# Fault injection Attacks on IoT devices: Threats & countermeasures

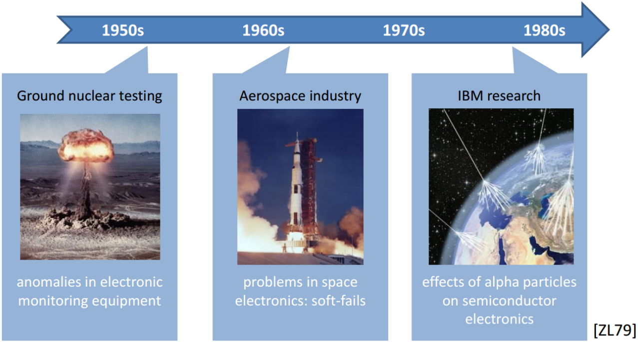
Vanthanh Khuat

April 2025

# Table contents

# 1. Introduction

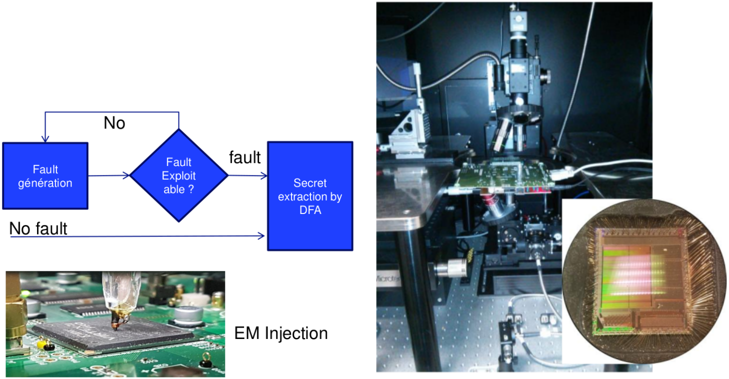
## Fault injection: from safety problem




- When operating in the hazard environments, electronic devices may not function correctly, and accidents may happen.

# 1. Introduction

## Fault injection: to security problem

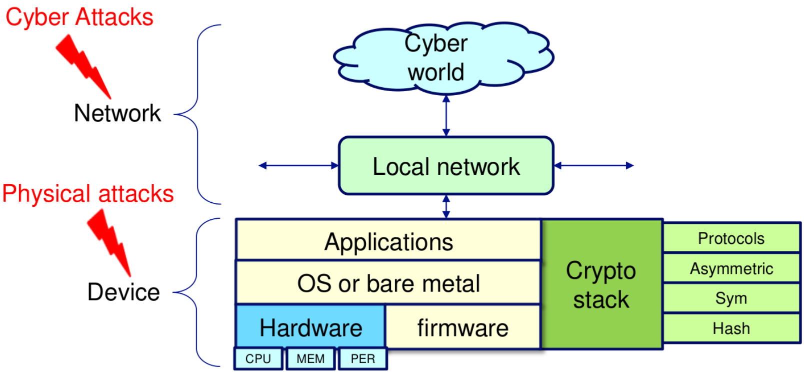



(a) Electromagnetic wave (b) Laser pulse

- Physical disturbances are used to extract information from electronic devices.

