Tampering with the flash memory of microcontrollers: permanent fault injection via laser illumination during read operations

Jean-Max Dutertre, Rodrigo Silva Lima, Matthieu Pommies, Anthony Bertrand, Raphael A. Camponogara Viera

Author's version
JCEN 2023
https://doi.org/10.1007/s13389-023-00335-z

Tampering with the Flash Memory of Microcontrollers: Permanent Fault Injection via Laser Illumination During Read Operations

Raphael Viera[1*], Jean-Max Dutertre[1], Rodrigo Silva Lima[1,2], Matthieu Pommies[2], Anthony Bertrand[2]

[1]Mines Saint-Etienne, CEA, Leti, Centre CMP, F - 13541, Gardanne, France.
[2]Centre Technologique ALPhANOV, Talence, France.

*Corresponding author(s). E-mail(s): raphael.viera@emse.fr;
Contributing authors: dutertre@emse.com; rodrigo.lima@emse.fr;
matthieu.pommies@alphanov.com; anthony.bertrand@alphanov.com;

Abstract

Modern Microcontroller Units (MCUs) often feature integrated Flash memory, which has been found to be vulnerable to hardware attacks. This type of memory is used to store critical data, including firmware, passwords, and cryptographic keys, making it a valuable target for attackers. Recent research has demonstrated the use of Laser Fault Injection (LFI) during runtime to corrupt firmware by targeting the Flash memory during read operations. However, these faults are non-permanent, as they only affect the read copies of the data without altering the actual data stored in the Flash memory, following a bit-set fault model induced on a single bit. In our work, we extend this fault model to the Flash memory of a 32-bit MCU, allowing us to induce permanent faults by compromising the stored data during read operations. In addition, we leverage Photoemission Analysis (PEA) for target identification and characterization, enhancing the precision of our attack. By utilizing a double-spot LFI technique, we are able to concurrently induce permanent bit-set faults at two distinct locations in the Flash memory, increasing the complexity and effectiveness of the attack. We also provide a practical example of how this fault model can be applied, wherein we iteratively change all 32 bits of a password to logic '**1**', successfully bypassing a basic counter for login attempts. It is important to note, however, that there are physical limitations associated with using multi-laser spots in this context, which we thoroughly discuss in our research. Nonetheless, our approach presents a powerful method for exploiting vulnerabilities in Flash memory of MCUs, underscoring the need for robust security measures to protect critical data and mitigate the risks associated with hardware attacks.

Keywords: Hardware Security, Fault Injection Attacks, Laser Fault Injection, Microcontroller, Flash memory

1 Introduction

Physical attacks have been shown to be a major threat for the security of Microcontroller Units (MCUs). They can be utilized to find and exploit several hardware-related vulnerabilities with the ultimate goal of retrieving secret information.

MCUs often integrate cryptographic functions for processing, transmitting, and storing sensitive data, utilizing secret parameters like encryption

keys, seeds, or passwords. These parameters are usually stored in a non-volatile memory (NVM) such as Flash memory, which is commonly embedded in MCUs. However, semi-invasive attacks such as Photoemission Analysis (PEA) and Fault Injection Analysis (FIA) have become prevalent methods to bypass the security mechanisms of these devices. [1–5].

PEA attacks have been utilized to reverse-engineer circuits, leveraging the emission of photons to map memories' addresses to their physical locations on devices. For instance, in [6], the authors demonstrate how PEA can be used to attack a target MCU. Furthermore, recent advancements have led to techniques that exploit light emissions to extract secret information from these devices, as discussed in [7].

Previous FIA related works demonstrate attacks on a Flash memory by means of LFI. The attacks reported in [8, 9] aimed at corrupting the memory content by disrupting its normal operations. Recent studies [10–12] 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'). A *bit-reset* (i.e., a '1' is turned into a '0') fault model during Flash programming operations was presented in [13]. This model induces permanent faults directly on the data stored in the Flash, providing new attack scenarios. Those studies also show how changing the location of the laser beam made it possible to choose which of the 32 bits was faulted.

As in [14], we induced multi-bit transient faults simultaneously on data read from a MCU's Flash memory. This paper reports an extension of previous threats. We were able to inject 2-bit permanent faults during read operations by exploiting a vulnerability in the programming flow necessary for updating a variable stored in the Flash. We showcase the feasibility of permanently rewriting sensitive data stored in the same page as non-sensitive, user-modifiable data. We also present physical related limitations of using multiple laser spots.

Taking advantage of how data are written in this particular MCU's Flash memory, we injected permanent faults into data that were neither explicitly written nor read, though the faults were induced during hidden read operations. By using a double-spot laser assembly, we induced *bit-set* faults on bits located on different memory addresses simultaneously. The faulted values were written back to the Flash together with genuine data. To validate this new fault model, we set all 32 bits of an unknown password stored on the Flash memory to logic '1'. To be able to modify the password without exhausting the number of allowed attempts, the second laser beam was used to hold the password try counter to its initial value (3). This allowed us to bypass a common security countermeasure against brute force attacks and to ultimately have access to elevated user privileges.

Photoemission Analysis was employed to characterize the target's Flash memory, using it to locate the logical addresses and to determine the position of bits more accurately, saving time during the characterization phase. The employed attack techniques necessitate some preparation of the target device, including the depackaging of the MCU's backside to visualize the circuit's light emissions and accurately target the laser beam. Further details regarding the methodology employed in these attacks will be elaborated upon.

This article is organized as follows: Section 2 provides background information on NOR Flash memories architecture and operating modes as well as a background on LFI and PEA. Section 3 describes our laser injection setup and the MCU target. Section 4 presents the process of characterizing the target. The obtained experimental results are given in section 5. Finally, a conclusion is drawn in section 6.

2 Background and Vulnerability Assessment

In this section, we discuss the fundamental architecture and operational modes of the embedded Flash memory in the MCU, specifically focusing on NOR Flash for the circuit under consideration. We also examine the impacts of LFI on NOR Flash memories and outline the principles of Photoemission Analysis.

2.1 NOR Flash Architecture

The NOR Flash memory was born in the mid 80's as an EPROM replacement [15]. The advanced

architecture of this non-volatile memory is effectively conceived to meet the requirements of the MCU by optimizing the trade-off between speed and power consumption. It also gives the possibility of using a single Intellectual Property (IP) design 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 (`Bit_0` to `N` in Fig. 1) are directly connected to the drains of memory cells.

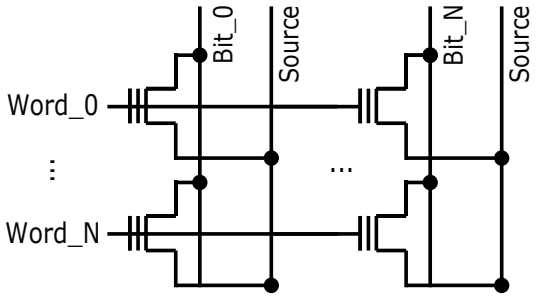


Fig. 1: NOR Flash memory architecture

2.2 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 (voltage values are given as an illustration). The write and erase operations to the Flash memory are managed by an embedded Flash Program/Erase Controller (FPEC).

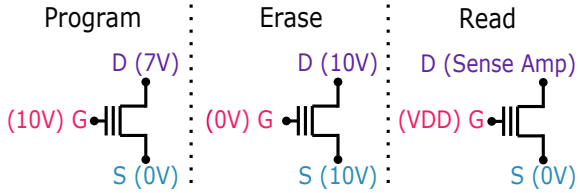


Fig. 2: Operating modes of the Flash memory: Program (Write), Erase, and Read

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 (connected to the cell's bit line) and compared with a threshold voltage reference (V_TREF in Fig. 3). If the observed threshold voltage is lower than V_TREF, it is determined that the memory cell contains a logic '1' (erased). If the observed threshold voltage is higher than V_TREF, the cell's content is equivalent to a logic '0' (programmed) [15].

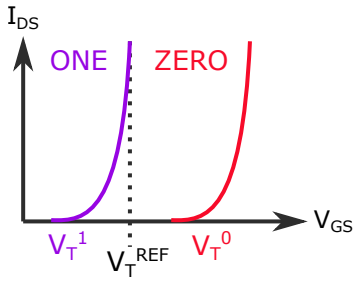


Fig. 3: Read operation mode at the electrical level. Observed waveforms by the sense amplifiers. Value is read as one or zero depending on the $I_{DS}-V_{GS}$ characteristics of the Flash memory cell

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 (cf. section 3.3), 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 [15]. As a result, electrons are pushed into the floating gate of a cell as illustrated in Fig. 4. 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 [15]. As a result, the electrons stored into a cell's floating gate are removed as illustrated in Fig. 5. At the logical level, an erased cell stores a logical '1'. In the case of our target (see 3.3), erasing is done on a whole memory page containing 1 kB of data.

