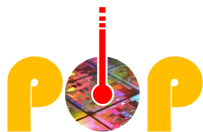


# Laser Fault Injection on power off devices

Paul Grandamme<sup>1,2</sup>, Lilian Bossuet<sup>1</sup>, Jean-Max Dutertre<sup>2</sup>

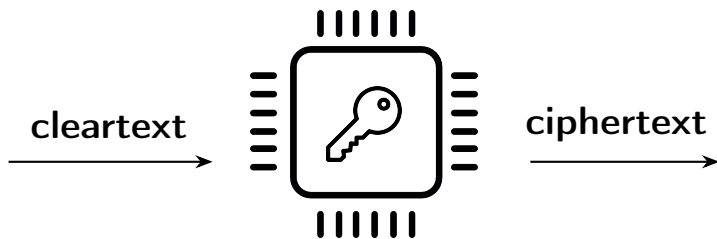
<sup>1</sup>Laboratoire Hubert Curien, Université Jean Monnet

<sup>2</sup>Mines Saint-Étienne, CEA-Leti

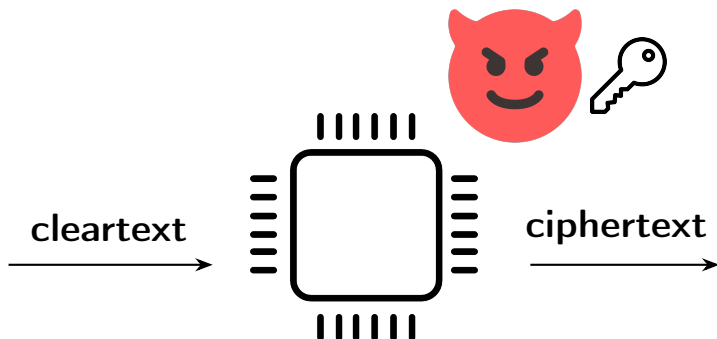


September 11, 2024

# Symmetric crypto



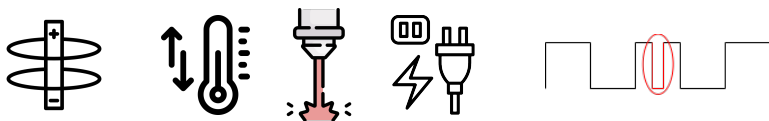
# Symmetric crypto



# Fault Attack and Fault Model

## Fault Attack

Disturbing the integrated circuit to **modify its behavior** in order to obtain information or disable internal protection mechanisms



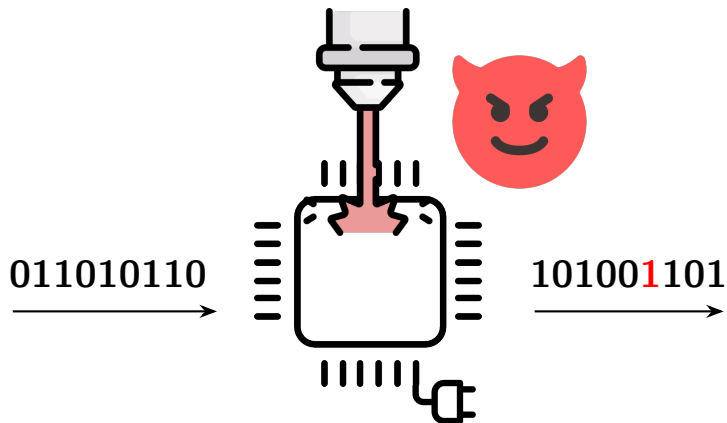
## Fault Model

Abstractions used to model the **impact** of a fault attack on a **circuit**

# Context

## State of the art

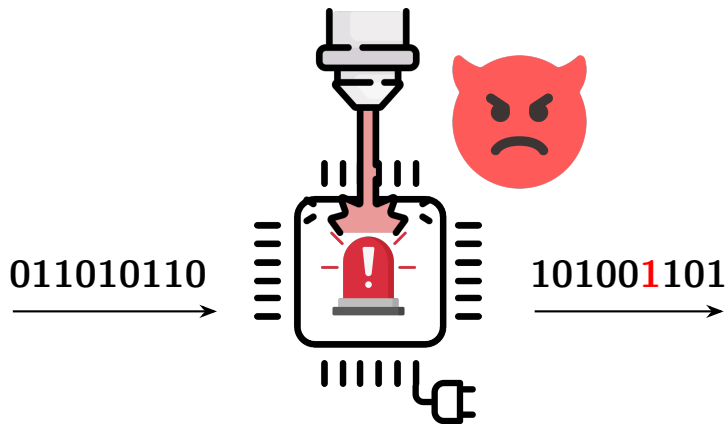
Almost all of the attacks performed on powered devices



