Fault injection attacks

—

from practice to theory

SERICS Autumn School on Hardware Security

Brice Colombier

October 30, 2025

Laboratoire
Hubert Curien
UMR ▪ CNRS ▪ 5516 ▪ Saint-Étienne

Who am I?

Finished engineering school in 2014
PhD in microelectronics in 2017
Background in electrical engineering, embedded systems and digital design

Associate professor at Université Jean Monnet in Saint-Étienne, France

- Electrical engineering: digital design, C programming, electronics, ...
- SESAM[1] team in Laboratoire Hubert Curien

Laboratoire
Hubert Curien
UMR ▪ CNRS ▪ 5516 ▪ Saint-Étienne

`https://bcolombier.fr`

[1] Secure Embedded Systems and Hardware Architecture (***S****ystèmes* ***E****mbarqués* ***S****écurisés et* ***A****rchitectures* ***M****atérielles*)

Who are you?

1. What is the field you are the most familiar with?
   - Physics / Microelectronics
   - Embedded systems / Digital design
   - Computer science / Programming
   - Mathematics / Cryptography

Who are you?

1. What is the field you are the most familiar with?
   - Physics / Microelectronics
   - Embedded systems / Digital design
   - Computer science / Programming
   - Mathematics / Cryptography
2. What do you know about fault injection attacks?
   - Never heard of it before today
   - Heard a few things about it, but not more
   - Read a lot about it, but never practiced it myself
   - Practiced it myself already

Modus operandi

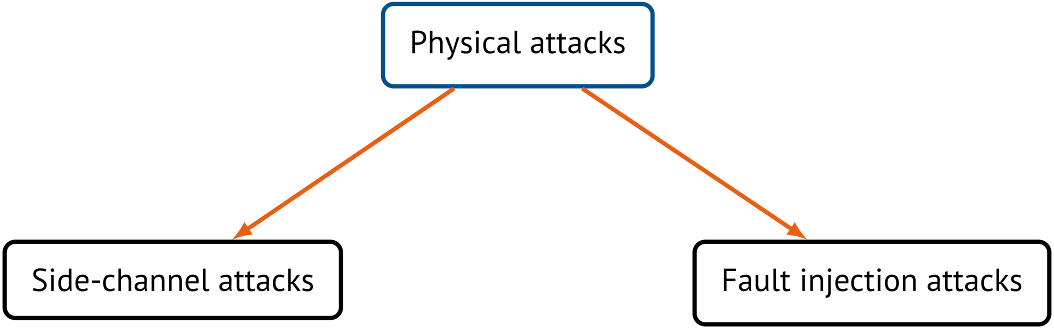
No practical session this morning...

→ But this afternoon yes!

Let's make this presentation interactive. Feel free to:

- 🗨 make comments
- ? ask questions

Definitions



Physical attacks

Side-channel attacks

Fault injection attacks

Definitions

Fault:
deviation from the normal operation of the device.

Definitions

Fault:
deviation from the normal operation of the device.

Fault injection:
deliberate attempt to deviate from the normal operation of the device.

Definitions

Fault:
deviation from the normal operation of the device.

Fault injection:
deliberate attempt to deviate from the normal operation of the device.

Fault attack:
exploitation of a given fault for malicious purposes in a given security context.

Definitions

Fault:
deviation from the normal operation of the device.

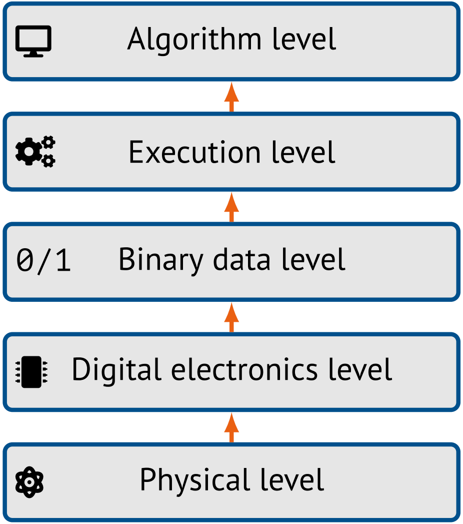
Fault injection:
deliberate attempt to deviate from the normal operation of the device.

Fault attack:
exploitation of a given fault for malicious purposes in a given security context.

Fault injection attack:
deliberate attempt to deviate from the normal operation of the device
and exploit it for malicious purposes.

Fault model

Fault model: self-contained description of the effect of the fault



Algorithm level	How the security algorithm is affected
Execution level	How program execution is affected
0/1 Binary data level	How data inside the device is affected
Digital electronics level	Flip-flops, logic gates, memory cells
Physical level	Microelectronics / transistors

Agenda

1. History
2. Techniques
3. Attacks
4. Countermeasures
5. Perspectives
6. Conclusion

Agenda

1. History
2. Techniques
3. Attacks
4. Countermeasures
5. Perspectives
6. Conclusion

Starting from reliability

Ever since electronic systems have been manufactured, we want them to be realiable.

Elsevier Microelectronics Reliability[2] has been in existence since 1962.
The IEEE Reliability Society celebrated its 75th anniversary this year.

Example article from 1968[3] by Texas Instrument:

1. General survey of integrated circuits failure mode
2. Reliability requirements
3. Reliability prediction
4. ...

[2] https://www.sciencedirect.com/journal/microelectronics-reliability

[3] W. Workman. “Failure Modes of Integrated Circuits and Their Relationship to Reliability”. In: Microelectronics Reliability (1968).

First published attacks

First publication[4] took ideas from pay-TV hackers (and others):

- alter the clock signal temporarily (5 MHz → 20 MHz)
- bridge a blown fuse with microprobe needles
- target set/reset EEPROM signals with microprobe needles to reprogram it
- constant data remanence in memory
- blocking some signals in protocols

[4] R. Anderson et al. “Low Cost Attacks on Tamper Resistant Devices”. In: Security Protocols. 1998.

Attackers taxonomy

Three classes of attackers[5] can be considered:

1. Clever outsiders
   - Insufficient knowledge of the system under attack
   - Moderately sophisticated equipment
   - Use existing weaknesses of the system
2. Knowledgeable insiders
   - Substantial specialized technical education and experience
   - Access to the full system description
   - Highly sophisticated equipment
3. Funded organisations
   - Teams of experts
   - Great funding
   - Sophisticated attack paths

[5] R. Anderson et al. “Low Cost Attacks on Tamper Resistant Devices”. In: Security Protocols. 1998.

Reliability meets cryptography

First publication by Boneh *et al.* at EUROCRYPT 1997[6] of the so-called Bellcore attack
Only theoretical: “the attack described in this paper is currently theoretical”.

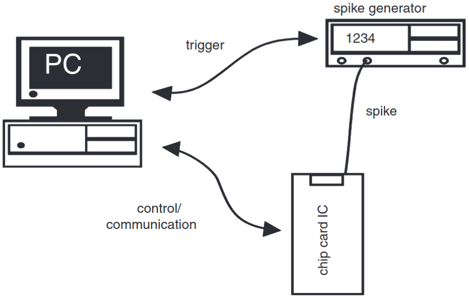
Faults considered:

- Causes:
    - *“some miraculous event”*
    - *“a miraculous fault”* x2
    - *“register fault”*: low-probability bit-flip in the processor’s registers
- Consequences:
    - Faulty decomposition in RSA-CRT
    - Faulty computation of a product in Fiat-Shamir identification scheme
    - Faulty computation in the Schnorr’s identification scheme

[6] D. Boneh et al. “On the Importance of Checking Cryptographic Protocols for Faults”. In: EUROCRYPT. 1997.

Reliability meets cryptography in practice

Attack put in practice (and published) three years later[7].



Experimental setup:

- target: a smart-card
- fault injection technique: voltage glitches
    - 90 % repeatability

Lots of interesting insights:

- Fault model (error scenario) discussion: data corruption, arithmetic errors,
- Perform side-channel analysis beforehand for synchronization,
- Software countermeasures are not sufficient: hardware ones are needed.

[7] C. Aumüller et al. “Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures”. In: CHES. 2002.