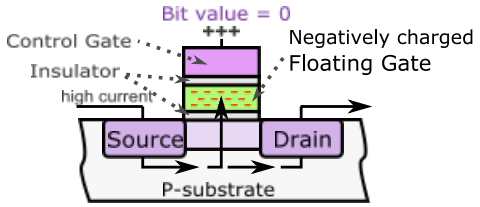


Fig. 4: Cross-section view of a Flash Memory cell. Program operation mode at the physical level

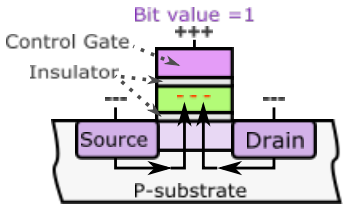


Fig. 5: Cross-section view of a Flash Memory cell. Erase operation mode at the physical level

As a result, writing data into a Flash memory involves a two-step process. Firstly, the targeted area, which encompasses the entire memory page, is erased by setting all of its bits to '1'. Secondly, the bits that need to be set to '0' are programmed. This process is influenced by the organization of MCU's Flash memories, which use pages to define the minimum granularity of erase operations. Consequently, when writing a 32-bit word into a Flash memory, the following steps are necessary: (1) reading the complete page data into the SRAM to prevent data loss during the erase step, (2) erasing the entire page, and (3) programming the page using the read data and the 32-bit word intended for writing. The vulnerability discussed in Section 5 takes advantage of this complex mechanism.

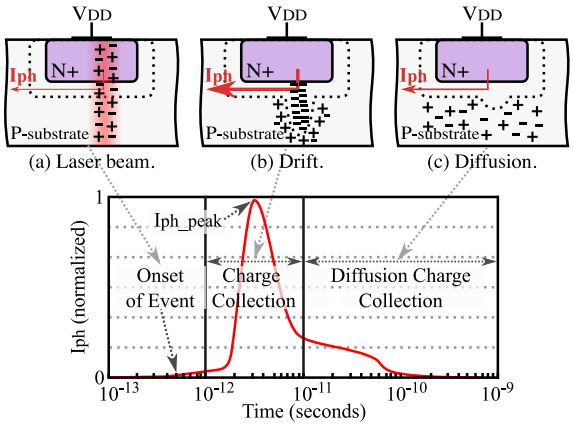
2.3 Effect of a Laser Shot at Transistor Level

ICs are known to be sensitive to induced transient currents. Such currents may be caused by a laser beam passing through the device, creating electron-hole pairs along its path [16]. These induced charge carriers generally recombine without any significant effect, unless they reach the strong electric field found in the vicinity of reverse-biased PN junctions (the reverse-biased junction is the most laser-sensitive part of circuits) [17]. In this case, the electrical field puts these charges into motion and a transient current flows. Each induced transient current has its particular characteristics such as polarity, amplitude and duration that depend on laser energy, laser shot location, device technology, device supply voltage and output load. The nature of these currents was first studied in the case of radioactive particles [18–22]. Laser illumination was first used as a way to emulate the effect of ionizing particles since the properties of the transient currents they both induce are similar.

Fig. 6 translates to the case of laser illumination the results of [17]. As shown in Fig. 6a, at the onset of an event caused by a laser shot, a track of electron hole pairs with high carrier concentration is formed along the path of the laser beam. When the resultant track traverses or comes close to the depletion region, carriers are rapidly collected by the electric field creating a current/voltage transient at that node. An interesting feature of the event is the distortion of the potential into a funnel shape [20, 23]. This funnel enhances the efficiency of the drift collection by extending the field depletion region deeper into the substrate (Fig. 6b). The profile of the funnel (size and distortion) depends on the substrate doping. This collection phase is completed in the picosecond range and is followed by a phase where diffusion begins to dominate the collection process (Fig. 6c). An additional charge is collected as electrons diffuse into the depletion region on a longer time scale (nanosecond range) until all excess carriers have been collected, recombined, or diffused away from the junction area. A laser-induced transient current is thus called 'photocurrent' [24, 25]. The corresponding current pulse $I_{Photocurrent}$ (I_{Ph}) resulting from these three phases is shown in Fig. 6d. The red arrows in Fig. 6 represent the transient current flowing from the sensitive drain to the $P_{substrate}$ biasing contact tied at GND.

2.4 Flash Memory Laser Fault Model

To the best of our knowledge there is no study at physical level reporting on how a laser injection affects the cells of a Flash memory. However, since the Flash memory cell is similar to an NMOS transistor with a floating gate, we suppose that the laser is able to induce a transient current in the Flash memory cell by irradiating its inverse



(d) Induced transient current.

Fig. 6: Charge generation and collection phases in a reverse-biased PN junction and the resultant transient current caused by the passage of a laser beam [16, 17]

polarised PN junction as it happens for the NMOS transistor.

In this case, laser shots may generate photocurrents in the Flash cell when there is a reverse biased PN junction between the drain and the $P_{substrate}$. as shown in Fig 7. Thus, an induced transient current (I_{Ph}) flows from the drain to the $P_{substrate}$ biasing contact (at GND).

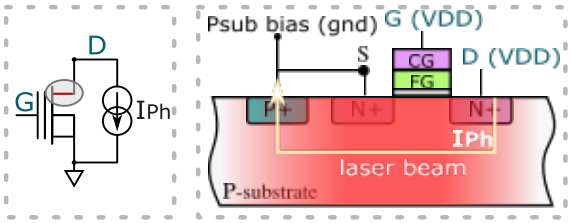


Fig. 7: Simplified representation of the laser-induced current component (I_{Ph}) in a Flash memory cell. On the left, its schematic view with an attached current source to simulate the laser-induced photo-current I_{Ph}. On the right, its cross-section view with the laser-induced current I_{Ph}

The authors of [10] and [11] 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 (the laser effect is then restricted to a single bit-line) and to choose its index while reading a 32-bit word with 100% success rate.

2.5 Photoemission Analysis

The emission of photons by electronic circuits has long been utilized by the Failure Analysis community to locate and diagnose defects within chips [1]. In CMOS devices, when a transistor changes states, photons are emitted if certain conditions are met. At the physical level, charge carriers are accelerated by the strong electric field present in the source-drain region and their kinetic energy is converted into luminous energy when leaving this zone [26], a phenomenon known as Hot-Carrier Luminescence.

This phenomenon is more noticeable in N-type transistors than in P-type transistors due to the greater mobility of electrons compared to holes [27]. The emitted photons have optical wavelengths ranging from $500\,nm$ to over $1200\,nm$, known as the Short Wave Infrared (SWIR) range [28, 29].

However, a single switching event has a very low probability of emitting photons, requiring a large number of switching operations to generate a detectable amount of light [1]. The operating frequency and supply voltage of a circuit also influence the number of emitted photons in a given interval: increasing the frequency equals to a higher amount of switching operations per second while a higher supply voltage improves the chances of emitting a photon.

This technique is best performed through the backside of a target, as the substrate is transparent to the SWIR wavelength and thus the emitted photons can traverse it. On modern circuits, the amount and complexity of metal connection layers on the front side block the majority of optical emissions. Furthermore, most defensive measures (e.g. shields) are present on the frontside only [30, 31].

By utilizing a specialized SWIR camera designed for this precise purpose and focusing it at a circuit's backside, it is possible to capture photons emitted by its operation. This capability proves particularly advantageous in several ways, including identifying active regions within the chip, pinpointing logic blocks, highlighting areas of interest, and efficiently mapping Flash memory addresses. As a result, employing this

technique significantly streamlines the target characterization phase, leading to substantial time savings.

3 Experimental Method

3.1 Laser setup for the experimental fault injection

The experimental results for LFI, as detailed in the following sections, were obtained by targeting the backside of the target using a laser source with a wavelength of 1,064 nm (in the near-infrared range). Laser pulses with durations ranging from 30 ns to 90 ns and a power of 200 mW were generated using this source. An objective with 5x magnification was used, resulting in a laser spot diameter of 15 μm.

