corresponding PW. This makes the attack even more threatening and flexible.

(a)

Cycle	1	2	3	4	5	6	7	8
Buffer (64 bits)	(i_1, i_2, i_3, i_4)			(i_5, i_6, i_7, i_8)			(i_9, i_{10}, i_{11}, i_{12})	
Cache	(i_1, i_2, i_3, i_4)			(i_5, i_6, i_7, i_8)				

(b)

Cycle	1	2	3	4	5	6	7	8
Buffer (64 bits)	(i_1, i_2, i_3, i_4)			(i_1, i_2, i_3, i_4)			(i_9, i_{10}, i_{11}, i_{12})	
Cache	(i_1, i_2, i_3, i_4)			(i_1, i_2, i_3, i_4)				

(c)

Cycle	1	2	3	4	5	6	7	8
Buffer (64 bits)	(i_1, i_2, i_3, i_4)			(nop, nop, nop, nop)			(i_9, i_{10}, i_{11}, i_{12})	
Cache	(i_1, i_2, i_3, i_4)			(nop, nop, nop, nop)				

Figure 8. Hypothesis on EMFI-induced different fault models on SAMD21G18A with cache enabled (a) normal execution instruction buffering process, (b) EMFI-induced replay of four instructions, (c) EMFI-induced skip of four instructions

Further investigation shows that the fault repeats every four clock cycles. We also reasoned that the fault behavior observed here is a type of sampling fault, which happens as the EM pulse is injected during the sampling windows. As shown in Fig. 8(a), in normal executing process at the clock cycle 2 the content of buffer is expected to be updated with the block of (i_5, i_6, i_7, i_8). However, due to the impact of the EM pulse, the fault occurs here. We ascribed the replay effect to EM-induced prevention on the update of the buffer, leading to the cache being updated with previous block of data, which is similar with the result observed in [12]. As the result, (i_1, i_2, i_3, i_4) are re-executed or replayed instead of (i_5, i_6, i_7, i_8) as shown in Fig. 8(b). Notice that the replay only happens as EM pulse with short PW is used. As a longer PW is used, the impact is rather in the content of the buffer, meaning the update process still happens but instructions or data inside are corrupted by the EM pulse. As the modified instructions are not recognized by the PC, they are turned into instructions equivalent to `nop`, consequently the skip of four instructions is observed as shown in Fig. 8(c).

Obviously, the PW has a great impact on the fault models achieved on the target. In both cases with cache disabled and enabled, the same multiple fault models of replay and skip of block instructions with fault rate up to 100% were achieved with the PW of 1.5 ns and 7.0 ns, respectively. In the following sections, the PW of 1.5 ns was used for generating the replay fault, and the PW of 7.0 ns was used for generating the skip fault.

IV. Impact of the PA

A. Impact of the PA on replay fault

We also studied the impact of the PA on the fault rate of the replay fault model. Based on the result in section III, we fixed the PW at 1.5 ns to produce the replay fault in the MCU. The PA was increased from −6 dBm to 0 dBm with an increment

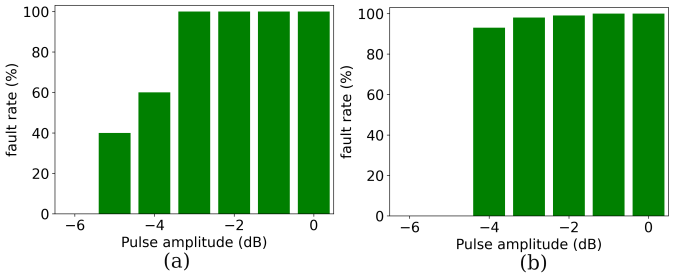

Figure 9. Influence of the PA on the fault rate of replay fault: (a) cache disabled: replay of two instructions, (b) cache enabled: replay of four instructions

of 1 dBm. The highest fault rates of replay of two instructions and four instructions that can be achieved for each PA value are shown in Fig. 9. As can be seen in Fig. 9(a), the replay of two instructions obtained with cache disabled can be observed with the PA starting from −5 dBm with fault rate up to 40%. The fault rate increases as the PA increases, and a fault rate of 100% can be reached starting from PA of −3 dBm. On the other hand, no replay fault is observed with PA of −6 dBm.