# Context

## State of the art

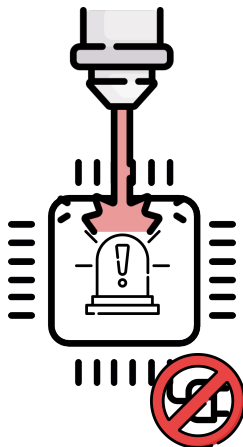
Almost all of the attacks performed on powered devices



# Context

## State of the art

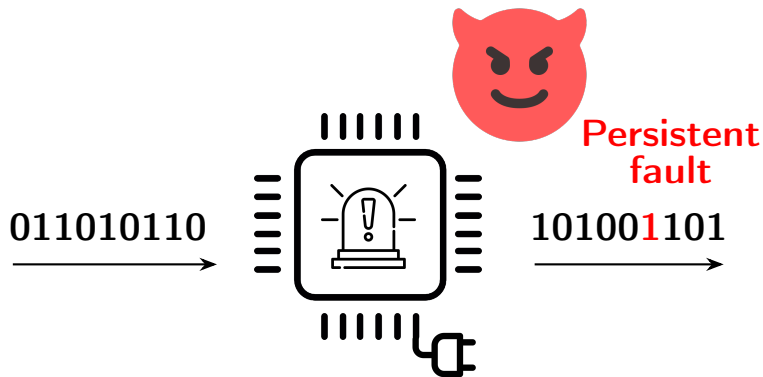
Almost all of the attacks performed on powered devices



# Context

## State of the art

Almost all of the attacks performed on powered devices





# Context

## Problem

How to bypass the sensors?

# Context

## Problem

How to bypass the sensors?

## Solution

Attack unpowered devices

# Context

## Problem

How to bypass the sensors?

## Solution

Attack unpowered devices

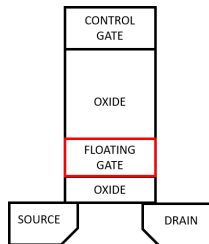
## Ideas

- Damage the device: break the sensors
- Corrupt stored information: Non-Volatile Memories (Flash)

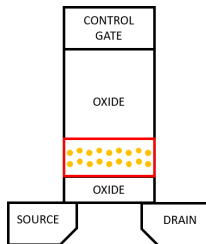
# Plan

- ① Flash memories
- ② Fault model
- ③ Experiments & Results
- ④ Conclusion

# Floating gate transistors in Flash memories

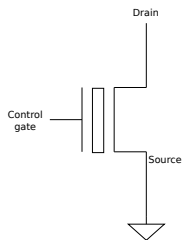


Discharged cell.



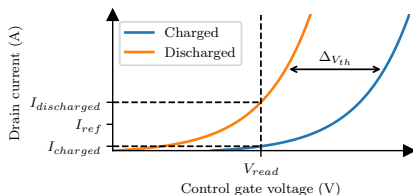
Charged cell.

# Read mechanism



Convention:

- Charged = '0'
- Discharged = '1'

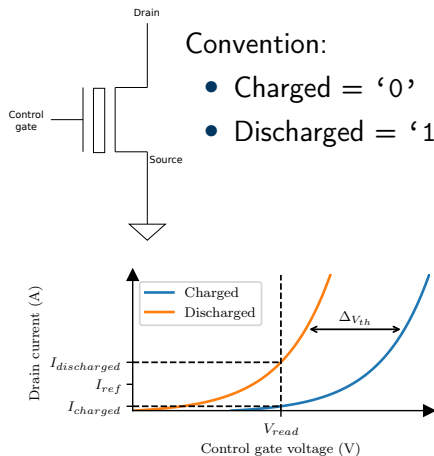


I-V characteristics.

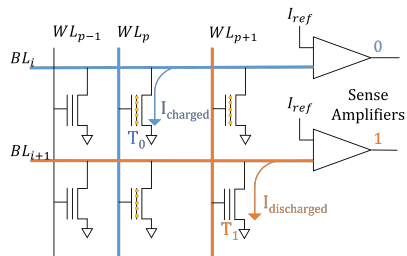
# Read mechanism

Convention:

- Charged = '0'
- Discharged = '1'



I-V characteristics.



