Permanent Laser Fault Injection into the Flash Memory of a Microcontroller

Raphael Viera, Jean-Max Dutertre, Mathieu Dumont, Pierre-Alain Moëllic

https://doi.org/10.1109/NEWCAS50681.2021.9462773
19th IEEE International New Circuits and Systems Conference (NEWCAS)

Permanent Laser Fault Injection into the Flash Memory of a Microcontroller

Raphael Viera*, Jean-Max Dutertre*, Mathieu Dumont†‡ and Pierre-Alain Moëllic†‡

*Mines Saint-Etienne, CEA Tech, Centre CMP, F - 13541 Gardanne, France
Email: {raphael.viera, dutertre}@emse.fr
†CEA Tech, Centre CMP, Equipe Commune CEA Tech - Mines Saint-Etienne, F-13541 Gardanne, France
‡Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France
Email: {mathieu.dumont, pierre-alain.moellic,}@cea.fr

***Abstract*—The Flash memory of a Microcontroller Unit (MCU) is an important part of its attack surface as it contains its firmware and its security related data (e.g. passwords and cryptographic keys). Recent research works report the use of Laser Fault Injections (LFI) to corrupt the firmware at run time by targeting the Flash memory during its read operations (data reads from Flash were also faulted). These faults, induced on a single bit and following a bit-set fault model, were non-permanent: the data stored in Flash stayed unaltered while only their read copies were corrupted. We report an extension of this fault model on the Flash memory of a 32-bit MCU. Using LFI, we were able to induce permanent faults into its Flash. Single bit faults, that followed a bit-reset fault model, were induced during the Flash write operations. As a proof of concept, we describe how we were able to iteratively set to zero all the bits of a 32-bit password using a laser pulse with relatively undemanding settings (15 μm beam diameter, and 3 μs pulse duration).**

I. Introduction

In the field of secure circuits characterization, physical attacks constitute a major threat for security engineers and semiconductors manufacturers. When an attacker has a physical access to a MCU, he can exploit hardware-based vulnerabilities in order to retrieve secret information.

Sensitive data, such as encryption keys or passwords, are usually stored in a non-volatile memory (NVM): the Flash memory, which is frequently embedded on MCUs. Fault injection analysis (FIA) have become a common way to defeat the security mechanisms of these devices [1].

Previous FIA related works demonstrate attacks in a Flash memory by means of LFI (Laser Fault Injection). The attacks reported in [2], [3] aimed at preventing changes of the memory content by disrupting its normal operations. Recent studies [4]–[6] demonstrated on 32-bit MCUs embedding a NOR Flash that LFI induces non permanent faults when instructions and data are read from the Flash. The stored data were left unmodified while the read data were faulted one bit at a time following a *bit-set* fault model (i.e. a 0 is turned into a 1). Changing the location of the laser beam made it possible to choose what bit of a 32-bit word was faulted with a 100% success rate.

This paper reports an improvement of the previous threats as we were able to induce permanent faults into a MCU's Flash memory using LFI during write operations. For the same locations used to induce faults in a Flash during read operations, we induced single bit *bit-reset* faults. Then to validate this fault model, we set to zero a password stored in the target's Flash memory.

This article is organized as follows. Section II provides the background information on NOR Flash memories arschitecture and LFI. Section III describes our laser injection setup and the used MCU target. The obtained experimental results are given in section IV. Finally a conclusion is drawn in section V.

II. Background

This section describes the basic architecture and operating modes of the MCU's embedded Flash memory: a NOR Flash for the circuit we considered. We also describe the effect of LFI on a Flash memory.

A. NOR Flash Architecture

NOR Flash memory was born in mid 80's as an EPROM replacement [7]. The advanced architecture of NOR Flash memory is effectively conceived to meet the requirements of MCU by optimizing the trade-off between speed and power consumption. It also gives the possibility of using a single chip to store both user code and user data.

All the features of the NOR Flash are inherently related to the memory cell concept and its memory array organization as depicted in Fig. 1. Each transistor is called a cell and it is made of a stacked-double-poly floating-gate MOS device. The memory cells are arranged in a NOR type array organization, which means that all the cells are parallel connected with a common ground node and the bit lines are directly connected to the drains of memory cells.