The replay of four instructions obtained with cache enabled is observed with PA starting from −4 dBm with the fault rate of 93%. The fault rate increases gradually to reach 100% starting from PA of −1 dBm as shown in Fig. 9(b). The fault is not observed with PA smaller than −4 dBm.

This experiment highlights the direct impact of the PA on the fault rate of the replay fault. To observe the fault, the PA needs to reach a certain value, and by increasing the PA higher fault rate up to 100% can be achieved.

B. Impact of the PA on skip fault

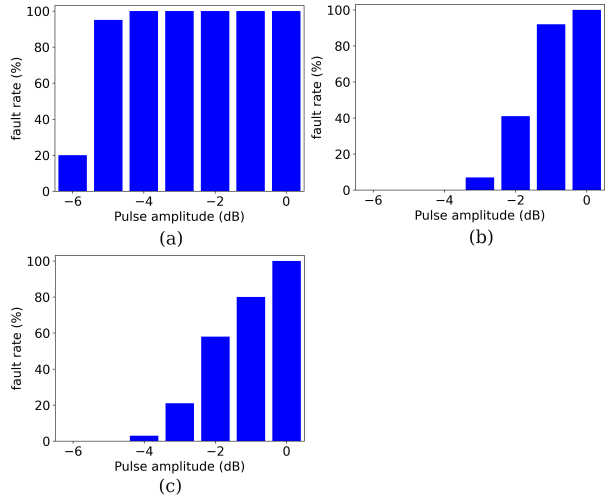

Figure 10. Influence of the PA on fault rate of skip fault, cache disabled (a) skip of two instructions (buffer1) (b) skip of two instructions (buffer2) (c) cache enabled: skip of four instructions

The impacts of the PA on the skip of two instructions and four instructions were also studied. Here, the PW was fixed at 7.0 ns, and we proceeded with the same PA variation from −6 dBm to 0 dBm with the increment of 1 dBm. The highest faults rate that can be obtained according to each tested PA are reported in Fig. 10. As the cache is disabled, the faults in the two buffers behave differently with increasing PA. As shown in Fig. 10(a) for buffer1, skip of two instructions is observed in with all the PA from −6 dBm to 0 dBm, with a fault rate of 100% that can be obtained from the PA of −4 dBm to 0 dBm. On the other hand for buffer2, the skip of two instructions can only be seen with the PA starting from −3 dBm; and as the PA increases, the fault rate increases accordingly. The fault rate can only reaches 100% with the highest tested PA value of 0 dBm.

It can be seen that buffer1 seems to be more sensitive to the EM pulse at current position of the probe than buffer2. However, it is obvious that with the PA of 0 dBm, it is possible to achieve skip of two instructions for both of the buffers.

The dependency of fault rate of skip of four instructions obtained when the cache is enabled on the PA is shown in Fig. 10(c). The skip can only be seen with the PA starting from −4 dBm with a linear increase to reach 100% of fault rate with the PA value of 0 dBm.

It is quite clear that for both cases of cache disabled and enabled, as the PW is fixed, the EMFI-induced fault models tend to be the same; and the PA mainly has the impact on the fault rate. As the PA increases, the fault rate increases accordingly.