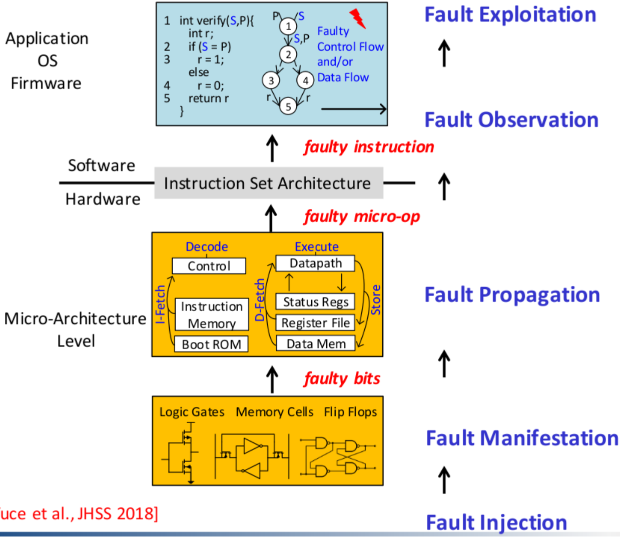
# 1. Introduction

## Physical attack vs Cyber attack



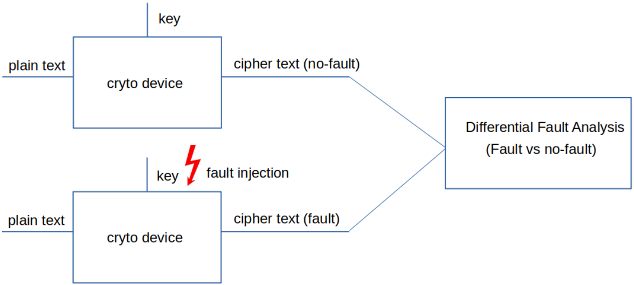
# 1. Introduction

## Fault injection anatomy




[Yuce et al., JHSS 2018]

# 1. Introduction
## Fault injection




- Fault Injection (FI) is an active side-channel attack in which the attacker induces stress to the target, forcing it to produce a faulted result.
- The faulted result is further used to extract secret information by differential fault analysis (fault vs no-fault).

There are several common techniques for FI.

- Clock glitch or Voltage glitch[1]
- Electromagnetic Fault Injection[2]
- Laser fault injection[3]

[1] **barenghi2009low**; **balasch2011depth**.
[2] **riviere2015high**; **beckers2019characterization**.
[3] **skorobogatov2002optical**; **dutertre2019experimental**.

# 1. Introduction

## Fault injection techniques comparison

| | Control on injection time | Control on localization | Control on # faulted bits | reproducibility | cost | ease of use |
|---|---|---|---|---|---|---|
| Clock glitch (digital) | very good | low | very good | good | low | very good |
| Power glitch (analog) | good | low | very good | good | average | good |
| Overclocking Underpowering Temperature | low | low | good | good | low | good |
| EM pertubation | good | average | very good | good | average | good |
| Laser | good | very good | very good | good | high | good |

# Table contents

# 2. Experimental setup and methodology

## Experimental setup Power Glitch

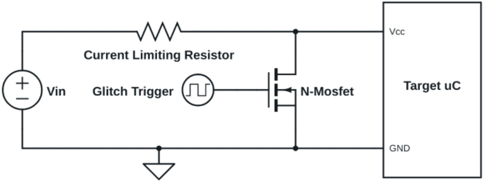



Figure: Voltage glitch experimental setup[4]

- a DDS signal generator outputs an arbitrary waveform from a software-defined set of parameters.

[4] **bozzato2019shaping**.

# 2. Experimental setup and methodology

## Experimental setup Electromagnetic Fault Injection (EMFI)

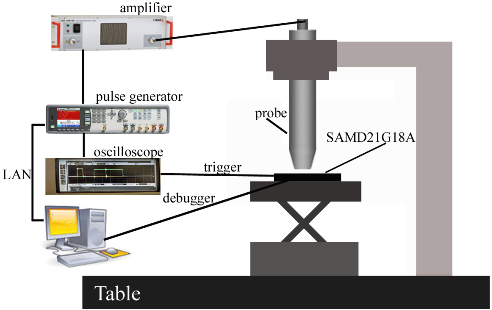


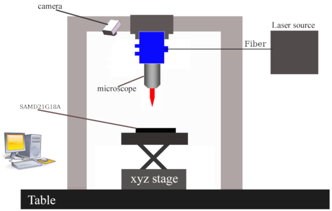

Figure: EMFI experimental setup[5]

- The Micro-controller (MCU) was configured to work at 12 MHz, with zero waitstate.
- The MCU was not **depackaged** and the Electromagnetic (EM) pulse was injected from the front side.