The IoT

Nowadays:

- 10^9 Billions of devices
- 📶 Connected to the Internet
- Handling sensitive data
- 🌍 Out there in the open
- $ Experimental equipment is accessible and getting cheaper by the day

The IoT

Nowadays:

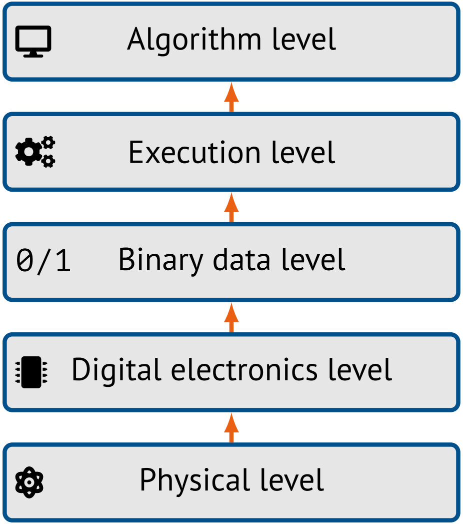
- 10^9 Billions of devices
- Connected to the Internet
- Handling sensitive data
- Out there in the open
- Experimental equipment is accessible and getting cheaper by the day

Fault injection attacks are more and more relevant

Agenda

1. History
2. Techniques
3. Attacks
4. Countermeasures
5. Perspectives
6. Conclusion

Techniques: restricted fault model



Algorithm level — …

Execution level — How program execution is affected

0/1 Binary data level — How data inside the device is affected

Digital electronics level — Flip-flops, logic gates, memory cells

Physical level — Microelectronics / transistors

This talk: microcontrollers only (FPGAs and ASICs are too specific).

Techniques

For every technique, we will examine:

- the attacker model
- its history
- the experimental setup
- the experimental parameters
- the fault model

Techniques:

Voltage glitch

Voltage glitch: Attacker model

Hypothesis: every electronic device is powered by an external power source:

- 🔋 Battery
- 🔌 Power supply

→ Dedicated connection on the board and input pin on the device.

Voltage glitch: Attacker model

Hypothesis: every electronic device is powered by an external power source:

- 🔋 Battery
- 🔌 Power supply

→ Dedicated connection on the board and input pin on the device.

An attacker can tamper with the power supply of the device and perform:

- ✗ Overpowering
- ✓ Underpowering

Only momentarily to inject the fault over a short period of time.

Voltage glitch: History

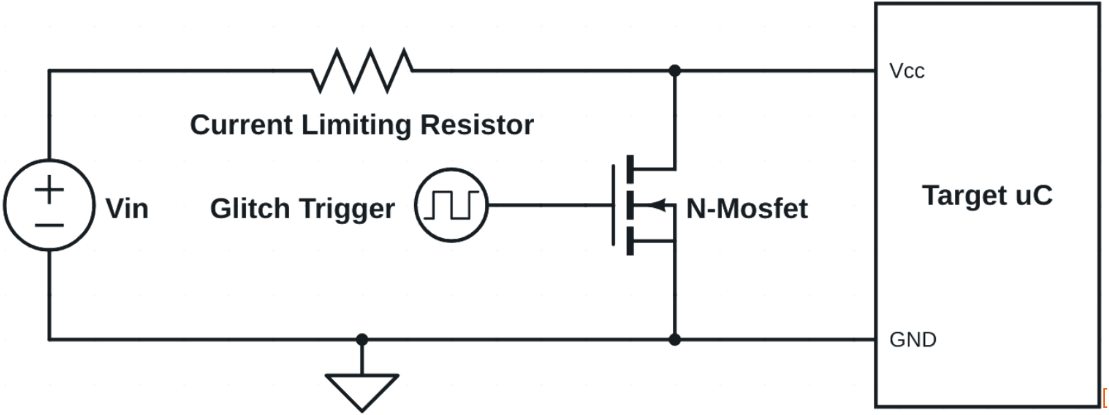
Introduced in 2008[8]:

- a 130 nm ASIC with a nominal voltage of 1.2 V.
- below 700 mV: I/O crashes
- single-bit errors around 800 mV

Used a controllable power supply with sub-millivolt accuracy.

[8] N. Selmane et al. “Practical Setup Time Violation Attacks on AES”. In: EDCC. 2008.

Voltage glitch: Experimental setup

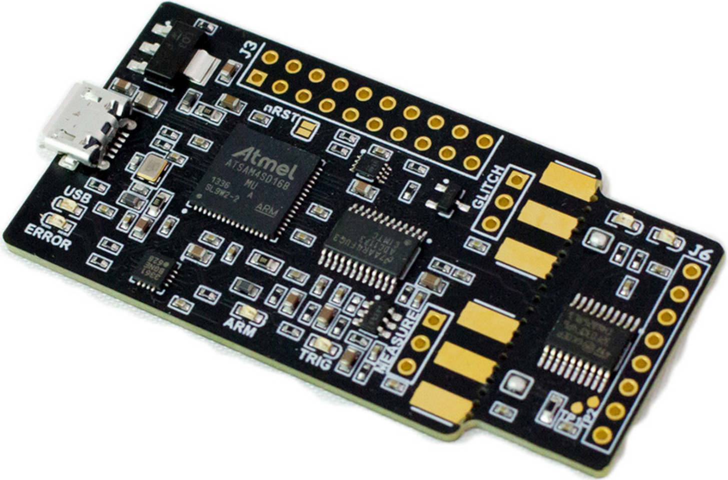


[9]

[9] C. Bozzato et al. “Shaping the Glitch: Optimizing Voltage Fault Injection Attacks”. In: TCHES (2019).

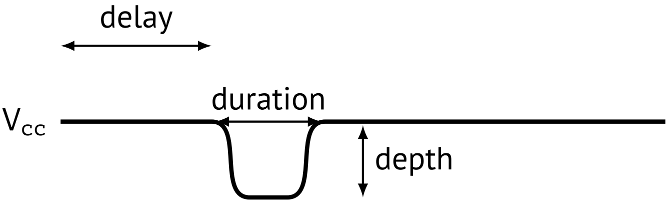
Voltage glitch: Experimental setup

ChipWhisperer Nano:



Voltage glitch: Parameters

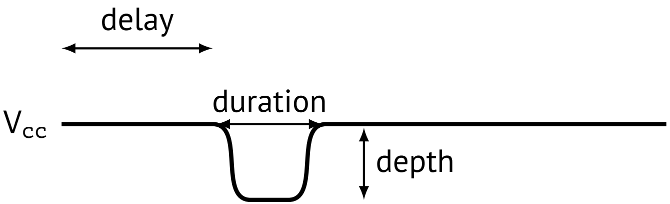
- delay
- duration (of the glitch)
- depth (w.r.t V_{cc})



[10] C. Bozzato et al. “Shaping the Glitch: Optimizing Voltage Fault Injection Attacks”. In: TCHES (2019).

Voltage glitch: Parameters

- delay
- duration (of the glitch)
- depth (w.r.t V_{cc})



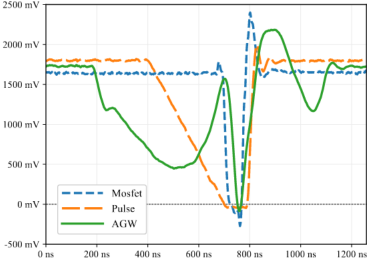
delay

duration

V_{cc}

depth

Further works have shown that the shape of the glitch matters too[10].



Hardware-dependent parameters:

- duration
- depth

Program-dependent parameters:

- delay

[10] C. Bozzato et al. “Shaping the Glitch: Optimizing Voltage Fault Injection Attacks”. In: TCHES (2019).

Voltage glitch: Fault model– physical level

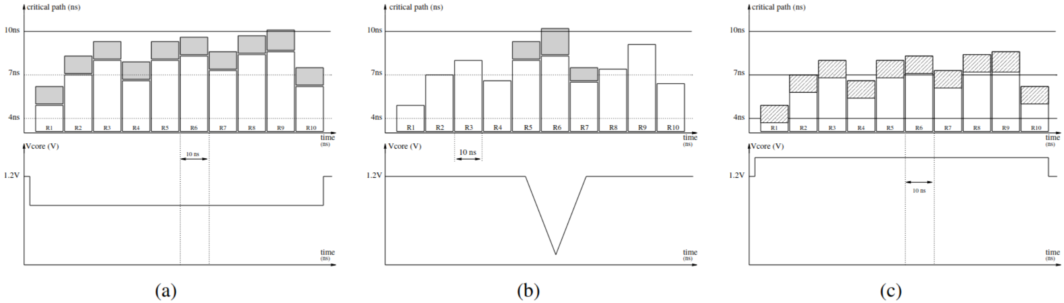
Physical level: A lower supply voltage has two effects:

- descreases switching speed of transistors
- increases the time needed to charge parasitic capacitances on wires

Both effect sum up to increase propagation times.

Voltage glitch: Fault model– digital level

Digital level: timing constraints may be violated



(a)

(b)

(c)

Fig. 2. Critical paths of the AES' rounds when subject to: (a) underpowering, (b) a negative power supply glitch, (c) overpowering.[11]

[11] L. Zussa et al. "Analysis of the Fault Injection Mechanism Related to Negative and Positive Power Supply Glitches Using an On-Chip Voltmeter". In: HOST. 2014.