V. Characterization of the Fault at Bit Level

As analyzed above, we assume that the skip of instructions is due to the EM-induced the corruption of one or multiple bits of the buffers content. An in-depth analysis of this corruption is proposed with the next experiments to identify the fault model at bit level (bit-set or bit-reset). For the following tests, we kept the same probe position and set the PW to 7.0 ns and the PA to 0 dBm to guarantee the highest fault rate on both 32-bit buffer (when the cache is disabled), and 64-bit buffer (when the cache is enabled). To detect bit-set fault, the buffer is filled with all bits at `0`. This is accomplished by using a test code consisting of successive same instructions `lsl r0,r0,#0x00` with the opcode of `0x0000`. With the same manner, bit-reset fault can be detected if fault occurs when the buffer is filled with all bits at `1`. Because there is no instruction with such opcode, the test code represents a successive same instructions `sub r7,r7,#0xff` of which the opcode is (`0x3fff`) (most of the bits are `1`). Fig. 11 shows the fault rate on the 32-bit and 64-bit buffers corresponding to cache disable and enable mode, respectively. It is obvious that very few faults occur as the buffers are filled with all `0`. On the other hand, many faults can be observed when most of the bits in the buffer are `1`. Indeed, the same behavior is reported on both the buffers when the cache is enabled or disabled. Through this test, we assume that at bit level, the fault induced by EMFI is bit-reset fault rather than bit-set. It should be noted that this behavior is reported only with a specific PW at the current probe position.

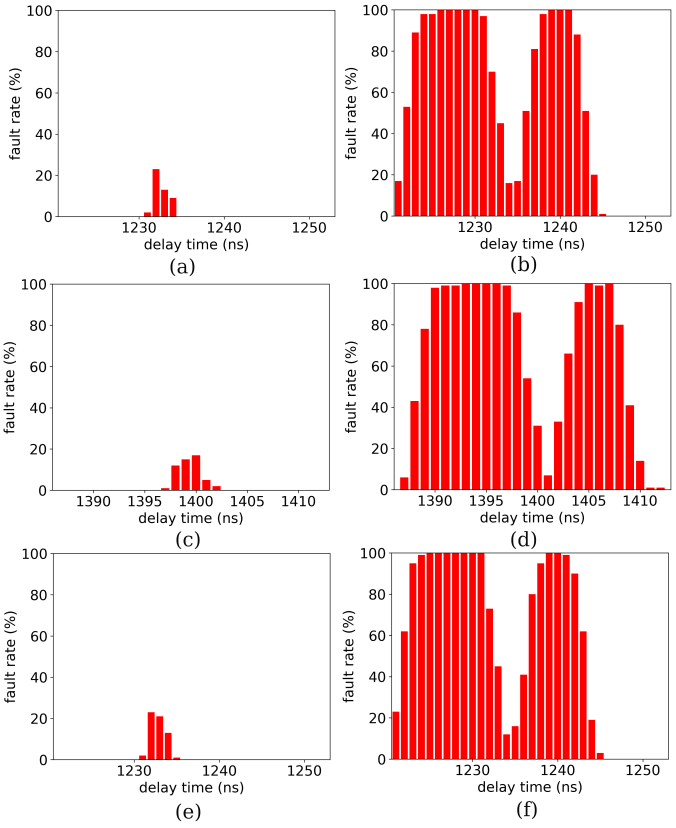

Figure 11. Fault rate is low for the bit-set test code when the cache is disabled: (a) 32-bit buffer1 and (c) 32-bit buffer2, and when the cache is enabled (e) 64-bit buffer. The fault rate is higher for the bit-reset test code, when the cache is disabled: (b) 32-bit buffer1 and (d) 32-bit buffer2, and when the cache is enabled (f) 64-bit buffer.