3.2 Camera setup for Photoemission Analysis

The PEA results were acquired using the highly-sensitive SWIR camera Ninox 640 II, coupled with x5 and x20 magnification lenses. This camera can be used to capture IR and PE images by changing its gain mode and exposure time. The x5 lens provides a general view of the entire Flash memory, while the x20 lens offers a more detailed view of the bitlines' disposition.

The camera's exposure time was set to 2,500 ms. Initially, we captured a PE image of the circuit powered on but idle, referred to as the "background-noise image". This image was later subtracted from captures of the circuit while executing a scenario to improve the Signal-to-Noise Ratio. Subsequently, we ran a simple write-erase routine loop and captured the photons emitted by the Flash memory during this process.

3.3 Target board and microcontroller

The target MCU used for the experiments embeds an ARM Cortex-M3 core with 128 kB of Flash memory manufactured in the CMOS 90 nm technology node. Its clock was set to the maximum MCU frequency of 24 MHz. An infrared picture of the target is shown in Fig. 8. The chip has an approximate dimension of 3 mm x 2.5 mm. The main parts of the chip are outlined including the

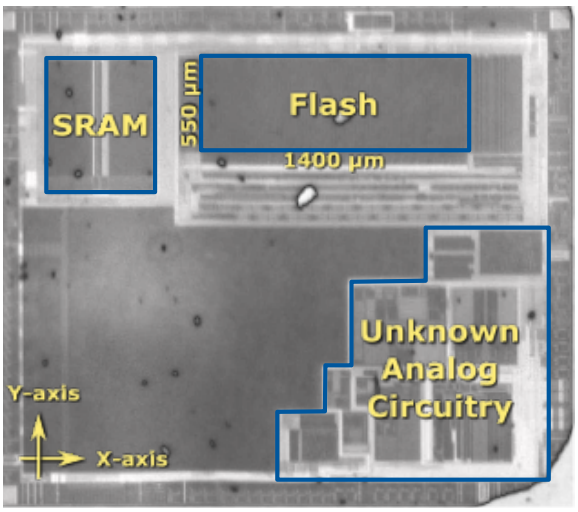


Fig. 8: Infrared picture of the targeted MCU

Flash memory which constitutes the target of this experiment. Since LFI requires the surface of the chip's die being visible, the MCU packaging was milled away with engraving tools [32]. The chip was then placed into a custom-made test platform. The board was mounted on a motorized XYZ-stage coupled with a CCD camera mounted on optical column (Fig. 9).

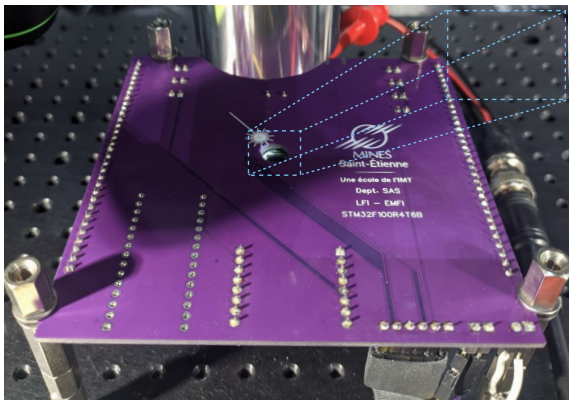


Fig. 9: Objective lenses above the targeted 32-bit MCU mounted on the custom-made platform

Target's Flash memory organization: it consists of two main components, a main memory block and an information block. The main memory block is comprised of 128 pages, each with a capacity of 1 kB. The organization of the flash memory, along with the corresponding base addresses in the

memory map, is presented in Table 1. The memory is organized as 32-bit wide memory words that can be used for storing both code and data.

Table 1: MCU's embedded Flash module organization.

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

3.4 Target preparation

As detailed on [32], there are several steps between choosing a particular target and performing a successful attack on it. This section briefly introduces the creation of a custom test platform and details different chip depackaging techniques.

3.4.1 Creating a custom support PCB

For the LFI and PEA attack scenarios presented in this paper, the target MCU needs to be powered and be able to interface with a host, to receive commands and send data back. It also needs to be soldered on a board with its backside exposed. Compared to a breadboard circuit, a PCB is more robust and compact, has lower parasitic capacitance and it is easier to mount to a test setup.

For this target, a 2-layer, 10x10 cm board was designed. The second layer can be used to route signals that would otherwise cross another track in the first layer. In this instance, this layer was defined as a ground plane, which protects against ground loops, shields against noise and facilitates the routing of ground pins [33, 34]. As shown in Fig. 9, a 7-mm diameter hole is cut out to provide access for attacks through the chip's backside.

3.4.2 Chip depackaging

To gain access to the target's interior from the backside, the attacker must first remove the resin layers and the copper plaque heatsink protecting the silicon chip inside. As the techniques presented in this paper are meant to be performed from the target's backside, i.e. from the silicon side, we will focus on the backside depackaging techniques employed by MicroPackS [35], our partner responsible for preparing our target.

Fig. 10 provides an example of the backside depackaging of a target.

Fig. 10: Backside depackaging. Left: target before decapping. Center: resin removed. Right: copper layer removed, backside silicon exposed

Chemical: This method is employed for removing the resin from either front or backside. The substances used are Nitric Acid (HNO_3), Sulfuric Acid (H_2SO_4) or a mixture of both, depending on the material of the resin and the bonding wires, as shown in Table 2. The parameters must be carefully chosen to avoid destroying the wires and ruining the chip.

The mix is applied to the MCU's surface and corrodes the resin until the heatsink is exposed. The copper layer is then removed using tweezers and the silicon is now exposed.

Mechanical: A milling machine [36] may alternatively be used to depackage a target, milling away each layer of the resin. Special care must be taken while depackaging the front side, as the circuit can be easily destroyed if the drill reaches it. The process must be completed using acidic solutions. As in the chemical method, it is common to lose samples while determining the drilling depth and position for a particular target.

For backside decapsulation, the operator can drill all the way to the copper heatsink, remove it, expose the silicon and grind it if it needs to be thinned for PE or LFI purposes. The silicon is then polished to allow PE observation and LFI through the die.

Table 2: Acidic solution parameters for two types of bonding wires

Bonding material	Gold wires	Copper wires
Solution	100% HNO_3	5/6 HNO_3, 1/6 H_2SO_4
Temperature	80°C	44°C

Laser: A laser beam can also be employed to melt and remove the resin covering the die. As with the mechanical techniques, it will destroy the circuit and the bonding wires if the beam comes into direct contact with them.

For frontside depackaging, the laser is used to remove most of the resin material and the final removal is done by chemical methods. For the backside, the laser drills all the way to the heatsink, which is then removed. This technique is faster than mechanical depackaging and allows finer movement control.

Summary: Frontside opening involves essentially removing the resin and any eventual shield (in the case of secure systems), thus giving an attacker direct access to the circuit inside. Opening from the backside, an attacker needs to remove the resin, then remove the copper layer (usually included as heatsink or as common ground) found between the resin and the die, grind and polish the silicon substrate if needed.

The target MCU used in this paper was mechanically opened from the backside using a ASAP-1 machine [36]. No thinning of the silicon layer was performed.

3.5 Vulnerability Analysis and Exploitation

In this section we look at our target's footprint, discover the vulnerabilities associated with it and propose an attack scenario that can be exploited by malicious attackers.

3.5.1 Step 1: Reconnaissance / Footprint

The key to successfully exploit or intrude a remote system is about the information we gather. The first step for attacking a system is reconnaissance, and as can be seen in Fig. 8, the target (Flash memory block) is easily identified since the surface of its regular structure covers a significant portion of the IC.

3.5.2 Step 2: Vulnerability Analysis

We went through the initial stages of characterising our target by doing footprint and reconnaissance. Now it is time to reveal its vulnerabilities and lay out an exploitation scenario.

As explained in section 2.2, writing data (made of both '0's and '1's) in the Flash Memory takes several steps. If a user wants to change the value at a specific address, the whole page containing the address must be erased. In order to avoid losing the data of other addresses, the whole content of the page must be transferred to SRAM before the page is erased. We consider this constraint as a vulnerability that can be exploited as will be explained in the next step.

3.5.3 Step 3: Exploiting the Vulnerability

Fig. 11 illustrates the vulnerability of the Flash memory due to its operating mode which is imposed by the FPEC as described in the previous step.