Voltage glitch: Fault model– binary level

Binary level: binary data is not / partially updated:

- Previous data may be kept in the registers:
  - fully
  - partially
- Because of precharge logic, data in the registers may be fully / partially:
  - set
  - reset
  - Consistent critical bits over multiple instructions
- Because of microarchitecural features, some extra data may still be present
  - Skip with forwarding[12]

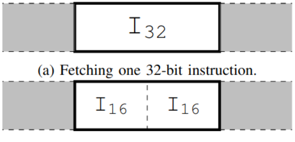
[12] I. Alshaer et al. “Inferred Fault Models for RISC-V and Arm: A Comparative Study”. In: DFT. 2024.

Voltage glitch: Fault model– execution level

Difference between the data width when fetched and the instruction length[13]

Several fault models:

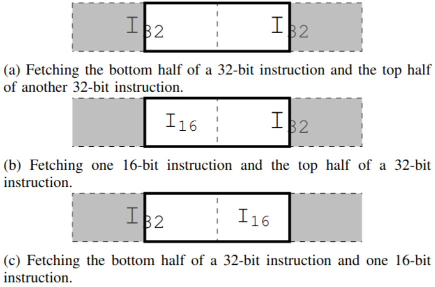
- Single instruction skip (2.a)
- Double instruction skip (2.b)
- Double instruction corruption (3.a)
- New instruction execution (3.b-c)



(a) Fetching one 32-bit instruction.

(b) Fetching two 16-bit instructions.

Fig. 2: Fetching aligned instructions.



(a) Fetching the bottom half of a 32-bit instruction and the top half of another 32-bit instruction.

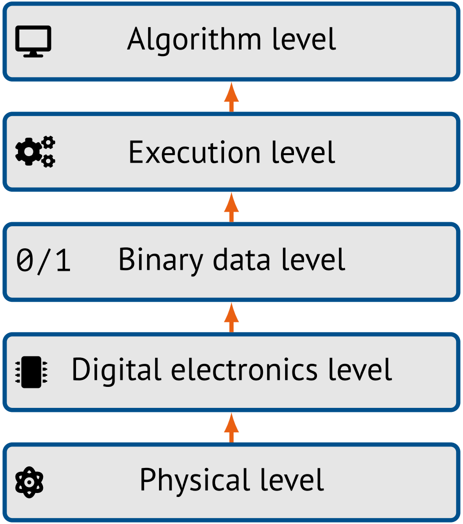
(b) Fetching one 16-bit instruction and the top half of a 32-bit instruction.

(c) Fetching the bottom half of a 32-bit instruction and one 16-bit instruction.

Fig. 3: Fetching misaligned instructions.

[13] I. Alshaer et al. “Variable-Length Instruction Set: Feature or Bug?” In: DSD. 2022.

Voltage glitch: Fault model



Algorithm level	
Execution level	Instr. & data skip / repeat / corruption / craft
0/1 Binary data level	Data is not/partially updated
Digital electronics level	Timing constraints violation
Physical level	Increases propagation times

Voltage glitch: Remarks

In practice, it might not be so easy:

- Internal voltage filtering / regulation
- Power management systems

Techniques:
Clock glitch

Clock glitch: Attacker model

Hypothesis: every electronic chip is clocked by an external clock signal:

- quartz
- PLL

→ Dedicated input pin / component on the board and input pin on the device.

Clock glitch: Attacker model

Hypothesis: every electronic chip is clocked by an external clock signal:

- quartz
- PLL

→ Dedicated input pin / component on the board and input pin on the device.

An attacker can tamper with the clock of the device and perform:

- ✗ Clock cycles skip
- ✓ Clock cycles shortening (a.k.a clock glitches)

Only momentarily to inject the fault over a short period of time.

Clock glitch: History

Overclocking is not usable since it is not precise.

Solution[14][15]: target a single clock cycle.

- Delay the main clock by a percentage of the period.
- XOR it with the main clock and a trigger signal.

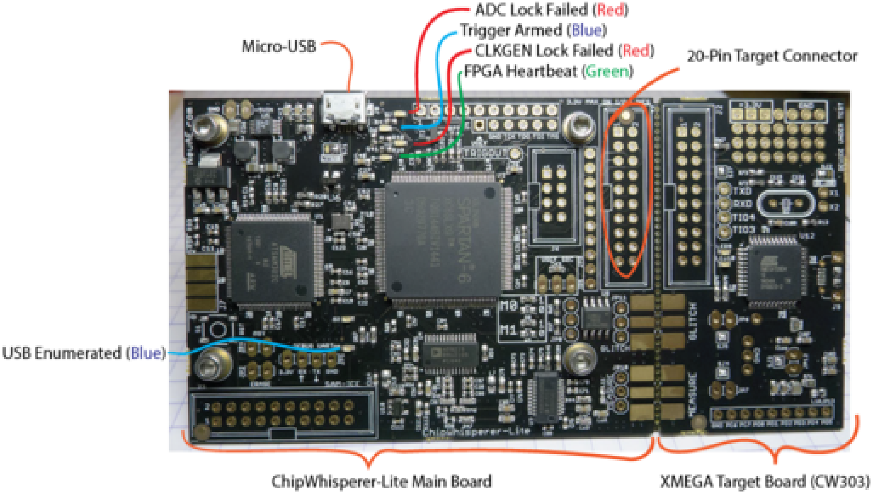
“Easily” feasible with a standard delay-locked loop, found in most FPGAs.

[14] R. Anderson et al. “Low Cost Attacks on Tamper Resistant Devices”. In: Security Protocols. 1998.

[15] M. Agoyan et al. “When Clocks Fail: On Critical Paths and Clock Faults”. In: CARDIS. 2010.

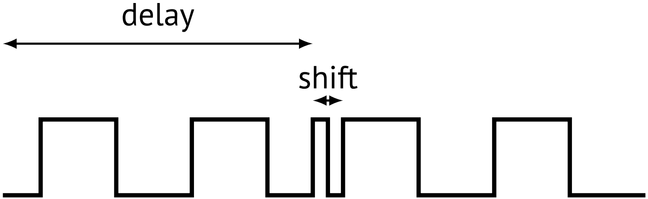
Clock glitch: Experimental setup

An advanced function generator or...
ChipWhisperer Lite:



Clock glitch: Parameters

- delay
- duration (of the glitch)
- shift (w.r.t the closest clock rising edge)



Hardware-dependent parameters:

- shift
- (duration)

Program-dependent parameters:

- delay

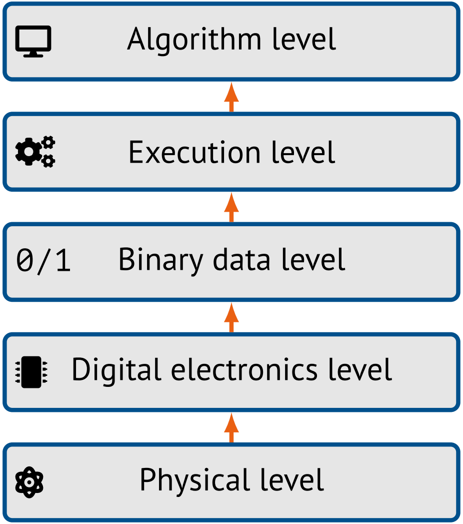
Clock glitch: Fault model

Physical level: N/A

Digital level: timing constraints may be violated

Binary & execution level: same as for voltage glitches

Clock glitch: Fault model



Algorithm level

Execution level

0/1 Binary data level

Digital electronics level

Physical level

Instr. & data skip / repeat / corruption / craft

Data is not/partially updated

Timing constraints violation

N/A

Clock glitch: Remarks

In practice, it might not be so easy:

- Clock management systems (PLLs, DLLs, frequency scaling)
- Multiple clock domains

Techniques:
Electromagnetic

Electromagnetic: Attacker model

An attacker can bring an injection probe sufficiently close to the chip package.

Much weaker than the previously considered ones:

- Voltage glitches: modify the power supply
- Clock glitches: modify the clock signal

→ What happens if it is processed again inside?

Electromagnetic: History

In 2002[16]: induce eddy currents inside the target chip by:

- wound a wire around a needle and connect it to the contacts of a camera flash gun without a bulb.

In 2007[17]: create sparks near the chip

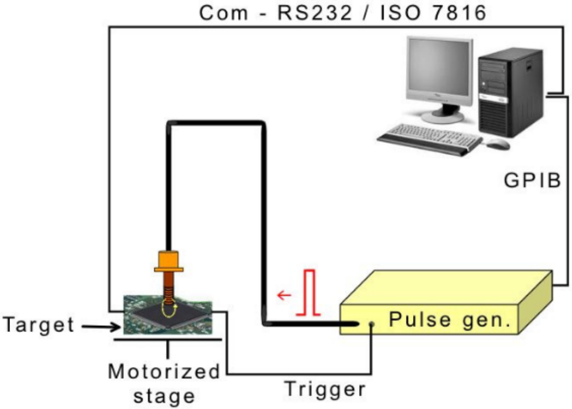
[16] J.-J. Quisquater et al. “Eddy Current for Magnetic Analysis with Active Sensor”. In: Proceedings of eSMART. 2002.