VI. Conclusion & Perspectives

In this paper, we reported on the observation of multiple fault models achieved with EMFI on a 32-bit MCU, including multiple instructions skip and multiple instructions replay by using the EM pulse with different PW. With a short pulse at 1.5 ns, replay of instructions was observed with fault rate up to 100%. While with a longer pulse, instructions modification was observed and by turning all the instructions into the ones equivalent to `nop`, skip of multiple instructions was observed. With the PW of 7.0 ns, the fault rate of multiple instructions skip reached up to 100%. The fault was ascribed to the impact of EM pulse on the Flash interface buffers. And different fault models were observed due to the fact that each PW has impact on different input of the buffer. As the cache is disabled, the buffer size is 32 bits; as the cache is enabled, the buffer size is 64 bits. The impact of the PA on the fault behavior was also systematically studied. The result showed that the PA has a direct impact on the fault rate, and should be chosen carefully to achieve a fault rate of 100%. By using the test code with a specific opcode containing maximum bit of `0` and `1`, the fault model at bit level was identified to be mostly bit-reset. Our result demonstrates both the precision and flexibility of EMFI for attacking MCU. This brings the adversary with more choices to implement password bypass, differential fault analysis to extract secret keys or deny of service when using EMFI to attack MCU. For the designer, this should be taken into consideration when implementing countermeasures against EMFI attack. Our future works will be centering on: (1) the verification of the fault models obtained here in other devices, (2) the study of mechanism under the faults, (3) the development of efficient countermeasures against the faults.

Acknowledgment

This work was partly funded by the SPARTA project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 830892.

References

[1] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 105–114. IEEE, 2011.

[2] Alessandro Barenghi, Guido Bertoni, Emanuele Parrinello, and Gerardo Pelosi. Low voltage fault attacks on the rsa cryptosystem. In *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 23–31. IEEE, 2009.

[3] Arthur Beckers, Josep Balasch, Benedikt Gierlichs, Ingrid Verbauwhede, Saki Osuka, Masahiro Kinugawa, Daisuke Fujimoto, and Yuichi Hayashi. Characterization of em faults on atmega328p. In *International Symposium on Electromagnetic Compatibility. IEEE*, 2019.

[4] Mathieu Dumont, Mathieu Lisart, and Philippe Maurine. Electromagnetic fault injection: how faults occur. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 9–16. IEEE, 2019.

[5] Mathieu Dumont, Philippe Maurine, and Mathieu Lisart. Modeling of electromagnetic fault injection. In *2019 12th International Workshop on the Electromagnetic Compatibility of Integrated Circuits (EMC Compo)*, pages 246–248. IEEE, 2019.

[6] Jean-Max Dutertre, Vincent Beroulle, Philippe Candelier, Stephan De Castro, Louis-Barthelemy Faber, Marie-Lise Flottes, Philippe Gendrier, David Hély, Regis Leveugle, Paolo Maistri, et al. 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 (FDTC)*, pages 1–6. IEEE, 2018.

[7] Jean-Max Dutertre, Timothé Riom, Olivier Potin, and Jean-Baptiste Rigaud. Experimental analysis of the laser-induced instruction skip fault model. In *Nordic Conference on Secure IT Systems*, pages 221–237. Springer, 2019.

[8] Microchip Technology Inc. SAM D21/DA1 Family. In *SAM D21/DA1 Family*.

[9] Haohao Liao and Catherine Gebotys. Methodology for em fault injection: Charge-based fault model. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 256–259. IEEE, 2019.

[10] Alexandre Menu, Shivam Bhasin, Jean-Max Dutertre, Jean-Baptiste Rigaud, and Jean-Luc Danger. Precise spatio-temporal electromagnetic fault injections on data transfers. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 1–8. IEEE, 2019.

[11] Nicolas Moro, Karine Heydemann, Emmanuelle Encrenaz, and Bruno Robisson. Formal verification of a software countermeasure against instruction skip attacks. *Journal of Cryptographic Engineering*, 4(3):145–156, 2014.

[12] Lionel Riviere, Zakaria Najm, Pablo Rauzy, Jean-Luc Danger, Julien Bringer, and Laurent Sauvage. High precision fault injections on the instruction cache of armv7-m architectures. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 62–67. IEEE, 2015.

[13] Sergei P Skorobogatov and Ross J Anderson. Optical fault induction attacks. In *International workshop on cryptographic hardware and embedded systems*, pages 2–12. Springer, 2002.