An attacker uses the vulnerability discovered in step 2 to induce a permanent single-bit fault in the Flash memory. In this example, the attacker achieves its goal by changing the value stored at an arbitrary address from 0x2 to 0x3 during the reading (transfer) of the whole page of the Flash memory to SRAM (1-Read Page to SRAM) before erasing its content (2-Erase Page). This takes advantage of the *bit-set* fault model obtained when exposing the Flash memory bit-line (of the corresponding faulted bit) to a laser perturbation during a read operation. When the whole page is copied again from the SRAM to the Flash memory (3-Write Page) with valid modifications, the faulted value will be copied as well, in this example, 0x2 will be permanently stored as 0x3. This behavior was observed even when the flash memory is protected by an Error Correction Code (ECC) that detects 2-bit errors and corrects 1-bit faults.

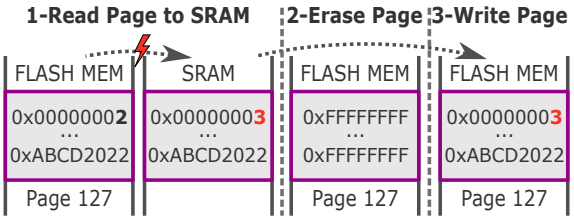


Fig. 11: Vulnerability of the Flash Memory due to its operating mode: Program (Write), Erase, and Read

Fig. 12 illustrates the attack scenario elaborated thanks to the vulnerability shown in Fig. 11. For this proof of concept, which is the main goal of this paper, we used two laser spots. We consider a password verification system limited to 3 tries. The admin password and the tries counter are stored in the same page of the Flash Memory (page 127). Any user who knows the current admin password has the permission to change it. The user has 3 tries to insert the correct admin password before the login system is blocked as a simple counter-measure against brute force attacks. Since the content of the whole page needs to be transferred to the SRAM in order to change and store the tries counter value, an attack is performed targeting the admin password and the tries counter. One laser spot illuminates the bit responsible for always keeping the tries counter value equal to 3 (i.e. without injecting this fault it would have been decreased to 2 as soon as a wrong password was tried), and another laser spot illuminates all 32-bits of the password sequentially in order to set the unknown admin password to a know value (`0xFFFFFFFF`) by bit-setting its content. At the end of the attack, this forged password (`0xFFFFFFFF`) will be the current password and since it is now a known value for the attacker, it can be changed to a new one. The end goal here is then to change the unknown admin password (`0x????????`) for a known one (`0xCAFE2021`).

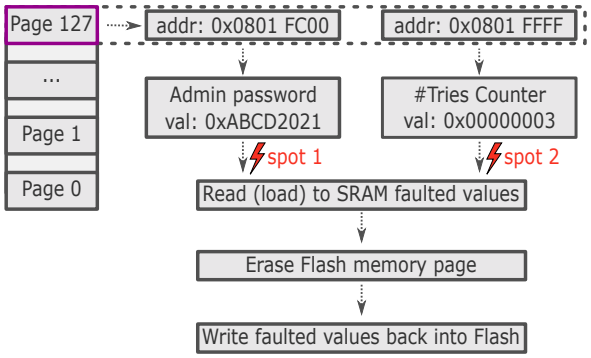


Fig. 12: Attack scenario using two laser spots. Modify the admin password and set the tries counter to be always equal to 3

3.6 Embedded Software and Aim of Experiment

Our work aimed at demonstrating the ability of LFI to inject permanent faults into a Flash memory using two laser spots.

3.6.1 Characterization of the Flash memory

First, we performed the characterization of the reading operation of the Flash memory. To that end, we wrote a dedicated test code as described by the pseudo-code in listing 1.

The code implemented in the MCU corresponds to the FPEC rules to deal with elementary operations of the Flash memory (read, erase, program). The implemented code receives via UART the address, value and page in which an operation will be performed, and the desired action. To send the commands via UART, a python script was implemented with different scenarios.

Listing 1: Program used to perform the characterization during reading operation (results reported in section 5.1)

```
// Erase a page of the flash memory
erase_flash_mem():
    page = get_from_UART("mem_page")
    FPEC_PageErase(page)

// Program the flash memory
program_flash_mem():
    addr = get_from_UART("mem_addr")
    val = get_from_UART("mem_val")
    FPEC_Program(addr, val);

// Lock the flash memory
lock_flash_mem():
    FPEC_Lock()

// Read the content of an address
read():
    addr = get_from_UART("mem_addr")
    read_from_mem(addr)
```

3.6.2 Attack scenario

We subsequently developed another code specifically tailored for the attack scenario, as shown in listing 2 and illustrated in Fig. 13.

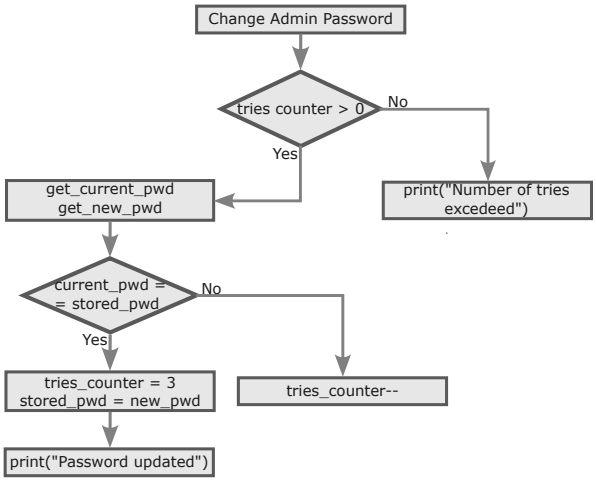


Fig. 13: Flowchart of the program used as the attack scenario

4 Characterizing the Target

4.1 Reverse engineering

Before running the experiment to overwrite the original and unknown password stored in the Flash memory, we first analyzed the addresses' disposition on the memory using PEA. Using the information gained from this analysis, we determined which parts of the Flash memory are vulnerable to LFI.

4.1.1 Employing Photoemission Analysis to locate memory addresses on the Flash

As a first step for characterizing the target, an attacker may implement a simple scenario to determine the precise position of Flash memory pages and addresses. If the attacker is unable to freely implement and execute arbitrary code on the target device, they may procure an identical MCU and use it as a clone of the original, as their internal components will be identical. Furthermore, performing this preliminary study on a clone component has the added benefit of not damaging the original target, as each Erase/Write cycle degrades the Flash cells. The Flash memory for this particular MCU has an expected endurance of 10,000 cycles [37].

The proposed scenario, illustrated by the flowchart in Fig. 14, uses the implemented FPEC_Program() and FPEC_PageErase() (listing 1) Flash memory functions. The Erase function will reset an user-selected page on the target's Flash memory, while the Program function will write a user-determined 32-bit word at a specific address. To maximize the photonic emissions, in this scenario all the words on a page were written to `0x00000000` sequentially and then the whole page was erased (i.e. all words written to `0xFFFFFFFF`).

Listing 2: Program used to perform the attack using two laser spots. Results are reported in sections 5.1 and 5.2

```
modify_admin_pwd():
  # Check if tries counter is greater than
      zero
  if tries_counter > 0:

    # Get current admin pwd and new admin pwd
        from UART
    current_adm_pwd = get_from_UART("
        current_adm_pwd")
    new_adm_pwd = get_from_UART("new_adm_pwd"
        )

    # Check if current admin pwd == stored
        admin pwd
    if current_adm_pwd == stored_adm_pwd:
        # set tries counter to default value
            (3) and store new_adm_pwd
        tries_counter = 3
        stored_adm_pwd = new_adm_pwd

        print("Password changed successfully!
            ")

        # Otherwise tell user that current
            admin pwd is incorrect and
            decrease tries_counter
    else:
      tries_counter -= 1

  # If tries counter is exhausted
  else:
    print("You have exhausted the number of
        tries.")
```

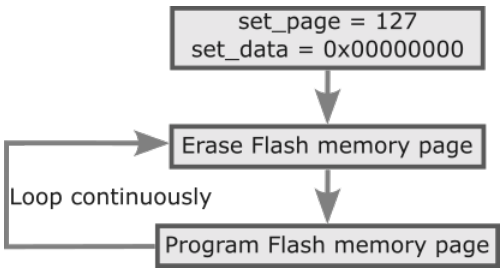


Fig. 14: The proposed scenario for PEA

This Erase-Write loop forces a particular page of the Flash memory to be set and reset continuously. As seen on Section 2.2, the Write and Erase operations both have elevated voltages and current draw. Thus, the region corresponding to the memory addresses contained in this page will emit photons, which are captured by the camera setup described on Section 3.2. The resulting image is then combined with an IR image of the Flash memory, revealing where the addresses are located by their emissions, as can be seen on Fig. 15. The IR image was taken on low-gain mode, 2 ms exposure time. The PEA image used the high-gain mode, 2,500 ms exposure time, with the camera cooled to -20°C. Both the IR photo and the PEA image were obtained using a x5 objective.