[17] J.-M. Schmidt et al. “Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results”. In: Austrian Workshop on Microelectronics. 2007.

Electromagnetic: History

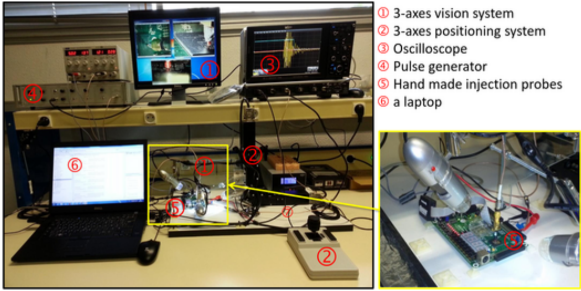
Problem: both methods suffer from very large jitter.

Fixed in 2012 by using a controllable pulse generator[18]

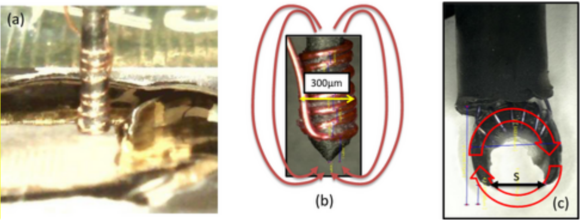


[18] A. Dehbaoui et al. “Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES”. In: FDTC. 2012.

Electromagnetic: Experimental setup– DIY

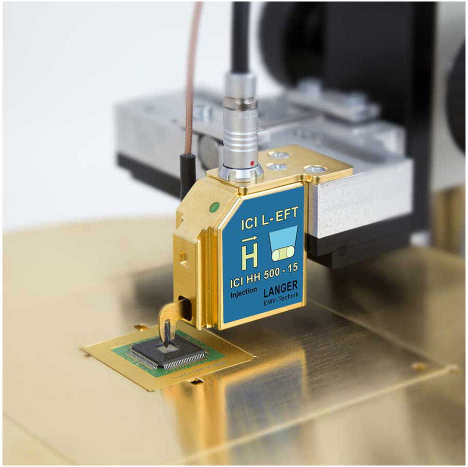


① 3-axes vision system
② 3-axes positioning system
③ Oscilloscope
④ Pulse generator
⑤ Hand made injection probes
⑥ a laptop



Electromagnetic: Experimental setup– Commercial

Commercial setups exist too (e.g. by Langer[19] or NewAE[20])



[19] https://www.langer-emv.de/en/category/fault-injection/116

[20] https://www.newae.com/chipshouter

Electromagnetic: Parameters

Hardware-dependent parameters:

- Probe position (x, y, z)
- Probe design (geometry, number of turns, etc)
- Rise/fall times
- Pulse width
- Pulse amplitude

Software-dependent parameters:

- delay

Electromagnetic: Fault model– physical level

Physical level: several phenomenon occur

- local voltage drop[21]
- voltage drop on the clock distribution network[22][23]
- alteration of the sampling capability of flip-flops[24][25]

[21] S. Ordas et al. “Evidence of a Larger EM-Induced Fault Model”. In: CARDIS. 2015.

[22] M. Ghodrati et al. “Inducing Local Timing Fault through EM Injection”. In: DAC. 2018.

[23] R. Nabhan et al. “A Tale of Two Models: Discussing the Timing and Sampling EM Fault Injection Models”. In: FDTC. 2023.

[24] S. Ordas et al. “Electromagnetic Fault Injection: The Curse of Flip-Flops”. In: Journal of Cryptographic Engineering (2017).

[25] S. Ordas et al. “Evidence of a Larger EM-Induced Fault Model”. In: CARDIS. 2015.

Electromagnetic: Fault model– digital level

Digital level: several fault models coexist

- timing faults: timing constraints are violated on the critical path by
- reset signal assertion on D flip-flops[26]
- sampling faults: around the clock edge, error in a sampling window[27]

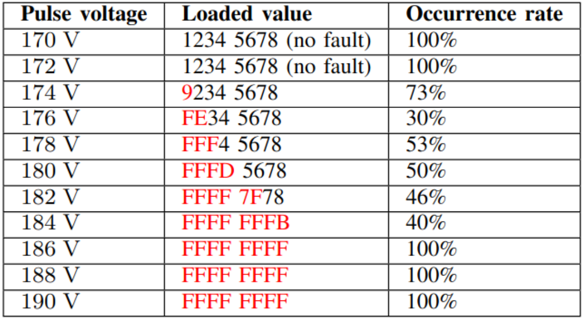
[26] S. Ordas et al. “Evidence of a Larger EM-Induced Fault Model”. In: CARDIS. 2015.

[27] S. Ordas et al. “Electromagnetic Fault Injection: The Curse of Flip-Flops”. In: Journal of Cryptographic Engineering (2017).

Electromagnetic: Fault model– binary & execution level

Progressive **Bit-set** on data and instruction fetched from Flash memory[28]:

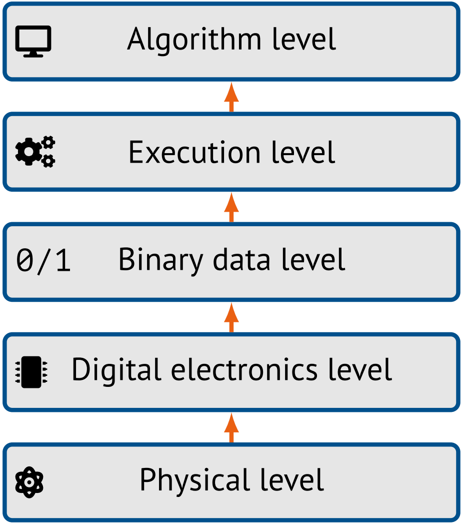
- Instruction corruption
- Data corruption



Pulse voltage	Loaded value	Occurrence rate
170 V	1234 5678 (no fault)	100%
172 V	1234 5678 (no fault)	100%
174 V	9234 5678	73%
176 V	FE34 5678	30%
178 V	FFF4 5678	53%
180 V	FFFD 5678	50%
182 V	FFFF 7F78	46%
184 V	FFFF FFFB	40%
186 V	FFFF FFFF	100%
188 V	FFFF FFFF	100%
190 V	FFFF FFFF	100%

[28] N. Moro et al. "Electromagnetic Fault Injection: Towards a Fault Model on a 32-Bit Microcontroller". In: FDTC. 2013.

Electromagnetic: Fault model



Algorithm level	
Execution level	Instruction / data corruption
0/1 Binary data level	Bit-set / reset
Digital electronics level	Timing constraints violation
Physical level	Voltage drops & DFF disturbance

Techniques:
Laser

Laser: History

Laser has been used (for a long time) to simulate the effect of radiations[29][30].

Laser fault injection was introduced by Skorobogatov in 2002[31]:
“We have carried them out using a flashgun bought second-hand from a camera store for $30 and with an $8 laser pointer”

[29] D. H. Habing. “The Use of Lasers to Simulate Radiation-Induced Transients in Semiconductor Devices and Circuits”. In: IEEE Transactions on Nuclear Science (1965).

[30] A. Johnston. “Charge Generation and Collection in P-n Junctions Excited with Pulsed Infrared Lasers”. In: IEEE Transactions on Nuclear Science (1993).

[31] S. P. Skorobogatov et al. “Optical Fault Induction Attacks”. In: CHES. 2002.

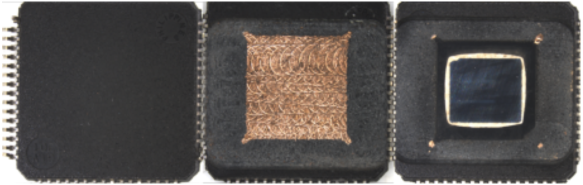
Laser: Attacker model

Hypothesis: every electronic device is made of silicon.

An attacker can access the backside of the device and shine a laser spot on it.

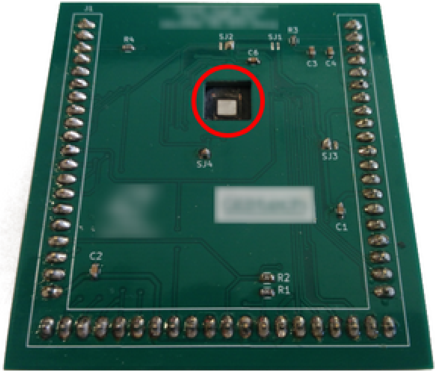
Sample preparation:

- mechanical polishing,
- chemical etching
  - hydrofluoric acid,
  - etc.



[32]

Custom board:



[32] R. S. Lima et al. "Target Preparation Methodology for Semi-Invasive Attacks on Microcontrollers". In: PAINE. 2022.