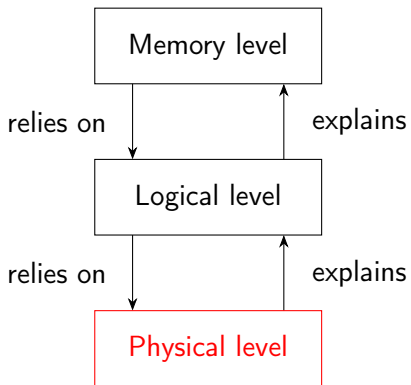
Read operation of a Flash memory.

# Plan

- ① Flash memories
- ② **Fault model**
- ③ Experiments & Results
- ④ Conclusion



# Abstraction levels



# Physical level

- Laser beam energy  $\Rightarrow$  Temperature increase  
 $\Rightarrow$  Floating gate discharge

Intensity of the laser beam:

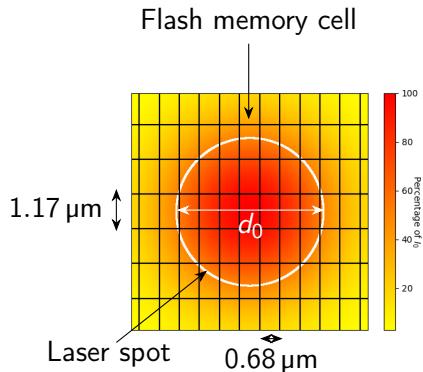
$$I(r) = I_0 \cdot e^{-\frac{2r^2}{\omega_0^2}} \quad (1)$$

with:

$$\omega_0 = \frac{2\lambda}{\pi \times NA} \quad (2)$$

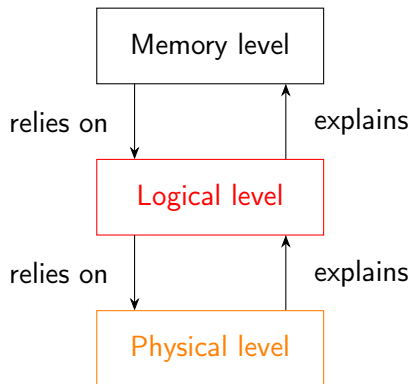
Full-Width-at-Half-Maximum  
 criterion  $\Rightarrow$

$$d_0 = \omega_0 \sqrt{\frac{\ln 2}{2}} \approx 5 \mu\text{m} \quad (3)$$



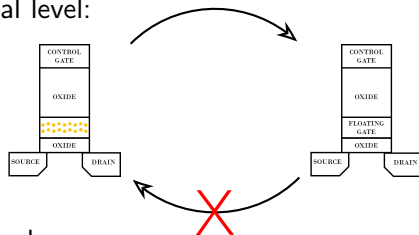
Heatmap induced by the laser exposition (numerical simulation with  $\lambda = 1064 \text{ nm}$  and  $NA = 0.16$ ).

# Abstraction levels

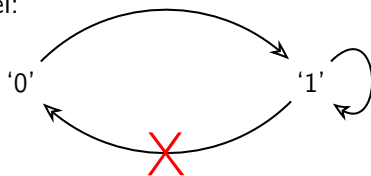


# Logical level

From the physical level:

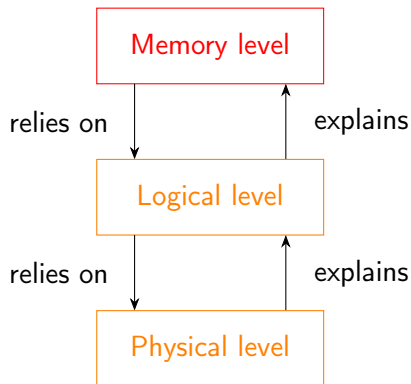


To the logical level:



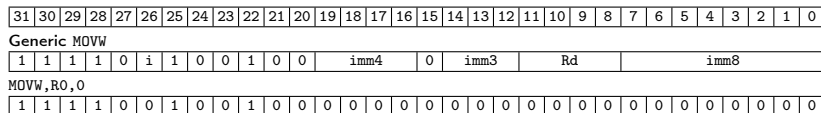
- ⇒ Unidirectional and data-dependent fault model
  - ⇒ Bitsets for our target
  - ⇒ Persistent faults (possible to reprogram the memory)

# Abstraction levels



# Memory level

## Code corruption (ARMv7 ISA)



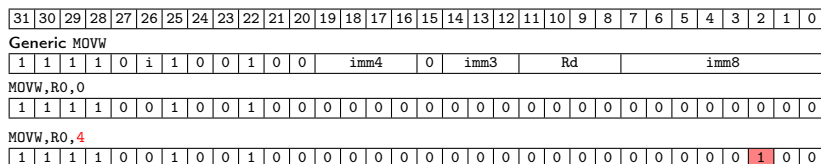
Examples of possible corruptions on a MOVW instruction<sup>1</sup>.