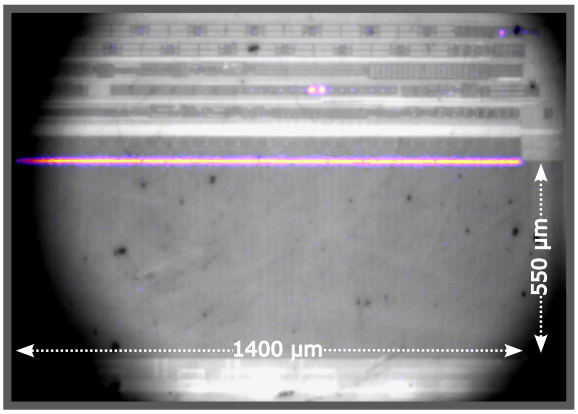


Fig. 15: Photonic emissions caused by erasing and then writing a page continuously on the MCU's Flash memory. Data written: `0x00000000`

The continuous line observed on the memory is caused by erasing page 127 then programming the 256 32-bit words contained therein to `0x00000000` in a loop. The entire operation is executed in 32 ms. With the acquisition time set to 2,500 ms,we estimate that 80 such loops take place during the capture of each raw image. Taking the total number of bits into account (256 words x 32 bits), there are 16,384 switching events in a loop (8,192 erases, 8,192 writes), making for 1,310,720 total operations captured on Fig. 15.

It has been observed that only the bits programmed to '0' emit light, those written to '1' will remain dark, as essentially no operation takes place since '1' is the reset (erased) value for this particular target.

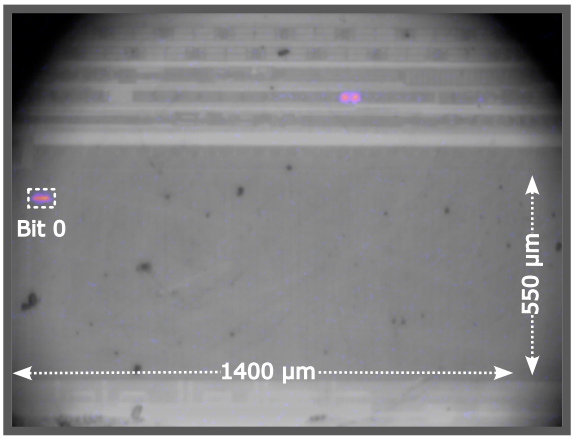


Fig. 16: The photonic emissions caused by erasing and then writing a page continuously on the MCU's Flash memory. Data written: `0xFFFFFFFE`

Leveraging this knowledge and changing the data written on each word (i.e. writing `0xFFFFFFFE`, where a single bit is '0', as shown on Fig. 16) it is possible to determine the disposition of bits in the memory, giving an attacker a more accurate spatial location of each bit of the 32-bit word. As shown on Fig. 16, the least significant bit of the 256 words is located on the left side. By writing other values, it was verified that the bits are positioned sequentially, with the most significant bit being located on the rightmost side.

4.1.2 Scanning process

The first step of laser injection characterization was using a laser beam to analyze the effects of an LFI on the Flash memory surface. We identified as shown in Fig. 17 an area equal to $1400\,\mu$m $\times$ $550\,\mu$m. The green dot represents spot 1, meaning that for this initial experiment spot 2 was OFF.

4.1.3 Delay and Pulse Width Characterization

The second step was to identify timing constraints, i.e., at which time after a trigger signal from the target was raised should the laser be activated as well as the laser pulse duration (pulse width). Fig. 18 shows that for different delays (instant of laser shot) and laser pulse widths, different addresses of the Flash memory were affected. However, it was observed that for a delay near $2{,}040\,ns$ all

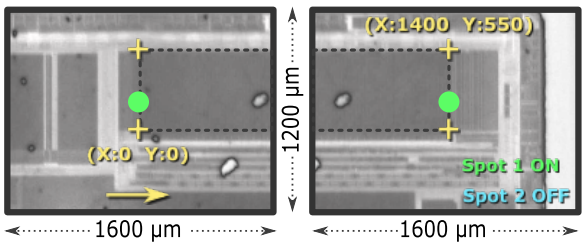


Fig. 17: Illustration of the scanning process using one laser spot. Lower bound: (X=0 μm, Y=0 μm). Upper bound: (X=1400 μm, Y=550 μm). ΔX and ΔY vary depending on the experiment

addresses were affected for all pulse widths. Therefore, in the experiments we consider a delay equal to 2,040 ns.

The function *read*() in listing 1 provides a trigger signal. This signal is drawn in purple, blue and green in Fig. 19 for the MCU working at different clock frequencies. An electronic synchronization board is then used to deliver a shot signal to the laser source, denoted laser pulse in Fig. 19 after the programmable delay selected in Fig. 18. For a more realistic scenario, the trigger signal used to synchronize the laser shots can be replaced by observing the electrical activity of the circuit using a shunt resistor [38] or by "listening" to the circuit's electromagnetic fingerprint [39].

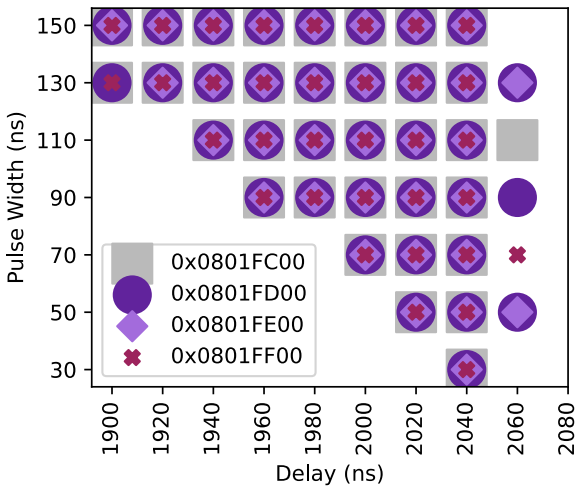


Fig. 18: Characterization of delay vs pulse width for different addresses. Power: 200 mW

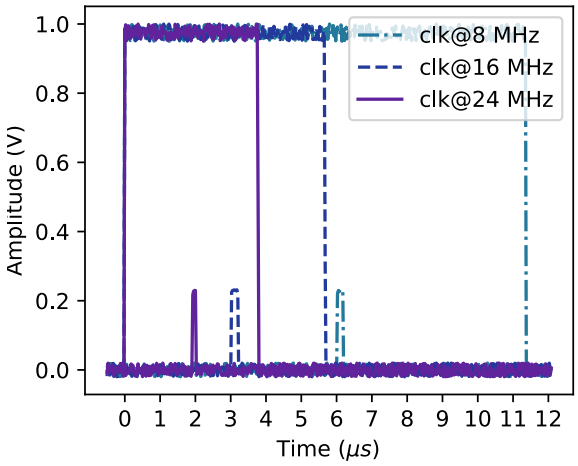


Fig. 19: Voltage waveforms sampled during the reading operation of a 32 bits word. MCU working at 8 MHz (*green*), 16 MHz (*blue*) and 24 MHz (*purple*). The trigger signal (square waveforms) is used to synchronize the laser shots with the read operations. The laser pulse with a power equal to 300 mW is represented by the waveform with voltage amplitude of ≈ 200 mV

4.1.4 Single-bit spatial effect

Fig. 20 shows the induced faults repeatability when focusing on a single-bit (bit 16 in this particular case), i.e., moving the laser spot along the x-axis of the Flash memory with a displacement step ΔX=1 μm. A normalized window of 30 μm was selected which represents the spacing between two consecutive bits. It can be observed a 100% repeatability at around 6 μm to 20 μm for pulse widths superior to 50 ns for the selected laser power (200 mW). With these values in mind, we chose 90 ns as the pulse width, which gives us a relaxed margin of around 14 μm for the sensible zone. This zone's width matches with the spot diameter of around 15 μm.