[5] **khuat2024software**; **khuat2021multiple**.

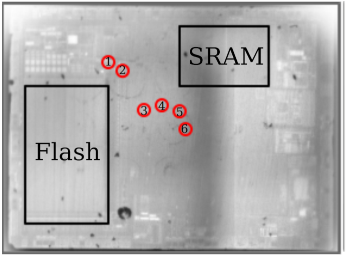
# 2. Experimental setup and methodology

## Experimental setup Laser Fault Injection (LFI)



(a)

(a) Laser bench



(b)

(b) Microchip back-side image

- Wavelength: 1064 nm, power: 0 - 3 W, Pulse Width (PW): 5 ns - 1 s (more details can be found in[6]).
- The MCU was depackaged and the laser pulse was injected from the back side.
- The MCU was configured to work at 12 MHz, with zero waitstate.

[6] **dutertre2019experimental**; **khuat2021laser**.

# 2. Experimental setup and methodology

## Device under test debugging

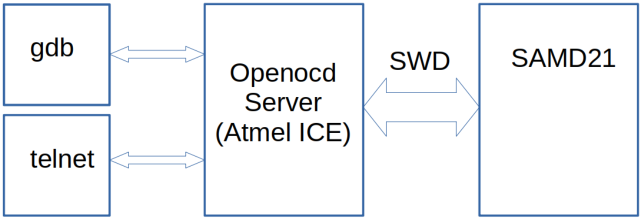



Figure: Device under test debugging

- The scheme allows debugging and stopping device to collect data automatically.

# 2. Experimental setup and methodology

## Test procedures

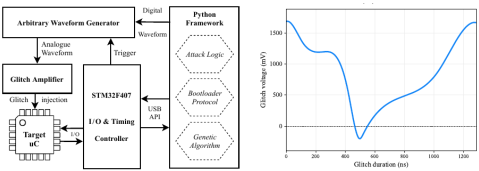
A test follows three main steps:

- (1) the target is reset and all system 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.

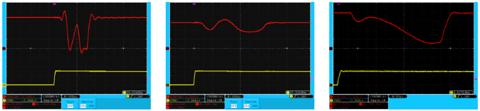
For each injection parameter, 100 tests are performed. At the beginning, a test without Fault Injection Attacks (FIA) was performed to make sure the program functions correctly and the data is used as the reference.

# Table contents

# 3.1. Clock or power glitch [7]




(a) V-FI setup for generating arbitrary glitch waveforms



(a) STM32F373 (b) TI MSP430F5172 (c) Renesas 78K0R

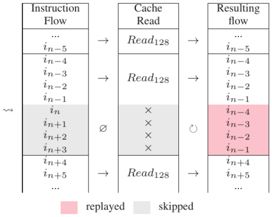
(b) Oscilloscope trace of the voltage glitch for the STM32F373 (a), the TI MSP430F5172 (b) and the Renesas 78K0R (c).

- Different shapes of voltage glitch were achieved.
- Be able to bypass security protections of several types of MCU.

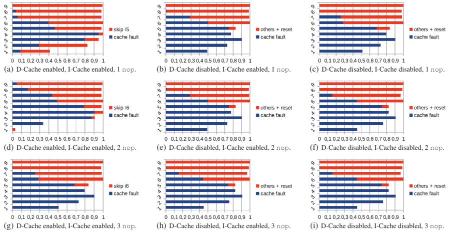
[7] **bozzato2019shaping**.