---

<sup>1</sup>Brice Colombier et al. "Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller". In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019, McLean, VA, USA, May 5-10, 2019*. IEEE.

# Memory level

## Code corruption (ARMv7 ISA)



Examples of possible corruptions on a MOVW instruction<sup>1</sup>.

<sup>1</sup>Brice Colombier et al. "Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller". In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019, McLean, VA, USA, May 5-10, 2019*. IEEE.

# Memory level

## Code corruption (ARMv7 ISA)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Generic MOVW																																		
1	1	1	1	0	i	1	0	0	1	0	0	imm4				0	imm3			Rd				imm8										
MOVW,R0,0																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
MOVW,R0,4																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
MOVW,R1,0																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Examples of possible corruptions on a MOVW instruction<sup>1</sup>.

<sup>1</sup>Brice Colombier et al. "Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller". In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019, McLean, VA, USA, May 5-10, 2019*. IEEE.



# Memory level

## Code corruption (ARMv7 ISA)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Generic MOVW																																		
1	1	1	1	0	i	1	0	0	1	0	0	imm4				0	imm3			Rd				imm8										
MOVW,R0,0																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
MOVW,R0,4																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
MOVW,R1,0																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
MOVW,R1,0																																		
1	1	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Examples of possible corruptions on a MOVW instruction<sup>1</sup>.

<sup>1</sup>Brice Colombier et al. "Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller". In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019, McLean, VA, USA, May 5-10, 2019*. IEEE.

# Memory level

## Code corruption (ARMv7 ISA)

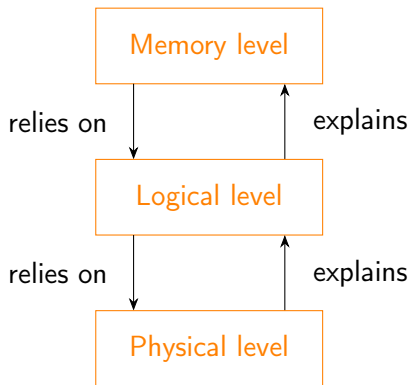
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Generic MOVW																																		
1	1	1	1	0	i	1	0	0	1	0	0	imm4				0	imm3			Rd				imm8										
MOVW,R0,0																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
MOVW,R0,4																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
MOVW,R1,0																																		
1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
MOVW,R0,0																																		
1	1	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Examples of possible corruptions on a MOVW instruction<sup>1</sup>.

⇒ Also possible to corrupt permanent data

<sup>1</sup>Brice Colombier et al. "Laser-induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-bit Microcontroller". In: *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019, McLean, VA, USA, May 5-10, 2019*. IEEE.

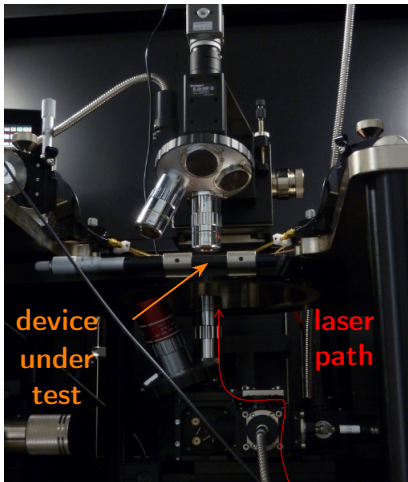
# Abstraction levels



# Plan

- ① Flash memories
- ② Fault model
- ③ Experiments & Results**
- ④ Conclusion

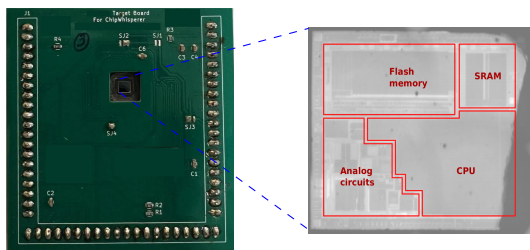
# Laser fault injection setup



- Same setup as powered laser fault injection
- 1064 nm laser source (near-IR)
- Pulse of 0.9 s
- 5  $\mu\text{m}$  spot with a x20 magnifying lens
- Backside view with an IR camera