4.1.5 LFI Scanning of the MCU Flash Memory

Fig. 21 reports the LFI sensitivity scan we carried out on the Flash array surface (targeting its memory cells) with ΔX=5 μm and ΔY=100 μm displacement steps. To achieve this result we used the values for the laser pulse width, power and delay characterized in the previous steps. For this

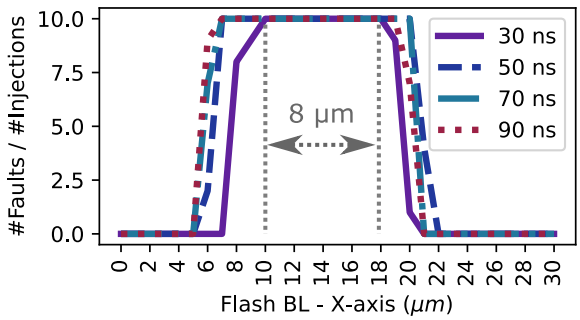


Fig. 20: Single-bit distribution of the read operation faults. Step: 1 μm. Power: 200 mW. Spot diameter: 15 μm

experiment, the value `0x00000000` was written at the memory address `0x0801FFF0`.

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 purple: for Y=100 μm the 32-bit locations are clearly visible. Changing the laser spot Y coordinates had no effect on the faulted-bit. These patterns of LFI sensitivity are similar to the ones observed by [10–12] when inducing faults during a read operation as well as [13] when inducing faults during the programming of the Flash memory.

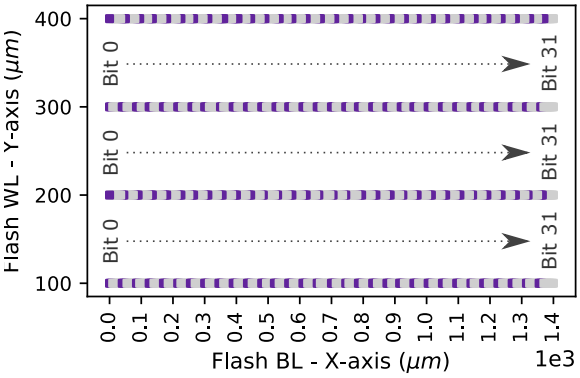


Fig. 21: LFI scanning of the MCU Flash memory: depending on the laser spot X position (from left to right) bit 0 to 31 of the read word were faulted following a *bit-set* fault model

4.2 Multi-Bit fault on Flash memory using two laser spots

After characterizing the Flash Memory with a single laser spot we proceed with its characterization using two laser spots.

4.2.1 Field of View

Fig. 22 shows the field of view (FOV) for different lenses. The background image corresponds to the infra-red picture of our target, i.e. the Flash Memory. Table 3 shows the values for the FOVs of the x5, x20 and x50 amplification lenses of our laser bench. As can be seen in Fig. 22 only the x5 lens covers the whole Flash Memory with dimension of 1400 μm × 550 μm. In this case if we want to target bit 0 with one laser spot and bit 31 with another laser spot, we are limited to use the x5 lens, which imposes a 15 μm laser spot diameter.

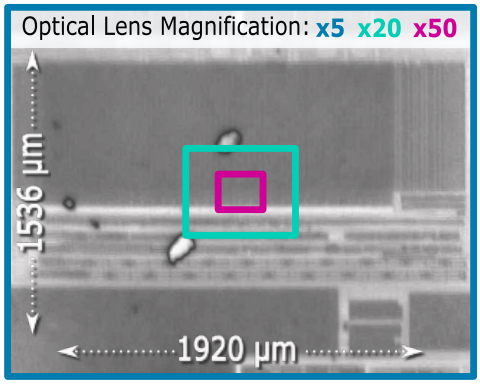


Fig. 22: Field of View for different lenses

4.2.2 Scanning process

In the next experiment, we characterized the Flash Memory with two laser spots. The Y-position is now fixed to 100 μm to focus on the X-Axis motion. The laser's Spot 1 position is set at X=0 μm. At each iteration, the Spot 1 induces a bit-set at the fixed position of 0 μm while Spot 2 moves along the X-Axis.

Results from the experiment are shown in Figure 24. The X-Axis correspond to the Spot 2 motion on the Flash area while the Y-Axis provided the associated faulted bit from bit 0 to bit 31. We observe that two *bit-set* faults are induced simultaneously on two different bit values of the

Table 3: Field of View of the IC when imaged on the Infra-Red Camera and LASER effect area without major distortions for different magnification lenses.

Lens Magnification	Field of View (WxH)	LASER effect area
x5	$1920\mu m$ $\times$ $1536\mu m$	$1600\mu m$ $\times$ $1200\mu m$
x20	$480\mu m$ $\times$ $384\mu m$	$400\mu m$ $\times$ $300\mu m$
x50	$192\mu m$ $\times$ $153\mu m$	$160\mu m$ $\times$ $120\mu m$

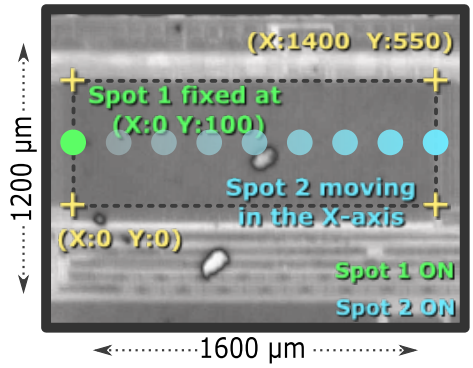


Fig. 23: Illustration of the scanning process using two laser spots

read word. For example with Laser Spot 1 and 2 at the respective position of 0 μm and 700 μm, the data read by Flash memory is `0x00020001`, meaning that bit 1 and bit 18 have been set to logical value '1'.

5 Permanently modifying the content of a Flash memory during read operation

5.1 First proof of concept: setting a password to a known all-1s using one laser spot

For the first proof of concept, we applied this fault model to iteratively set to one a 32-bit password stored in Flash. In this case there was no protection implemented and, for such a small-sized password, the attacker can try to find it by means

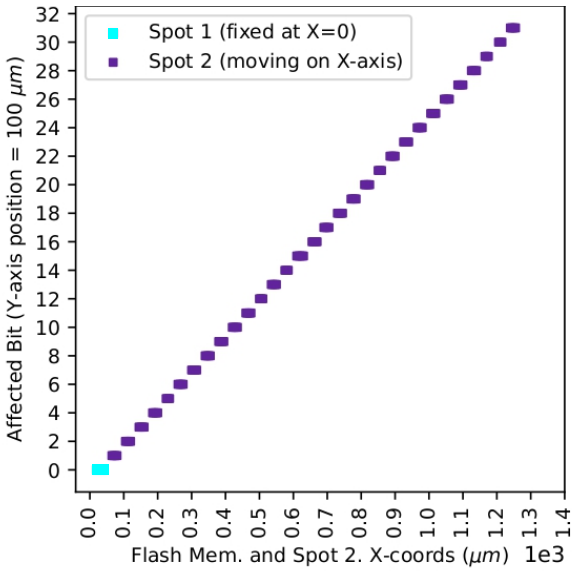


Fig. 24: Scan of the two laser spots on the MCU Flash memory. Spot 1 is fixed at X=0 μm, while Spot 2 is moved along the X-Axis. At each iteration, two faults are induced simultaneously, Spot 1 induced bit-set on bit 0 while Spot 2 induced bit-set from bit 1 to 31 depending on the X position

of brute force. However, it becomes progressively harder to achieve this goal as the size of passwords grows.

For this reason, we decided to take another approach. The code in Listing 1 was used 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 it was 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 set the password to `0xFFFFFFFF` (a known value). In a conservative approach, it was 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 '0's in the password were set to one (the bits already at '1' were not modified by the LFI). This process is illustrated in Table 4 which depicts actual data from the experiments and highlights (in red) the induced *bit-sets*. Therefore the unknown password was replaced by a known all-1s one that can be used to give access to secure information stored in the Flash memory. By performing this experiment several times, the

same result was observed in every attempt, thus giving a 100% success rate.