Laser: Experimental setup – laser source

Laser wavelength is related to the bandgap of the target material (1.12 eV for silicon)

An near-infrared laser source is needed:

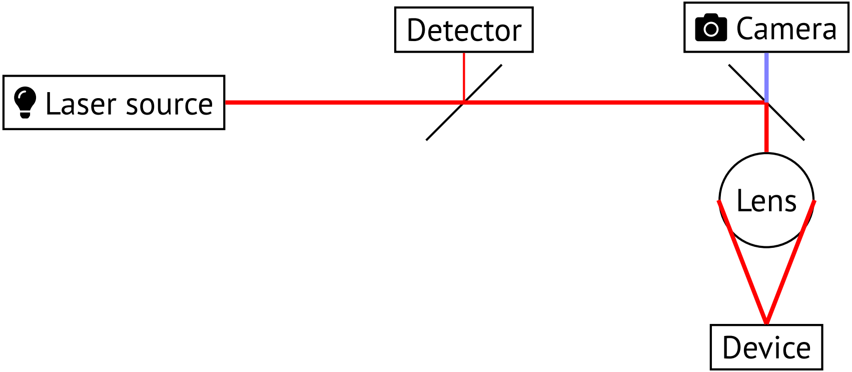
- 980 nm
- 1064 nm

Major drawback: we (humans) cannot see it:

- need an IR camera for positioning,

⚠ no palpebral reflex: 🦯

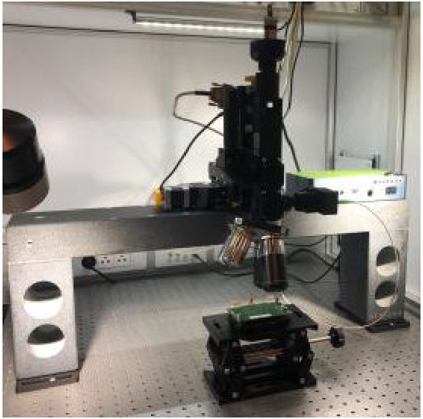
Laser: Experimental setup– optical path



- **Optical fibers** guide the laser,
- **Multiple objective lenses** are available,
  - different laser **spot sizes**,
  - different **absorption**.
- The **device** moves (XYZ).

Laser: Experimental setup– commercial setups

S-LMS by ALPhANOV[33]



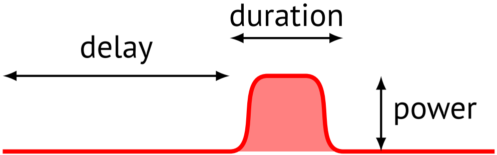
DS1101A by KEYSIGHT[34]

[34] https://www.alphanov.com/en/products-services/double-laser-fault-injection

[34] https://www.keysight.com/us/en/product/DS1101A/fault-injection-laser-system

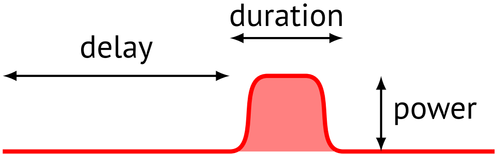
Laser: Parameters

- x position on the die
- y position on the die
- duration
- power
- delay



Laser: Parameters

- x position on the die
- y position on the die
- duration
- power
- delay



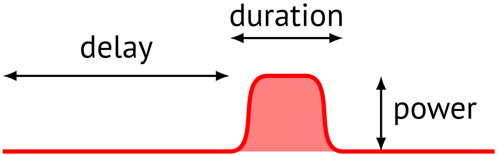
Example:

- 25 mm^2 chip (5 mm × 5 mm)
- laser spot size: 5 µm
- 1 value of power
- 100 values of delay

5×10^9 possibilities trials
Assume 10 trials per second

Laser: Parameters

- x position on the die
- y position on the die
- duration
- power
- delay



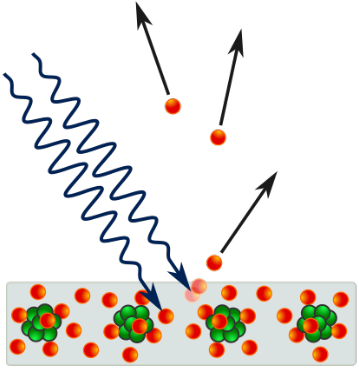
Example:

- 25 mm^2 chip (5 mm × 5 mm)
- laser spot size: 5 µm
- 1 value of power
- 100 values of delay

5 × 10^9 possibilities trials
Assume 10 trials per second
→ 4 months

Laser: Fault model– physical level

Physical level: photoelectric effect: photons are absorbed and electrons are emitted.


[35]

Only happens at a given wavelength related to the material bandgap:

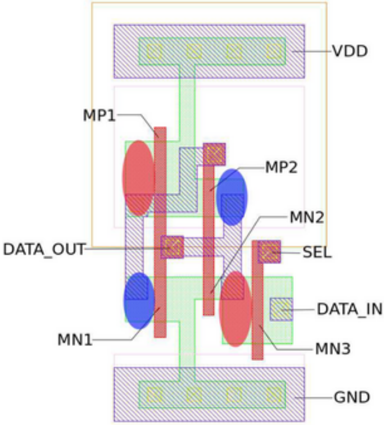
- 1.12 eV for silicon
- $\lambda \simeq 1100\,\text{nm}$

[35] Diagram by Ponor: `https://commons.wikimedia.org/wiki/File:Photoelectric_effect_in_a_solid_-_diagram.svg`

Laser: Fault model– digital level : SRAM cells

Digital level: an electric current is created by the electric field found in PN junctions.

SRAM cells can be set or reset: sensitive areas are the drains of the OFF transistors[36]

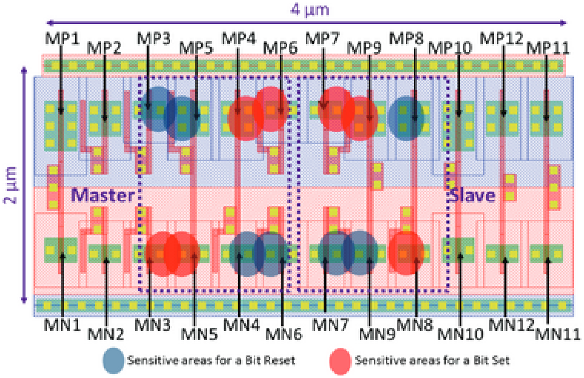


[36] C. Roscian et al. “Fault Model Analysis of Laser-Induced Faults in SRAM Memory Cells”. In: FDTC. 2013

Laser: Fault model– digital level : D flip-flops

Digital level: an electric current is created by the electric field found in PN junctions.

D flip-flops follow a similar pattern[37]



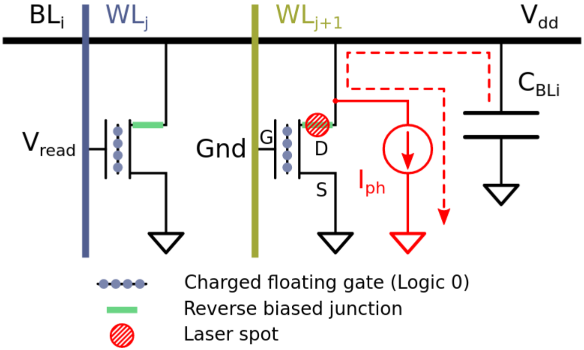
[37] C. Champeix et al. “SEU Sensitivity and Modeling Using Pico-Second Pulsed Laser Stimulation of a D Flip-Flop in 40 Nm CMOS Technology”. In: DFT. 2015

Laser: Fault model– digital level : Flash memory cells

Digital level: an electric current is created by the electric field found in PN junctions.

Flash memory cells can behave as if the floating was not charged[38]

The other way around, at write time, is feasible too[39]



[39] B. Colombier et al. “Laser-Induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-Bit Microcontroller”. In: HOST. 2019

[39] R. Viera et al. “Permanent Laser Fault Injection into the Flash Memory of a Microcontroller”. In: NEWCAS. 2021

Laser: Fault model– binary level

Binary level: data stored in D flip-flops and SRAM cells can be:

- set
- reset
- flipped

Data / instruction fetched from Flash memory can be:

- reset[40]
- set[41]

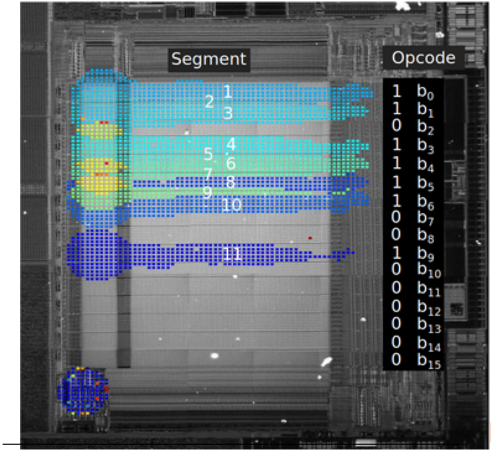
at a single-bit level, depending on the sense amplifier implementation.