## Hardware target

- STM32F1 open on the backside (32-bit microcontroller)
- Same device preparation as powered laser fault injection
- 128 pages of 1 kB of Flash memory
- 2048 bitlines and 512 wordlines
- No memory protection mechanism



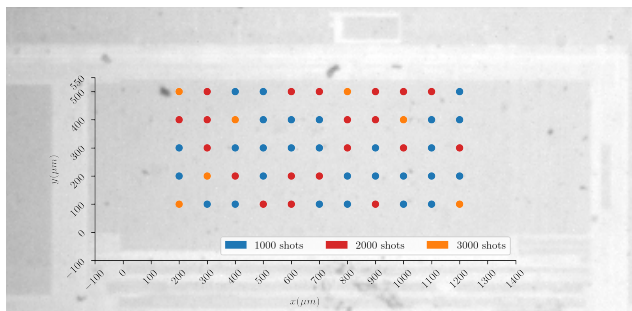
Board (left) and infrared image (right) of the hardware target.

## Power OFF mapping of injected faults

```
Require:  $X_{min}, X_{max}, X_{step}, Y_{min}, Y_{max}, Y_{step}$   
1: for  $x \in \text{range}(X_{min}, X_{max}, X_{step})$  do  
2:   for  $y \in \text{range}(Y_{min}, Y_{max}, Y_{step})$  do  
3:     reset target memory  
4:     do  
5:       move laser to (x,y)  
6:       power target off  
7:       for  $i \in [0, \dots, 999]$  do  
8:         laser shot  
9:         power target on  
10:        dump target memory  
11:      while #faults == 0  
12:        mapping[x][y] = #faults  
13: return mapping[x][y]
```

## Power OFF laser sensitivity map

- Memory initialized to 0x00000000 (programmed) before the laser exposure

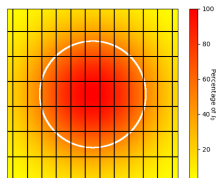
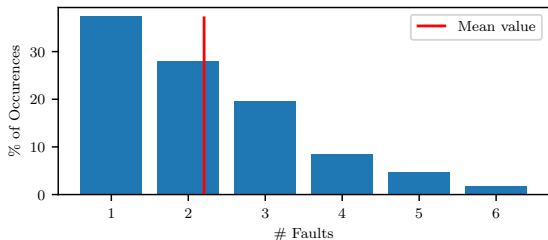


Mapping of obtained faults.  $P_{laser} = 1\text{ W}$ ,  $f_{laser} = 1\text{ Hz}$ ,  $T_{pulse} = 0.9\text{ s}$ .

- ⇒ Fault address and fault value known for each position
- bitsets obtained



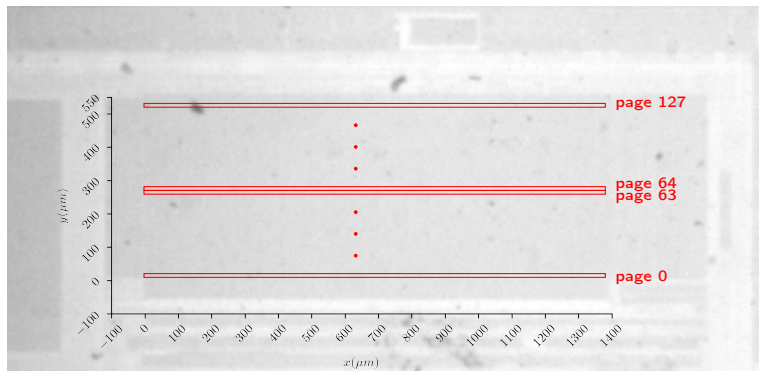
# Experimental distribution



Experimental distribution of the number of injected faults.