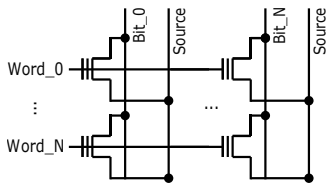

Figure 1: NOR Flash memory architecture.

B. NOR Flash Operating Modes

A NOR Flash memory allows three elementary operations: Program (Write), Erase and Read as illustrated in Fig. 2. The high voltages needed for Program/Erase operations are internally generated in the MCU chip (the program and erase voltages are given as an illustration). The write and erase operations to the Flash memory are managed by an embedded Flash Program/Erase Controller (FPEC).

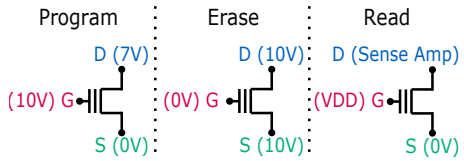

Figure 2: Operating modes of the Flash memory: Program (Write), Erase, and Read.

Read: to read a value from the Flash memory at the application level, it is sufficient to create a pointer to the specific address and read its content as a common memory space. At the electrical level, the threshold voltage of a cell is measured by a sense amplifier which in turn determines if its content is equivalent to a logic 0 (programmed) or logic 1 (erased) [7].

Program (Write): the embedded Flash memory can be programmed at run time using in-application programming (download programming data into memory while the application is running). In the case of our target (see III-B), only 16 bits (from a 32-bit word) can be programmed at a time. At the electrical level, each cell can be programmed by using channel-hot-electron injection [7]. As a result, electrons are pushed into the floating gate of a cell. At logical level, a programmed cell stores a logical 0 (it is not possible to write a 1, which is done through erasing).

Erase: at the physical level, the erase operation is done using the Fowler-Nordheim tunneling effect [7]. As a result, the electrons stored into a cell's floating gate are removed. At logical level, an erased cell stores a logical 1. In the case of our target (see III-B), erasing is done on a whole memory page: 1 kB of data.

Hence, writing data (made of both 0s and 1s) in a Flash Memory is made in several steps: 1st a given section of the memory is erased, 2nd 0s are written in the relevant places.

C. Flash Memory Laser Fault Model

When a laser beam passes through the silicon substrate of a target, it creates electron-hole pairs along the path of the laser. If these induced charge carriers reach the strong electric field found in the vicinity of reverse biased PN junctions, the electrical field puts these charges into motion and a transient current flows. [4] and [5] describe how such a transient current may discharge the bitline of a Flash memory cell during a read operation resulting in a *bit-set* transient fault. As LFI is local, it makes it possible to fault a single bit and to choose its index while reading a 32-bit word with 100% success rate.

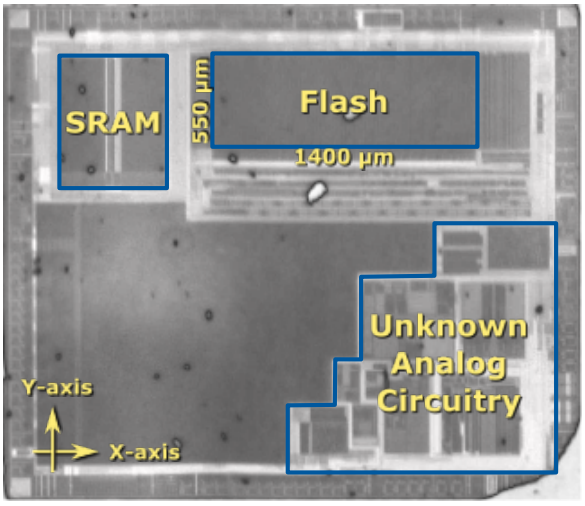

Figure 3: Infrared picture of the targeted microcontroller.

III. Experimental Method

A. Laser setup for the experimental fault injection

All experimental results reported in the next sections were obtained through the target backside using a laser source with a 1,064 nm wavelength (near infrared range). This source was used to generate laser pulses of a duration equal to 3 μs and a power of 200 mW. By using an objective with x5 magnification, the obtained laser spot diameter is 15 μm.