[40] D. S. V. Kumar et al. "An In-Depth and Black-Box Characterization of the Effects of Laser Pulses on ATmega328P". In: CARDIS. 2019.

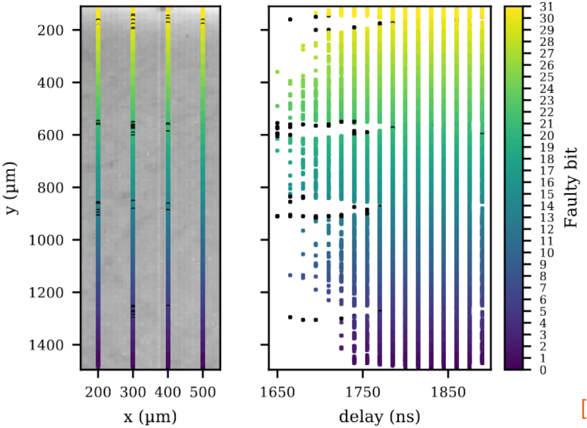
[41] B. Colombier et al. "Laser-Induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-Bit Microcontroller". In: HOST. 2019.

Laser: Fault model– execution level

Execution level: Single-bit set/reset on instruction or data from Flash



[42]

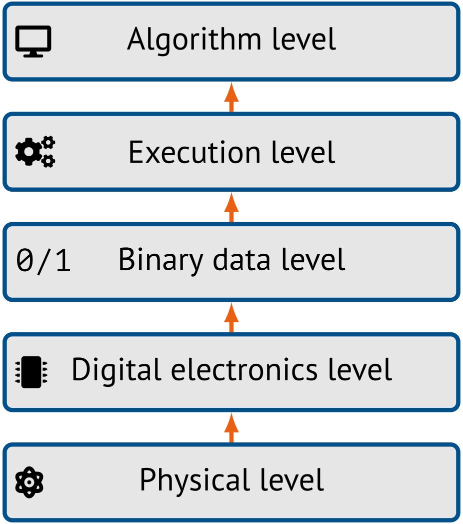


[43]

[43] D. S. V. Kumar et al. “An In-Depth and Black-Box Characterization of the Effects of Laser Pulses on ATmega328P”. In: CARDIS. 2019

[43] B. Colombier et al. “Laser-Induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-Bit Microcontroller”. In: HOST. 2019

Laser: Fault model



Algorithm level

Execution level

0/1 Binary data level

Digital electronics level

Physical level

Data / instruction corruption

Single-bit bit-set / reset

Transistors are forced to conduct / currents

Photoelectric effect: charges are created

Techniques: Summary

Technique	Type	Main fault model	Cost
Clock glitch	Semi-invasive	Instruction skip	**$**
Voltage glitch	Semi-invasive	Instruction skip	**$**
EM	Non-invasive	Instruction skip / data corruption	**$$**
Laser	Invasive	Single-bit bit set	**$$$**

Techniques: The trigger

Big Question: "When should the fault be injected"?

→ Inside the sensitive program!

[44] C. O'Flynn et al. "ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research". In: COSADE. 2014.

Techniques: The trigger

Big Question: “When should the fault be injected”?
→ Inside the sensitive program!

Big Question bis: “When does the sensitive program start (and stop)”?
→ Use a trigger signal 😐

[44] C. O’Flynn et al. “ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research”. In: COSADE. 2014.

Techniques: The trigger

Big Question: “When should the fault be injected”?
→ Inside the sensitive program!

Big Question bis: “When does the sensitive program start (and stop)”?
→ Use a trigger signal 😐

Trigger signal generation:

- Use a dedicated pin: 0 → 1 → 0 (strong assumption...)
- Match data on a communication bus (UART, USB analyzer, ...)
  - plaintext,
  - key,
  - start command
- Match a pattern (approximately) on the power consumption side-channel.
  - sum-of-absolute-differences module in ChipWhisperer[44]

[44] C. O’Flynn et al. “ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research”. In: COSADE. 2014.

Techniques: the ones that did not make it to this talk

😢

- Temperature:
    - *“a 50-watt spotlight bulb”*[45]
- Body-bias[46]
- X-rays[47]
- FIB (Focused Ion Beam)

[45] S. Govindavajhala et al. “Using Memory Errors to Attack a Virtual Machine”. In: SP. 2003.

[46] P. Maurine et al. “Yet Another Fault Injection Technique: By Forward Body Biasing Injection”. In: YACC. 2012.

[47] S. Anceau et al. “Nanofocused X-Ray Beam to Reprogram Secure Circuits”. In: CHES. 2017.

Agenda

1. History
2. Techniques
3. **Attacks**
4. Countermeasures
5. Perspectives
6. Conclusion

Attacks:
VerifyPIN

Attacks on VerifyPIN:

```
Input: user_PIN
for $i \in [0 \ldots 3]$ do
    if user_PIN[$i$] != ref_PIN[$i$] then
        return false
    end
end
return true
```

Attacks on VerifyPIN:

```
Input: user_PIN
for $i \in [0...3]$ do
    if user_PIN[$i$] != ref_PIN[$i$] then
        return false
    end
end
return true
```

Your turn

Attacks:
AES

Attacks on AES: Reminder

```
Input: $P$ (128-bit plaintext) and $K$ (128-bit secret key)
$K^1, \ldots, K^{10} \leftarrow \texttt{KeySchedule}(K)$;
$S \leftarrow P \oplus K$;
for $r \in [1 \ldots 10]$ do
    $S \leftarrow \texttt{SubBytes}(S)$;
    $S \leftarrow \texttt{ShiftRows}(S)$;
    if $r \neq 10$ then
        $S \leftarrow \texttt{MixColumns}(S)$;
    end
    $S \leftarrow S \oplus K^r$;
end
$C \leftarrow S$;
return $C$
```

S

S_0	S_4	S_8	S_{12}
S_1	S_5	S_9	S_{13}
S_2	S_6	S_{10}	S_{14}
S_3	S_7	S_{11}	S_{15}

Attacks on AES: Safe-error

Published in 2003[48], relies on an asymmetric fault model.

Without loss of generality, let’s consider a **bit-set** (0 → 1) fault model.

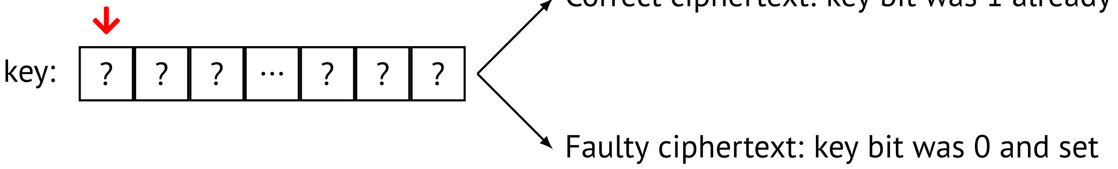
key:	?	?	?	⋯	?	?	?

[48] J. Blömer et al. “Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)”. In: Financial Cryptography. 2003.

Attacks on AES: Safe-error

Published in 2003[48], relies on an asymmetric fault model.

Without loss of generality, let’s consider a **bit-set** (0 → 1) fault model.



Correct ciphertext: key bit was 1 already

Faulty ciphertext: key bit was 0 and set

Key bits are recovered one-by-one: 128 faulty ciphertexts are needed for AES-128

[48] J. Blömer et al. “Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)”. In: Financial Cryptography. 2003.

Attacks on AES: Faulting the last AddRoundKey

```
Input: $S$ (the AES state after a round) and $K^r$ (128-bit round key)
for $col \in [0...3]$ do
    for $row \in [0...3]$ do
        $S_{4\times col+row} = S_{4\times col+row} \oplus K^r_{4\times col+row}$
    end
end
```

Bit-set on the loop counter increment constant (+1 → +5)[49] for early exit

$$\begin{array}{|c|c|c|c|}\hline C_0 & C_4 & C_8 & C_{12} \\ \hline C_1 & C_5 & C_9 & C_{13} \\ \hline C_2 & C_6 & C_{10} & C_{14} \\ \hline C_3 & C_7 & C_{11} & C_{15} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline C_0 & \tilde{C}_4 & \tilde{C}_8 & \tilde{C}_{12} \\ \hline \tilde{C}_1 & \tilde{C}_5 & \tilde{C}_9 & \tilde{C}_{13} \\ \hline \tilde{C}_2 & \tilde{C}_6 & \tilde{C}_{10} & \tilde{C}_{14} \\ \hline \tilde{C}_3 & \tilde{C}_7 & \tilde{C}_{11} & \tilde{C}_{15} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & K^{10}_4 & K^{10}_8 & K^{10}_{12} \\ \hline K^{10}_1 & K^{10}_5 & K^{10}_9 & K^{10}_{13} \\ \hline K^{10}_2 & K^{10}_6 & K^{10}_{10} & K^{10}_{14} \\ \hline K^{10}_3 & K^{10}_7 & K^{10}_{11} & K^{10}_{15} \\ \hline \end{array}$$

[49] B. Colombier et al. “Laser-Induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-Bit Microcontroller”. In: HOST. 2019.