# 3.2 EMFI

## Faulting cache operation[8]




(a) EMFI faulting cache read



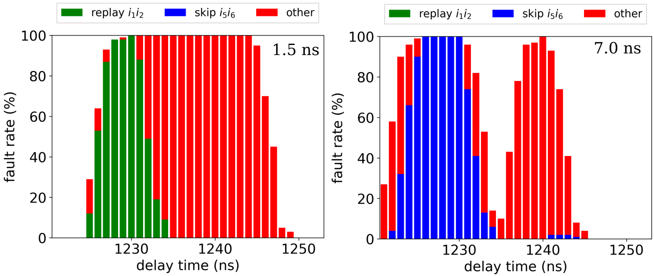

(b) Fault model probability depending on injected power

- The target is ARMv7-M architecture.
- Be able to fault cache read process with success rate up to 96%.

[8] **riviere2015high**.

# 3.2 EMFI

## Multiple and reproducible fault[9]




(a) (b)

- (a) EM pulse with pulse width of 1.5 ns -> replay of two instructions with fault rate up to 100%.
- (b) EM pulse with pulse width of 7.0 ns -> skip of two instruction with fault rate up to 100%.

[9] **khuat2021multiple**.

# 3.2 EMFI

## Multiple and reproducible fault: hypothesis

| Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Buffer1 | $(i_1, i_2)$ | | $(i_5, i_6)$ | | | | $(i_9, i_{10})$ | |
| Buffer2 | $(i_3, i_4)$ | | | | $(i_7, i_8)$ | | | |

(a) normal execution instruction buffering process

| Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Buffer1 | $(i_1, i_2)$ | | $(i_1, i_2)$ | | | | $(i_9, i_{10})$ | |
| Buffer2 | $(i_3, i_4)$ | | | | $(i_7, i_8)$ | | | |

(b) EMFI-induced replay of two instructions on buffer1

| Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Buffer1 | $(i_1, i_2)$ | | *(nop, nop)* | | | | $(i_9, i_{10})$ | |
| Buffer2 | $(i_3, i_4)$ | | | | $(i_7, i_8)$ | | | |

(c) EMFI-induced skip of two instructions on buffer1

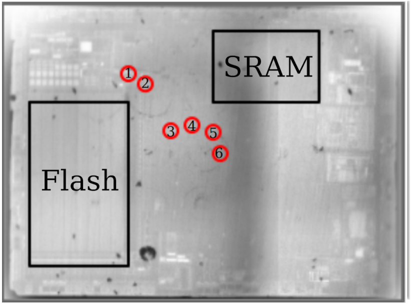
| Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Buffer1 | $(i_1, i_2)$ | | $(i_5, i_6)$ | | | | $(i_9, i_{10})$ | |
| Buffer2 | $(i_3, i_4)$ | | | | *(nop, nop)* | | | |

(d) EMFI-induced skip of two instructions on buffer2

- There exist two 32-bit buffers at the flash interface, each of them is updated every four clock cycles.
- Replay fault is caused by EM-induced prevention of buffer updating process.
- Skip fault is caused by EM-induced instructions corruption.

# 3.3 Laser fault injection

## LFI Faults at six positions

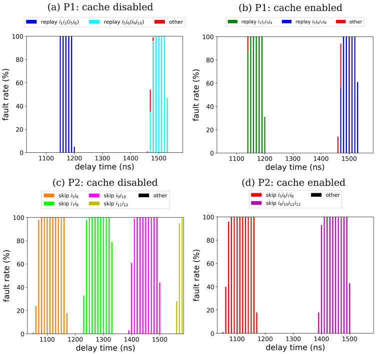



(b)

- Laser power: 1.5 W, PW: 50 ns.
- Six positions marked with red circular shapes with different fault behavior were found.

# 3.3 Laser fault injection

## from The flash interface to the execution pipeline: P1 and P2



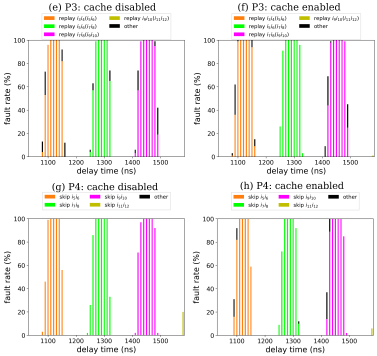

- The fault is related to block of two or four instructions depending on the cache operation mode;
- Two fault models: skip and replay of instruction block are observed;

- The fault behavior is the same with results obtained in[10], in which we ascribed the fault to impact of EMFI and LFI to the **Flash interface buffer**.