B. Target board and microcontroller

The target MCU we used for our experiments embeds an ARM Cortex-M3 core with 128 kB of Flash memory. It is manufactured in the CMOS 90 nm technology node. Its clock was set to 7.4 MHz. An infrared picture of the target microcontroller is shown in Fig. 3. The chip has a dimension of 3 mm x 2.5 mm. The main parts of the chip are outlined including the Flash memory which constitutes the target of this experiment. Since LFI requires the surface of the chip's die being visible, the microcontroller packaging was milled away with engraving tools. The chip was then placed into a test board compatible with the ChipWhisperer platform [8]. The board was mounted on a motorized XYZ-stage and a CCD camera mounted on optical objectives (Fig. 4).

Target's Flash memory organization: it has a main memory block containing 128 pages of 1 kB and an information block as shown in Table I (the corresponding base addresses in the memory map are given). The memory is organized as 32-bit wide memory words that can be used for storing both code and data.

C. Embedded Software and Aim of Experiment

Our work aimed at demonstrating the ability of LFI to inject permanent faults at write time into a Flash memory. To that end, we wrote a dedicated test code described by the pseudo-code in listing 1. The same code was used by the experiments reported in sections IV-A and IV-B.

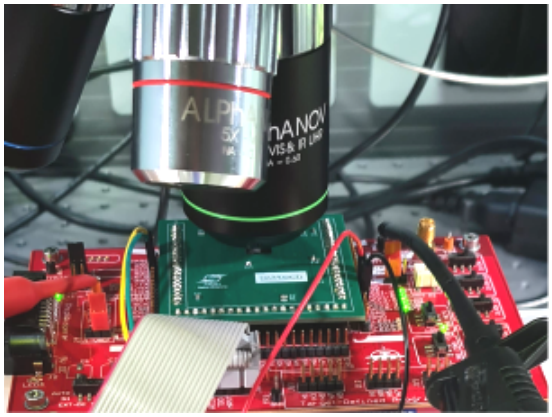

Figure 4: Objective lenses above the targeted 32-bit MCU mounted on the *Chipwhisperer* platform.

Table I: MCU's embedded Flash module organization.

Block	Name	Base Address	Size
Main mem.	Page 0	0x0800 0000 - 0x0800 03FF	1 kB
	...	...	...
	Page 127	0x0801 FC00 - 0x0801 FFFF	1 kB
Information	System mem.	0x1FFF F000 - 0x1FFF F7FF	2 kB
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16

Listing 1: Program used to perform the experiments reported in sections IV-A and IV-B

```
1  int main(void)
2    while(1)
3      UART_wait_for_cmd(); // calls program_memory()
4
5  void program_memory(is_init, val)
6    if (is_init) // Write data into memory
7      is_init = false;
8      u32 val = val;
9    else // Read existing data from memory
10     val = *( volatile u32*)0X0801FFF0;
11
12   FPEC_FlashPageErase(127); // Erase page 127 (1 KB)
13   FPEC_FlashWrite(addr, val, is_init); // Program addr
14   val = *( volatile u32*)0X0801FFF0; // Read after LFI
15   UART_send_value(val) // Send val through UART
16
17 void FPEC_Write(addr, data, is_init)
18   ...
19   if (is_init) // Write into memory
20     *( volatile u16*)0X0801FFF0 = data[i];
21   else // Set trigger right before writing into memory
22     trigger_high();
23     __asm("NOP");
24     *( volatile u16*)0X0801FFF0 = data[i];
25     __asm("NOP");
26     trigger_low();
27   ...
```

The code begins with the `main()` function, it waits for an UART message that will call the `program_memory()` function. Using FPEC rules, this function writes into the Flash memory, at address `0x0801FFF0`, the value `0xFFFFFFFF` for section IV-A or the original password (`0xABCD2021`) for section IV-B. If `is_init=false`, the program will read the content of the memory address and rewrite the same content again (used by the experiment of section IV-B), otherwise it will only write the original value into memory. Each writing operation calls the `FPEC_write()` function which sets a synchronization trigger for the laser source. The laser is then shot after a programmed delay. As reported in the next section IV, at each iteration of the writing operation, a specific bit in memory can be reset from logic '1' to logic '0'.