Table 4: Iteratively setting the password to one (faults in red)

Bit	Value (Hex)	Value (Bin)
-	0xABCD2021	1010 1011 1100 1101 0010 0000 0010 0001
1	0xABCD2023	1010 1011 1100 1101 0010 0000 0010 0011
2	0xABCD2027	1010 1011 1100 1101 0010 0000 0010 0111
3	0xABCD202F	1010 1011 1100 1101 0010 0000 0010 1111
4	0xABCD203F	1010 1011 1100 1101 0010 0000 0011 1111
6	0xABCD207F	1010 1011 1100 1101 0010 0000 0111 1111
7	0xABCD20FF	1010 1011 1100 1101 0010 0000 1111 1111
8	0xABCD21FF	1010 1011 1100 1101 0010 0001 1111 1111
9	0xABCD23FF	1010 1011 1100 1101 0010 0011 1111 1111
10	0xABCD27FF	1010 1011 1100 1101 0010 0111 1111 1111
11	0xABCD2FFF	1010 1011 1100 1101 0010 1111 1111 1111
12	0xABCD3FFF	1010 1011 1100 1101 0011 1111 1111 1111
14	0xABCD7FFF	1010 1011 1100 1101 0111 1111 1111 1111
15	0xABCDFFFF	1010 1011 1100 1101 1111 1111 1111 1111
17	0xABCFFFFF	1010 1011 1100 1111 1111 1111 1111 1111
20	0xABDFFFFF	1010 1011 1101 1111 1111 1111 1111 1111
21	0xABFFFFFF	1010 1011 1111 1111 1111 1111 1111 1111
26	0xAFFFFFFF	1010 1111 1111 1111 1111 1111 1111 1111
28	0xBFFFFFFF	1011 1111 1111 1111 1111 1111 1111 1111
30	0xFFFFFFFF	1111 1111 1111 1111 1111 1111 1111 1111

5.2 Second proof of concept: setting a password to a known all-1s using two laser spots

For the second proof of concept we used the attack scenario described in section 3.5.3 which involves the use of two laser spots to perform a successful attack.

Table 5 shows the reported values from the experiment. The first column shows the faulted bit (from '0' to '1'), the second column shows the password stored in the Flash memory. The third column represents the value of the number of tries still available to insert the correct password and the fourth column tells us if the entered password matches the password stored in the Flash memory.

As can be seen in Table 5 the protection method against brute force attacks was fairly easily bypassed thanks to the use of two laser spots. As a result, the stored password `0xABCD2021`, which is unknown to a normal user, was updated to a known password (`0xFFFFFFFF`) and then changed to one defined by the attacker, in this case `0xCAFE2021`.

Table 5: Iteratively modifying the admin password and setting the counter of tries to be always equal to 3 (induced faults are marked in red). Total iterations: 32

Bit #	Password stored in Flash	# Tries (Flash -> SRAM)	Success?
-	0xABCD2021	3 -> 3	–
2	0xEBCD2021	2 -> 3	Fail
4	0xFBCD2021	2 -> 3	Fail
6	0xFFCD2021	2 -> 3	Fail
11	0xFFED2021	2 -> 3	Fail
12	0xFFFD2021	2 -> 3	Fail
15	0xFFFF2021	2 -> 3	Fail
17	0xFFFFA021	2 -> 3	Fail
18	0xFFFFE021	2 -> 3	Fail
20	0xFFFFF021	2 -> 3	Fail
21	0xFFFFF821	2 -> 3	Fail
22	0xFFFFFC21	2 -> 3	Fail
23	0xFFFFFE21	2 -> 3	Fail
24	0xFFFFFF21	2 -> 3	Fail
25	0xFFFFFFA1	2 -> 3	Fail
26	0xFFFFFFE1	2 -> 3	Fail
28	0xFFFFFFF1	2 -> 3	Fail
29	0xFFFFFFF9	2 -> 3	Fail
30	0xFFFFFFFD	2 -> 3	Fail
31	0xFFFFFFFF	3 -> 3	Success
-	0xCAFE2021	3 -> 3	-

6 Conclusions and Discussion

The experiments presented in this paper assess the extension of the transient *bit-set* fault model to a permanent *bit-set* model during read operations. They also demonstrate the usefulness of employing Photoemission Analysis as a way to study the LFI target, i.e. the MCU's Flash memory, prior to the actual attack to speed up the target characterization phase and improve the precision of the laser beam positioning. Such analysis provides detailed information about addresses location and bit disposition in the memory. By including a section on target preparation, we provide an extensive guide on a combined PEA + LFI attack, covering the depackaging of the chosen target, designing its support board, and analyzing the results and conclusions from the attack.

The faults are induced repeatably when targeting the Flash array during a read operation. This is an extension of previous works that reported laser-induced transient *bit-sets*. Thanks to a vulnerability found in the Flash controller, this work presents an attack scenario devised to induce permanent faults by compromising the stored data during read operations.

To demonstrate the effectiveness of our new fault injection model, we implemented an attack scenario. In this scenario, we targeted an unknown

32-bit password stored in the Flash memory. Initially, we set the password to an all-1s value and subsequently changed it to a known password. Concurrently, a second laser spot was used to hold the password try counter at its original value of 3. While an attacker could potentially use a single spot to hold the try counter at its default value and attempt a brute-force attack to uncover the correct password, this method becomes increasingly impractical as the password size grows. The results we obtained were based on a laser beam diameter of $15, \mu m$ and a pulse duration of $90, ns$, which are relatively accessible settings for an attacker.

There are certain constraints associated with this attack scenario. Specifically, it requires knowledge of the password range and the addresses of the try counter variable stored in the Flash. Additionally, the fact that both the password and try counter are stored on the same page makes the attack easier to execute since it allows for similar timing requirements for accessing both pieces of data. However, this attack scenario relaxes one of the constraints found in previous works, which required prior knowledge of a user's password. In those cases, the attack relied on repeated reads and subsequent page writes to the Flash memory triggered by password updates. By eliminating the need for prior knowledge of a user password, this attack opens up the possibility of targeting systems that typically do not provide a user access profile. Overall, these factors contribute to the feasibility and potential impact of the attack, enabling exploitation of vulnerabilities in systems that lack user access profiles.

A simple firmware countermeasure that would hinder the LFI attack (though not invalidate it) is to explicitly assign different Flash memory pages for sensitive data such as the password and, in this case, the try counter. In terms of physical countermeasures that protect against both PEA and LFI attacks, one of the most straightforward approaches is to implement a metallic shield on the backside that the MCU can use to detect tampering. This solution significantly increases the cost and effort required to carry out a successful attack (BOR18). However, this type of countermeasure is expensive to implement and typically reserved for secure MCUs due to its cost.

We present a practical attack scenario which increases both the relevance and the threat level of previous works. Though we used a code-generated trigger signal to synchronize the laser shots with the target activity, it can be replaced by observing the electrical activity of the circuit using a shunt resistor or by "listening" to the circuit's electromagnetic fingerprint.

Having demonstrated a practical application for the proposed 2-bit *bit-set* model on a particular target, we may consider the generalization of this attack scenario to other MCU targets as a perspective for future works. Further investigation is necessary to determine the feasibility of 2-bit *bit-reset* attacks. If such fault model is deemed possible and repeatable, the development of scenarios employing a mix of multiple bits *bit-set* and *bit-reset* faults is another prospective work that may further aggravate the associated threat, presenting yet another angle of attack.

References

[1] Sergei Skorobogatov. Using optical emission analysis for estimating contribution to power analysis. pages 111–119, 09 2009. doi: 10.1109/FDTC.2009.39.

[2] Yadi Zhong and Ujjwal Guin. Fault-injection based chosen-plaintext attacks on multicycle aes implementations. In Proceedings of the Great Lakes Symposium on VLSI 2022, GLSVLSI '22, page 443–448, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393225.

[3] Mathieu Dumont, Pierre-Alain Moëllic, Raphael Viera, Jean-Max Dutertre, and Rémi Bernhard. An overview of laser injection against embedded neural network models. In 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), pages 616–621, 2021. doi: 10.1109/WF-IoT51360.2021.9595075.

[4] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. Proceedings of the IEEE, 94(2), Feb 2006. ISSN 0018-9219. doi: 10.1109/JPROC.2005.862424.

[5] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and

countermeasures. Proceedings of the IEEE, 100(11), Nov 2012. ISSN 0018-9219. doi: 10.1109/JPROC.2012.2188769.

[6] Tuba Kiyan, Heiko Lohrke, and Christian Boit. Comparative assessment of optical techniques for semi-invasive sram data readout on an msp430 microcontroller. pages 266–271, 11 2018. doi: 10.31399/asm.cp.istfa2018p0266.