Attacks on AES: Round-modification (1)

Perform only one round of AES[50]:

$$\tilde{C} = MC(SR(SB(P \oplus K))) \oplus K^1$$

Let two faulty ciphertexts (no correct ciphertext required):

$$\tilde{C}^a = MC(SR(SB(P^a \oplus K))) \oplus K^1 \qquad \tilde{C}^b = MC(SR(SB(P^b \oplus K))) \oplus K^1$$

By XORing them together:

$$\tilde{C}^a \oplus \tilde{C}^b = MC(SR(SB(P^a \oplus K))) \oplus MC(SR(SB(P^b \oplus K)))$$

$$MC^{-1}(\tilde{C}^a \oplus \tilde{C}^b) = SB(P^a \oplus K) \oplus SB(P^b \oplus K) \quad \text{for every key byte}$$

The last equation holds only for two key byte values: 2^{16} complexity for full key.

[50] M. Tunstall et al. “Round Reduction Using Faults”. In: FDTC. 2005.

Attacks on AES: Round-modification (2)

Skip the last full round (9) of AES[51]:

$$c = SR(SB[MC(SR(SB(S^8))) \oplus K^9]) \oplus K^{10} \qquad \tilde{C} = SR(SB(S^8)) \oplus K^{10}$$

Combining them:

$$SB^{-1}(SR^{-1}(c \oplus K^{10})) = MC(\tilde{C} \oplus K^{10}) \oplus K^9$$

Repeating it for another (correct, faulty) pair and XORing with the previous equation:

$$SB^{-1}(SR^{-1}(C^a \oplus K^{10})) \oplus SB^{-1}(SR^{-1}(C^b \oplus K^{10})) = MC(\tilde{C}^a \oplus \tilde{C}^b)$$

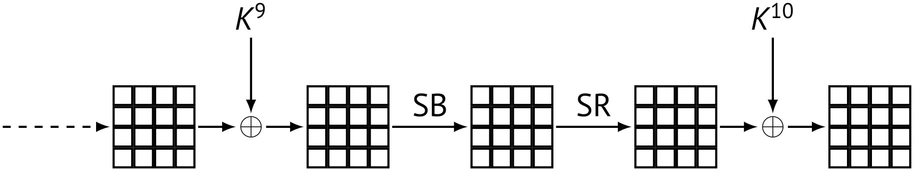
Similarly, this holds only for two key byte values: 2^{16} complexity for full key.

[51] J.-M. Dutertre et al. “Fault Round Modification Analysis of the Advanced Encryption Standard”. In: HOST. 2012.

Attacks on AES: Differential fault analysis

Differential fault analysis: in 1997[52] on DES, adapted to AES in 2003/4[53][54].

Fault model: **single-bit bit-flip** at the end of the 9th round.

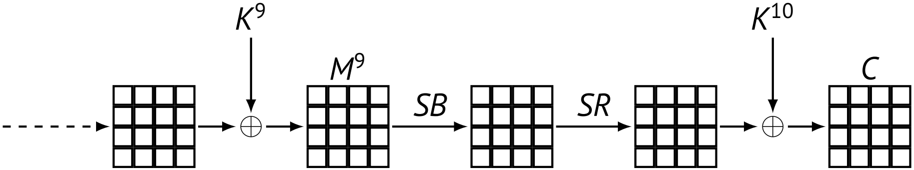


[52] E. Biham et al. “Differential Fault Analysis of Secret Key Cryptosystems”. In: Annual International Cryptology Conference. 1997.

[53] G. Piret et al. “A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad”. In: CHES. 2003.

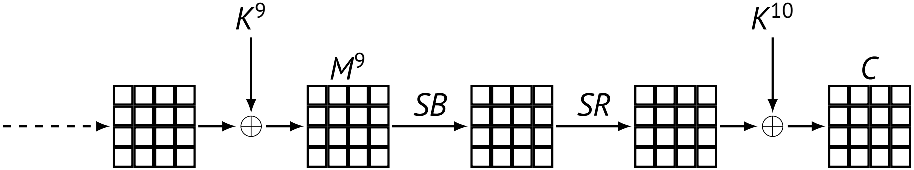
[54] C. Giraud. “DFA on AES”. In: International Conference on Advanced Encryption Standard. 2004.

Attacks on AES: Differential fault analysis

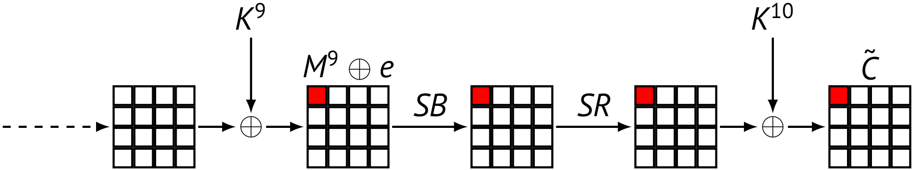


Ciphertext byte: $C_i = SR(SB(M_i^9)) \oplus K_i^{10}$

Attacks on AES: Differential fault analysis

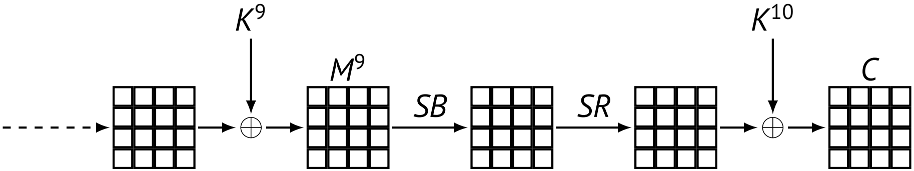


Ciphertext byte: $C_i = SR(SB(M_i^9)) \oplus K_i^{10}$

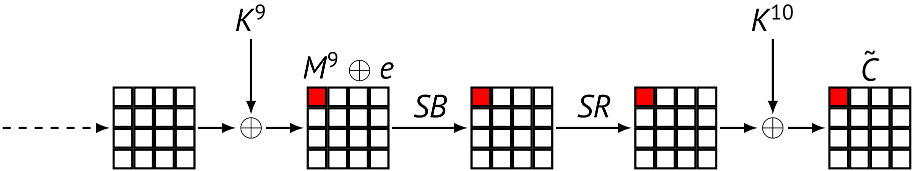


Faulty ciphertext byte: $\tilde{C}_i = SR(SB(M_i^9 \oplus e_i)) \oplus K_i^{10}$

Attacks on AES: Differential fault analysis



Ciphertext byte: $C_i = SR(SB(M_i^9)) \oplus K_i^{10}$



Faulty ciphertext byte: $\tilde{C}_i = SR(SB(M_i^9 \oplus e_i)) \oplus K_i^{10}$

$$C_i \oplus \tilde{C}_i = SB(M_i^9) \oplus SB(M_i^9 \oplus e_i)$$

Attacks on AES: Differential fault analysis

$$C_i \oplus \tilde{C}_i = SB(M_i^9) \oplus SB(M_i^9 \oplus e_i) \tag{1}$$

```
repeat
    for $e_i \in [\texttt{0x01}, \texttt{0x02}, \texttt{0x04}, \texttt{0x08}, \texttt{0x10}, \texttt{0x20}, \texttt{0x40}, \texttt{0x80}]$ do
        for $M_i^9 \in [0...255]$ do
            if Eq. (1) holds then
                Increment the score of $M_i^9$
            end
        end
    end
until an $M_i^9$ value has a sufficiently high score;
return the $M_i^9$ value with the highest score
```

Attacks on AES: Differential fault analysis

Next steps:

- Recover all 16 bytes $M^9_{i \in [0,15]}$ (take `ShiftRow` into account for other rows)
  - Possibly simultaneously (independence)
- Use $C_i = SR(SB(M^9_i)) \oplus K^{10}_i$ to recover K^{10}_i
- Reverse the AES key schedule to recover the secret key K.