IV. Results

A. LFI into a Flash memory during write operation

Before running the experiment to overwrite the original and unknown password stored in the Flash memory, we studied which parts of the Flash memory are LFI sensitive. For this experiment, the value `0xFFFFFFFF` was written at the memory address `0x0801FFF0`.

As highlighted in listing 1, each write operation provides a trigger signal (it is drawn in black in Fig. 5) that outlines the duration of this operation. An electronic synchronization board is then used to deliver a shot signal to the laser source, denoted Pulse Picker in Fig. 5 (blue) after a programmable delay. An image of the actual laser pulse is also depicted in red; its pulse width was set to 3 μs. In order to write 32 bits into the memory, FPEC determines that two 16-bit write operations are necessary (as shown in Fig. 5). A write operation lasts 56 μs, setting the delay (measured from the rising edge of the trigger signal to the shot signal) at 44 μs, we were able to induce permanent *bit-reset* faults into one bit at the written address of the Flash memory.

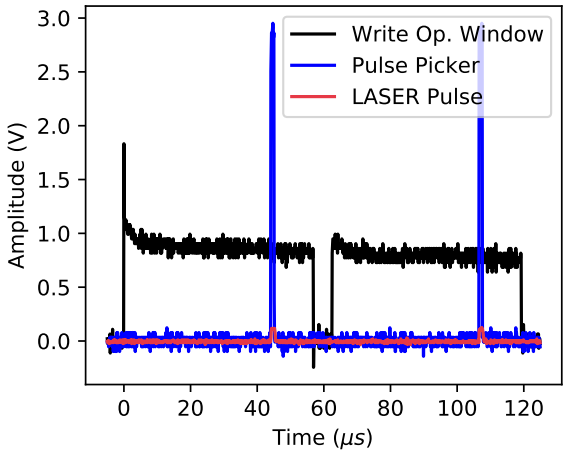

Figure 5: Voltage waveforms sampled during Flash writing operations: a trigger signal (in *black*) outlined every write operation (16-bit). It is used to synchronize the laser shots with the write operations (the laser command is drawn in *blue* and the laser shot itself in *red*).

Fig. 6 reports the LFI sensitivity scan we carried out on the Flash array surface (targeting its memory cells) with ΔX=5 μm

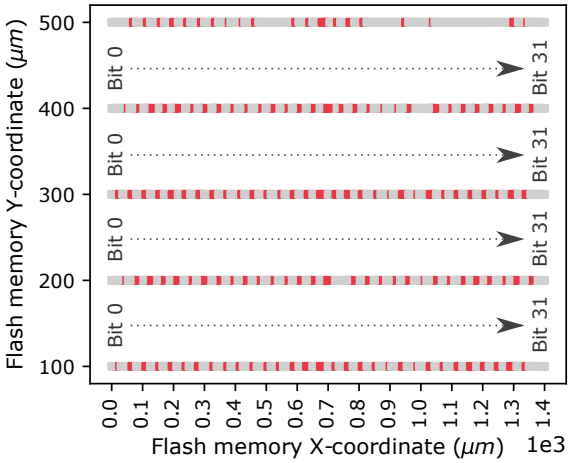

Figure 6: LFI scanning of the MCU Flash memory: depending on the laser spot X position (from left to right) bit 0 to 31 of the written word were faulted according a *bit-reset* fault model.

and ΔY=100 μm displacement steps. Moving the laser spot along the X-axis made it possible to fault independently every bit of a 32-bit word; locations where faults were induced are marked in red: for Y=100 μm the 32 bit locations are clearly visible. While changing the laser spot Y coordinates had no effect on the faulted-bit. These patterns of LFI sensitivity are the same that were observed by [4]–[6] when inducing faults during a read operation; except that they induced *bit-set* faults.

This experiment assessed that single *bit-reset* faults can be induced by laser during a write operation in a predictable and repeatable manner using a large 15 μm laser spot diameter.