[7] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple photonic emission analysis of aes. volume 3, pages 3–15. Springer Science and Business Media LLC, feb 2013. doi: 10.1007/s13389-013-0053-7.

[8] S. Skorobogatov. Optical fault masking attacks. In 2010 Workshop on Fault Diagnosis and Tolerance in Cryptography, pages 23–29, 2010. doi: 10.1109/FDTC.2010.18.

[9] F. Cai, G. Bai, H. Liu, and X. Hu. Optical fault injection attacks for flash memory of smartcards. In 2016 6th International Conference on Electronics Information and Emergency Communication (ICEIEC), pages 46–50, 2016. doi: 10.1109/ICEIEC.2016.7589684.

[10] Brice Colombier, Alexandre Menu, Jean Max Dutertre, Pierre Alain Moellic, Jean Baptiste Rigaud, and Jean Luc Danger. Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller. IEEE International Symposium on Hardware Oriented Security and Trust, HOST, pages 1–10, 2019. doi: 10.1109/HST.2019.8741030.

[11] Alexandre Menu, Jean Max Dutertre, Jean Baptiste Rigaud, Brice Colombier, Pierre Alain Moellic, and Jean Luc Danger. Single-bit Laser Fault Model in NOR Flash Memories: Analysis and Exploitation. Workshop on Fault Detection and Tolerance in Cryptography, FDTC, pages 41–48, 2020.

[12] 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), pages 1–7, 2020. doi: 10.1109/IOLTS50870.2020.9159712.

[13] Raphael Viera, Jean-Max Dutertre, Mathieu Dumont, and Pierre-Alain Moëllic. Permanent laser fault injection into the flash memory of a microcontroller. In 2021 19th IEEE International New Circuits and Systems Conference (NEWCAS), pages 1–4, 2021. doi: 10.1109/NEWCAS50681.2021.9462773.

[14] Brice Colombier, Lilian Bossuet, Paul Grandamme, Julien Vernay, Emilie Chanavat, Lucie Bon, and Bruno Chassagne. Multi-spot Laser Fault Injection Setup: New Possibilities for Fault Injection Attacks. In 20th Smart Card Research and Advanced Application Conference - CARDIS 2021, Lübeck, Germany, November 2021. URL https://hal.archives-ouvertes.fr/hal-03353863.

[15] G. Campardo, R. Micheloni, and D. Novosel. VLSI-Design of Non-Volatile Memories. Springer Berlin Heidelberg, 2005. ISBN 9783540265009. URL https://books.google.fr/books?id=g5jI-nyMbBMC.

[16] A. H. Johnston. Charge generation and collection in p-n junctions excited with pulsed infrared lasers. IEEE Trans. Nucl. Sci., 1993. ISSN 0018-9499. doi: 10.1109/23.273491.

[17] R. C. Baumann. Radiation-induced soft errors in advanced semiconductor technologies. IEEE Transactions on Device and Materials Reliability, 5(3):305–316, Sept 2005. ISSN 1530-4388. doi: 10.1109/TDMR.2005.853449.

[18] D. H. Habing. The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. IEEE Transactions on Nuclear Science, 12(5):91–100, Oct 1965. ISSN 0018-9499. doi: 10.1109/TNS.1965.4323904.

[19] T. C. May and M. H. Woods. Alpha-particle-induced soft errors in dynamic memories. IEEE Transactions on Electron Devices, Jan

1979. ISSN 0018-9383. doi: 10.1109/T-ED.1979.19370.

[20] C. M. Hsieh, P. C. Murley, and R. R. O'Brien. A field-funneling effect on the collection of alpha-particle-generated carriers in silicon devices. IEEE Electron Device Letters, 2(4): 103–105, April 1981. ISSN 0741-3106. doi: 10.1109/EDL.1981.25357.

[21] G. C. Messenger. Collection of charge on junction nodes from ion tracks. IEEE Transactions on Nuclear Science, 1982. ISSN 0018-9499. doi: 10.1109/TNS.1982.4336490.

[22] F. Wang and V. D. Agrawal. Single event upset: An embedded tutorial. In 21st International Conference on VLSI Design, Jan 2008. doi: 10.1109/VLSI.2008.28.

[23] Chang-Ming Hsieh, P. C. Murley, and R. R. O'Brien. Collection of charge from alpha-particle tracks in silicon devices. IEEE Transactions on Electron Devices, 30(6):686–693, Jun 1983. ISSN 0018-9383. doi: 10.1109/T-ED.1983.21190.

[24] A. G. Jordan and A. G. Milnes. Photoeffect on diffused p-n junctions with integral field gradients. IRE Trans. on Electron Devices, 1960. ISSN 0096-2430. doi: 10.1109/T-ED.1960.14688.

[25] J. L. Wirth and S. C. Rogers. The transient response of transistors and diodes to ionizing radiation. IEEE Trans. on Nuclear Science, 1964. ISSN 0018-9499. doi: 10.1109/TNS2.1964.4315472.

[26] S. Villa, A. L. Lacaita, and A. Pacelli. Photon emission from hot electrons in silicon. Phys. Rev. B, 52:10993–10999, Oct 1995. doi: 10.1103/PhysRevB.52.10993. URL https://link.aps.org/doi/10.1103/PhysRevB.52.10993.

[27] F. Stellari, F. Zappa, S. Cova, and L. Vendrame. Tools for non-invasive optical characterization of cmos circuits. In International Electron Devices Meeting 1999. Technical Digest (Cat. No.99CH36318), pages 487–490, 1999. doi: 10.1109/IEDM.1999.824199.

[28] Jeff Bude, Nobuyuki Sano, and Akira Yoshii. Hot-carrier luminescence in Si. Phys. Rev. B, 45:5848–5856, Mar 1992. doi: 10.1103/PhysRevB.45.5848. URL https://link.aps.org/doi/10.1103/PhysRevB.45.5848.

[29] A.D. Trigg. The infrared photoemission microscope as a tool for semiconductor device failure analysis. In Proceedings of the 1997 6th International Symposium on the Physical and Failure Analysis of Integrated Circuits, pages 21–26, 1997. doi: 10.1109/IPFA.1997.638067.

[30] T. Ishii. Functional failure analysis technology from backside of vlsi chip. Proc. of the 20th Int. Symp. For Testing and Failure Analysis, ASM Int., 1994, pages 41–47, 1994. URL https://cir.nii.ac.jp/crid/1573668925480841344.

[31] Nidish Vashistha, M Tanjidur Rahman, Olivia P. Dizon-Paradis, and Navid Asadizanjani. Is backside the new backdoor in modern socs?: Invited paper. In 2019 IEEE International Test Conference (ITC), pages 1–10, 2019. doi: 10.1109/ITC44170.2019.9000127.

[32] Rodrigo Silva Lima, Raphael Viera, Jean-Max Dutertre, Anne-Lise Ribotta, Matthieu Pommies, and Anthony Bertrand. Target preparation methodology for semi-invasive attacks on microcontrollers. pages 1–7, 2022. doi: 10.1109/PAINE56030.2022.10014827.

[33] Lee W. Ritchey, John Zasio, and Kella J. Knack. Right the First Time: A Practical Handbook on High Speed PCB and System Design. Speeding Edge, 2006.

[34] Peter Wilson. The Circuit Designer's Companion. Newnes, 2018.

[35] Micro-PackS - A technical platform. https://www.pf-micropacks.org/en/micro-packs/la-plate-forme. Accessed: 2022-07-21.

[36] Analog Selected Area Preparation System - ASAP-1. https://www.ultratecusa.com/wp-content/uploads/2020/03/

ASAP-1-Brochure-low-res-S-10-07.pdf. Accessed: 2022-07-26.

[37] Datasheet STM32F100x4, STM32F100x6, STM32F100x8, STM32F100xB. STMicroelectronics, 11 2016. Rev. 9.

[38] Oliver Bernard M. Ed. Electronic Measurements and Instrumentation. McGraw-Hill Inc., Auckland, 1985.

[39] Xu Zhijian, Tang Qiang, Song Yanyan, Zhang Dongyao, and Zhou Changlin. Side channel leakage information based on electromagnetic emission of stm32 micro-controller. In 2019 12th International Workshop on the Electromagnetic Compatibility of Integrated Circuits (EMC Compo), pages 204–206, 2019.