[10]**vkhuat_emc_europe_2021**; **vkhuat_dsd_2021**.

# 3.3 Laser fault injection

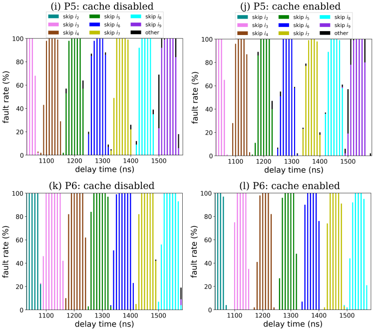
## from The flash interface to the execution pipeline: P3 and P4




- The fault is related to a block of two instructions for both cache operation modes;
- Two fault models of skip and replay of a block of two instructions are observed.

# 3.3 Laser fault injection

## from The flash interface to the execution pipeline: P5 and P6




- The fault is related to a single instruction;
- Single instruction skip was obtained at position P5 and P6.
- There is a phase shift of one clock cycle between the fault at position 5 and 6.

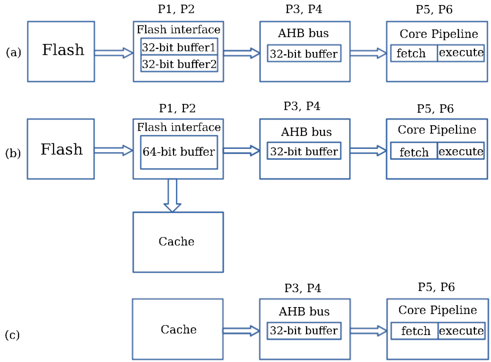
# 3.3 Laser fault injection

## Fault identification

- **Position P1**: the replay of a block of instructions due to laser-induced prevention of the Flash interface buffer updating process;
- **Position P2**: the modification of a block instructions (including skip) due to laser-induced bit corruption of instruction's opcodes in the Flash interface buffer;
- **Position P3**: the replay of two instructions due to laser-induced prevention of loading data into the AHB bus;
- **Position P4**: the modification of two instructions (including skip) due to laser-induced bit(s) corruption of instructions loaded into the AHB bus;
- **Position P5**: the modification of a single instruction (including skip) due to laser-induced fault in the core pipeline fetch stage;
- **Position P6**: the modification of a single instruction (including skip) due to laser-induced fault in the core pipeline execution stage.

# 3.3 Laser fault injection

## Proposed core architecture




- (a) cache disabled;
- (b) cache enabled: cache miss;
- (c) cache enabled: cache hit.

# 3.4. Fault model at bit level

## test code

```
lsl r0,r0, #0x00
lsl r0,r0, #0x00
lsl r0,r0, #0x00
lsl r0,r0, #0x00
```

(a) bit-set detection

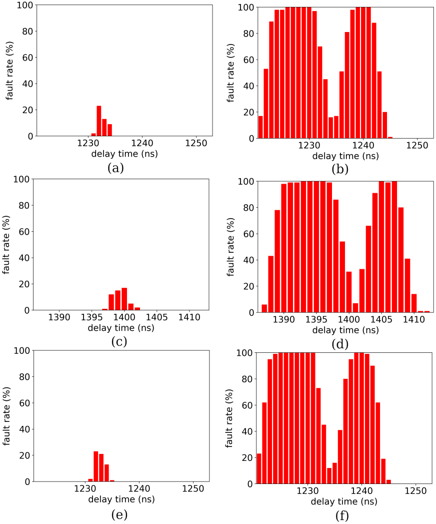
```
sub r7,r7, #0xff
sub r7,r7, #0xff
sub r7,r7, #0xff
sub r7,r7, #0xff
```