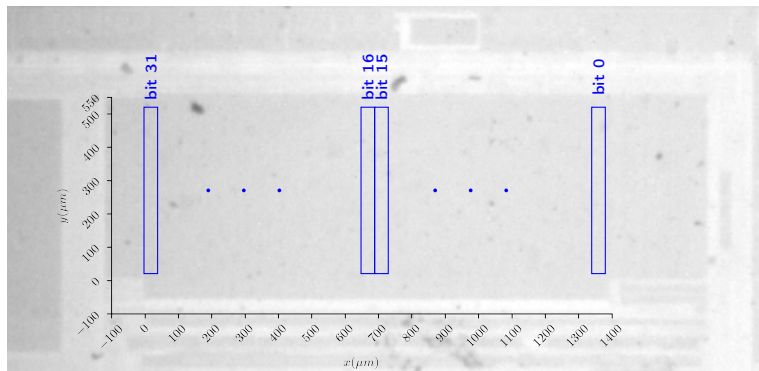
- 2.2 faulty bits on average on 0x00000000 data (median value of 2.2)
- **Single bit faults** in 33% of cases
- No correlation between #shots and #faults

# Reverse engineering 1/2



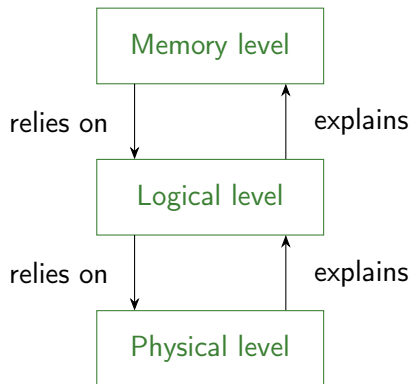
Reverse engineering of the Flash memory mapping: page-level.

## Reverse engineering 2/2



Reverse engineering of the Flash memory mapping: bit-level.

# Abstraction levels



# Plan

- ① Flash memories
- ② Fault model
- ③ Experiments & Results
- ④ Conclusion**

# Conclusion

- Switching Off your Device Does Not Protect Against Fault Attacks
- Semiconductor manufacturers must include systematically memory protection mechanisms
- Same countermeasures for Power ON and Power OFF

# Thank you for your attention !

This work was supported by research grants of the projects POP (ANR-21-CE39-0004) and PROPHY (ANR-22-CE39-0008) from the french Agence Nationale de la Recherche

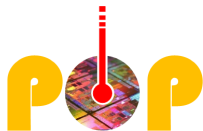


# Laser Fault Injection on power off devices

Paul Grandamme<sup>1,2</sup>, Lilian Bossuet<sup>1</sup>, Jean-Max Dutertre<sup>2</sup>

<sup>1</sup>Laboratoire Hubert Curien, Université Jean Monnet

<sup>2</sup>Mines Saint-Étienne, CEA-Leti

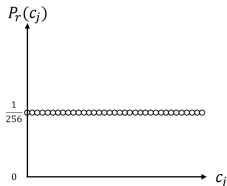
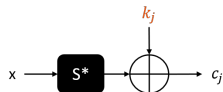
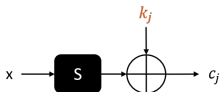


September 11, 2024

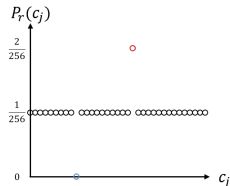


# Persistent Fault Analysis: Principle<sup>2</sup> (CHES 2018)

- Persistent fault injection on the S-box
- Statistical study on the bytes of the ciphertexts



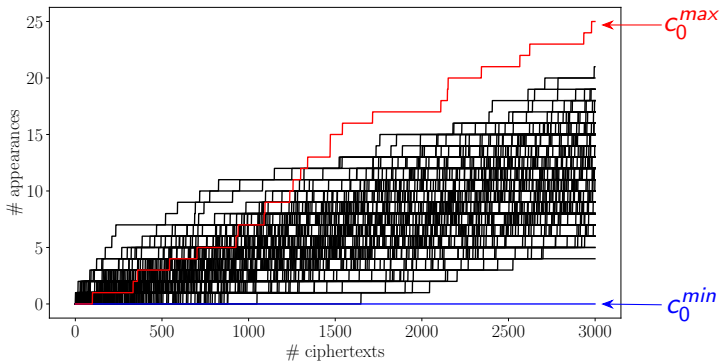
Without fault.



With one faulty byte.

<sup>2</sup>Fan Zhang et al. "Persistent Fault Analysis on Block Ciphers". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.3 (2018), pp. 150–172.

# Appearance of byte values



Number of appearance of byte values for byte 0.

- Firmware reverse engineering  
⇒ S-box stored between 0x080012F4 and 0x080013F3
- From the previous mapping  
⇒ the position  $(x, y) = (44.3, 300)$  is in the S-box range
- Laser fault injection  
⇒ fault 0x00000002 at the address 0x08001310  
32<sup>nd</sup> value of S-box faulted (0xC2 instead of 0xC0)

⇒ Successful PFA