B. Password zeroing

As a proof of concept, we then applied this new fault model to iteratively set to zero a 32-bit password stored in Flash. We used the code in Listing 1 with the Y-axis position set to 100 μm, while the X-axis was swept from 0 μm to 1400 μm with a ΔX=5 μm step. For this experiment we assumed that the password was unknown (though its value was `0xABCD2021`), and that the only available information was its address in memory (`0x0801FFF0`). The experiment aim was to reset the password to `0x00000000` (a known value). In a conservative approach, we used 280 steps ($\frac{1400\,\mu m}{5\,\mu m}$) along the X axis to iterate on the 32 bits of the password. As a result, all the 1s in the password were reset to zero (the bits already at 0 were not modified). This process is illustrated in Table II which depicts actual data from the experiments and highlights (in red) the induced *bit-resets*. Therefore the unknown password was replaced by a known all-0s one that can be used to give access to secure information saved in the Flash memory. By performing this experiment several times, we could observe the same result each time, thus giving a 100% success rate.

V. Conclusions and Discussion

Our experiments assess that LFI can be used during the write operations of a Flash memory: single *bit-reset* faults

Table II: Iterative zeroing of the password (faults in red).

Bit #	Value (Hex)	Value (Bin)
-	`0xABCD2021`	1010 1011 1100 1101 0010 0000 0010 0001
0	`0xABCD2020`	1010 1011 1100 1101 0010 0000 0010 0000
5	`0xABCD2000`	1010 1011 1100 1101 0010 0000 0000 0000
14	`0xABCD0000`	1010 1011 1100 1101 0000 0000 0000 0000
16	`0xABCC0000`	1010 1011 1100 1100 0000 0000 0000 0000
18	`0xABC80000`	1010 1011 1100 1000 0000 0000 0000 0000
19	`0xABC00000`	1010 1011 1100 0000 0000 0000 0000 0000
22	`0xAB800000`	1010 1011 1000 0000 0000 0000 0000 0000
23	`0xAB000000`	1010 1011 0000 0000 0000 0000 0000 0000
24	`0xAA000000`	1010 1010 0000 0000 0000 0000 0000 0000
25	`0xA8000000`	1010 1000 0000 0000 0000 0000 0000 0000
27	`0xA0000000`	1010 0000 0000 0000 0000 0000 0000 0000
29	`0x80000000`	1000 0000 0000 0000 0000 0000 0000 0000
31	`0x00000000`	0000 0000 0000 0000 0000 0000 0000 0000

are induced repeatably when targeting the Flash array. This is an extension of the previous state-of-the-art [4]–[6] that reported laser-induced single *bit-set*s during read operations. As an illustration we reset a 32-bit password stored in the Flash memory. These results were obtained using a laser beam diameter of 15 μm and a pulse duration of 3 μs, which are relatively undemanding settings (i.e. that an attacker can access more easily).

This work is a first proof of feasibility, its relevance may be increased by devising a more practical attack scenario. Though we used a trigger signal to synchronize the laser shots with the target activity, it can be replaced by observing the electrical activity of the target as a write operation involves the activation of the MCU charge pump to produce the required high voltage.

At this point, the underlying physical phenomenon explaining the *bit-reset* fault model is still to be identified (that of the LFI-induced *bit-set* during read operations is given in [4]). Testing the use of multi-spots LFI is another prospective work that may increase further the associated threat.

References

[1] A. Barenghi, L. Breveglieri, I. Koren *et al.*, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, Nov 2012.

[2] S. Skorobogatov, "Optical fault masking attacks," in *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2010, pp. 23–29.

[3] F. Cai, G. Bai, H. Liu *et al.*, "Optical fault injection attacks for flash memory of smartcards," in *2016 6th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2016, pp. 46–50.

[4] B. Colombier, A. Menu, J. M. Dutertre *et al.*, "Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller," *IEEE HOST*, 2019.

[5] A. Menu, J. M. Dutertre, J. B. Rigaud *et al.*, "Single-bit Laser Fault Model in NOR Flash Memories: Analysis and Exploitation," *FDTC*, 2020.

[6] K. Garb and J. Obermaier, "Temporary laser fault injection into flash memory: Calibration, enhanced attacks, and countermeasures," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2020, pp. 1–7.

[7] G. Campardo, R. Micheloni, and D. Novosel, *VLSI-Design of Non-Volatile Memories*. Springer Berlin Heidelberg, 2005.

[8] C. O'flynn and Z. Chen, "ChipWhisperer: An open-source platform for hardware embedded security research," in *Lecture Notes in Computer Science*, vol. 8622 LNCS. Springer Verlag, 2014, pp. 243–260.