(b) bit-reset detection

- The opcode of lsl r0,r0,#0x00 is 0x0000 (all bits' values are 0 )
- The opcode of sub r7,r7,#0xff is 0x3fff (most of the bits'values are 1)

# 3.4. Fault model at bit level

## fault rate




- Bit set fault detection
  - (a) 32-bit buffer1
  - (c) 32-bit buffer2
  - (e)64-bit buffer
- Bit reset fault detection
  - (b) 32-bit buffer1
  - (d) 32-bit buffer2
  - (f) 64-bit buffer.
- At bit level the fault is rather bit-reset than bit-set.

# Table contents

# 4.CM against fault injection

## Introduction

- Hardware countermeasure (CM)
  - is implemented by adding or changing one part of the device.
  - can be very effective to some specific types of attacks.
  - adds more cost to the devices and is hard to update.
- Software CM
  - is implemented by changing the software of the device.
  - adds no cost to the devices.
  - can be updated constantly.
  - can be adapted to different types of attacks.
  - increases the size of the program and brings down the speed.

# 4. CMs against fault injection

## Redundancy-based CM

Table: Redundancy-based software defense against FI[11]

| Initial code | Redundancy-based defense |
|---|---|
| `ldr r1, [r0]` | `ldr r1, [r0]`<br>`ldr r2, [r0]`<br>`cmp r1, r2`<br>`bne <error>` |

[11] **barenghi2010countermeasures**.

# 4. CMs against fault injection

## Based on code transformation and duplication

Table: Duplication-based software defense against the single instruction skip Fault Model (FM)[12]

| **Initial code** | **Idempotent instr.** | **Duplication** |
|---|---|---|
| `add r1, r0, #1` | | `add r1, r0, #1`<br>`add r1, r0, #1` |
| `add r1,r1,r2` | `add r3, r1, r2`<br>`mov r1, r3` | `add r3, r1, r2`<br>`add r3, r1, r2`<br>`mov r1, r3`<br>`mov r1, r3` |

[12] **moro2014formal**.

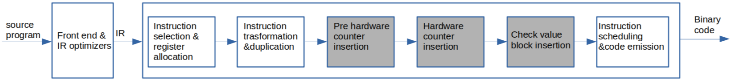
# 4. CMs against fault injection

## Based on code transformation and duplication: **limitations**

- The existing CMs were designed for single instruction skip only.
- Recently, there are many works demonstrating that multiple instruction(s) skips can be easily obtained with either EMFI or LFI.

# 4. CMs against fault injection

## CM : Based on code duplication and dedicated counter[13]

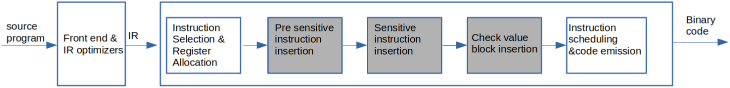



- The CM is effective to both single and multiple instruction(s) skip.
- The code size overhead is 293% the execution time overhead is 201%.

[13] **khuat2024software**.

# 4. CMs against fault injection

## CM : using a Sensitive Instruction as a sensor




- The CM is effective to both single and multiple instruction(s) skip.
- The code size overhead is 200% the execution time overhead is 149.5%.

# Table contents

# 5. Conclusions & future works

## Conclusions

- FI poses significant threat on MCU.
- Different types of fault models such as: bit flip, instructions skip, instruction replay were obtained with EMFI and LFI.
- Up to hundreds of instruction skips were achieved with LFI.

# 5. Conclusions & future works

## Future works

- Study the mechanism of the faults at physical level.
- Investigate more on the threat of the fault and design effective counter-measures against the fault.
- develop tools or software for automation of hardening code at assembly level.

Thanks for your attention!

&

Questions?