Success rate:

- After 1 fault: the number of possible values for M^9_i drops from 256 to 14 at most
- After 2 faults: 50 % chances to get a single M^9_i
- After 3 faults: 97 % chances to get a single M^9_i

Attacks on AES: Differential fault analysis

Attack feasibility:

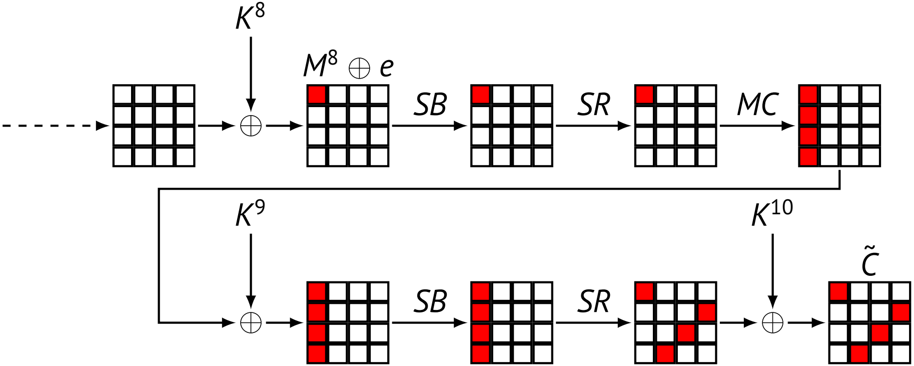
- ⊟ Encrypt the same plaintext n times, one correct and $n-1$ faulty
- ⊟ Single-bit bit-flip fault
- ⊞ Random position in the byte
- ⊞ Successful injection is easy to detect
- ⊞ Time is on your side
  - *“using a microscope, a modified camera flash and a computer”*[55]
  - continuous underpowering[56]

[55] C. Giraud. “DFA on AES”. In: International Conference on Advanced Encryption Standard. 2004.

[56] N. Selmane et al. “Practical Setup Time Violation Attacks on AES”. In: EDCC. 2008.

Attacks on AES: Differential fault analysis

Improvement: rool-back further (end of 8th round) to attack 4 bytes at a time[57].



Can go even further[58] (beginning of 8th round) at the cost of more hypotheses (2^{32})

[57] C. Giraud. "DFA on AES". In: International Conference on Advanced Encryption Standard. 2004.

[58] M. Tunstall et al. "Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault". In: WISTP. 2011.

Attacks on AES: Fault sensitivity analysis

Fault sensitivity analysis[59] requires no faulty ciphertexts, only faulty behaviours.

Hypotheses:

- device behaviour is data-dependent
    - delay, etc.
- therefore, fault sensitivity is data-dependent
    - Hamming weight sensitivity

Idea: perform a correlation power analysis on the fault sensitivity information.

[59] Y. Li et al. “Fault Sensitivity Analysis”. In: CHES. 2010.

Attacks on AES: Statistical fault analysis

Statistical fault analysis[60] requires no correct ciphertexts, only faulty ones.

Hypotheses:

- the fault model is biased:
  - “stuck at” fault model

Idea: make hypotheses on the key byte and compute a distinguisher for the faulty intermediate value:

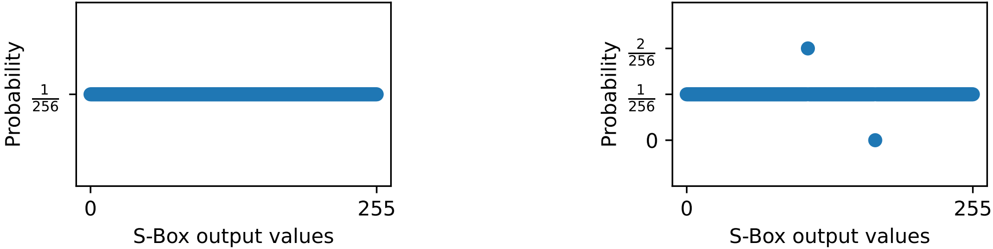
- maximum likelihood (assuming the fault distribution is perfectly known)
- mean Hamming weight bias
- distance to the uniform distribution

[60] T. Fuhr et al. “Fault Attacks on AES with Faulty Ciphertexts Only”. In: FDTC. 2013.

Attacks on AES: Persistent fault analysis

Persistent fault: persists until the next reboot[61]

Fault injection in the S-Box to make it slightly surjective → bias



One S-Box output is never present (SB_{min})

1. Record ciphertexts and observe which ciphertext byte is never present (c_{min})
2. Recover the key byte: $k_i = SB_{min} \oplus c_{min}$ after 1.5k trials approximately

[61] F. Zhang et al. "Persistent Fault Analysis on Block Ciphers". In: TCHES (2018).

Attacks on AES: Persistent fault analysis in practice

In practice, a permanent fault can be injected by multiple means:

- Faulting data in RAM
  - Rowhammer[62]
  - laser[63]
- Removing charges from the floating gate in Flash memory cells:
  - laser: local heating[64]
  - X-rays: charges creation → V_{th} shift[65]

[62] F. Zhang et al. “Persistent Fault Analysis on Block Ciphers”. In: TCHES (2018).

[63] F. Zhang et al. “Persistent Fault Attack in Practice”. In: TCHES (2020).

[64] P. Grandamme et al. “Switching Off Your Device Does Not Protect Against Fault Attacks”. In: TCHES (2024).

[65] P. Grandamme et al. “X-Ray Fault Injection in Non-Volatile Memories on Power OFF Devices”. In: PAINE. 2023.

Attacks:

Post-quantum cryptography

Attacks on PQC: Fujisaki-Okamoto transform

The Fujisaki-Okamoto transform[66] is used to prevent chosen-ciphertext attacks.

After decryption, encrypt again and check for a match.

Widely used, widely attackable by skipping the equality test[67].

[66] E. Fujisaki et al. “Secure Integration of Asymmetric and Symmetric Encryption Schemes”. In: Journal of Cryptology (2013).

[67] K. Xagawa et al. “Fault-Injection Attacks Against NIST’s Post-Quantum Cryptography Round 3 KEM Candidates”. In: ASIACRYPT. 2021.

Attacks on PQC: Classic McEliece

Classic McEliece[68] is a **Key Encapsulation Mechanism**

- KeyGen() $\rightarrow$ ($\mathbf{H}_{\text{pub}}$, k_{priv})
- Encaps($\mathbf{H}_{\text{pub}}$) $\rightarrow$ ($\mathbf{s}$, k_{session})
- Decaps($\mathbf{s}$, k_{priv}) $\rightarrow$ (k_{session})

Encaps (Niederreiter encryption[69]) encapsulates a secret value to be shared.

- Encaps($\mathbf{H}_{\text{pub}}$) $\rightarrow$ ($\mathbf{s}$, k_{session})

  Generate a random vector $\mathbf{e} \in \mathbb{F}_2^{\mathbf{n}}$ of Hamming weight $\mathbf{t}$ (($\mathbf{n}$; $\mathbf{t}$): security parameters)

  Compute $\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$

  Compute the hash: $k_{\text{session}} = H(1, \mathbf{e}, \mathbf{s})$

[68] M. R. Albrecht et al. Classic McEliece: Conservative Code-Based Cryptography. 2022.

[69] H. Niederreiter. “Knapsack-Type Cryptosystems and Algebraic Coding Theory”. In: Problems of Control and Information Theory (1986).

Attacks on PQC: Classic McEliece

Classic McEliece[68] is a **Key Encapsulation Mechanism**

- KeyGen() $\rightarrow (\mathbf{H}_{\text{pub}}, k_{\text{priv}})$
- Encaps($\mathbf{H}_{\text{pub}}$) $\rightarrow (\mathbf{s}, k_{\text{session}})$
- Decaps($\mathbf{s}, k_{\text{priv}}$) $\rightarrow (k_{\text{session}})$

Encaps (Niederreiter encryption[69]) encapsulates a secret value to be shared.

- Encaps($\mathbf{H}_{\text{pub}}$) $\rightarrow (\mathbf{s}, k_{\text{session}})$

  Generate a random vector $\mathbf{e} \in \mathbb{F}_2^{\mathbf{n}}$ of Hamming weight $\mathbf{t}$ $((\mathbf{n}; \mathbf{t})$: security parameters)

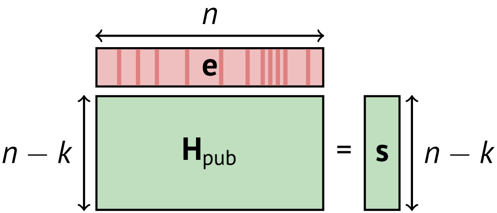
  Compute $\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$

  Compute the hash: $k_{\text{session}} = H(1, \mathbf{e}, \mathbf{s})$

[68] M. R. Albrecht et al. Classic McEliece: Conservative Code-Based Cryptography. 2022.

[69] H. Niederreiter. “Knapsack-Type Cryptosystems and Algebraic Coding Theory”. In: Problems of Control and Information Theory (1986).

Attacks on PQC: Classic McEliece – parameters



n	k	t	Security level
3488	2720	64	128
4608	3360	96	196
6688	5024	128	256
6960	5413	119	256
8192	6528	128	256

Attacks on PQC: Classic McEliece – Matrix-vector mult.

The $\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$ multiplication is performed over $\mathbb{F}_2$.

```
Input: H, e
s = [0, ..., 0];
for $r \in [0 \ldots n - k]$ do
    for $c \in [0 \ldots n - k]$ do
        s[r] ^= H[r][c] & e[c]
    end
end
return syn
```

Attacks on PQC: Classic McEliece – Matrix-vector mult.

The $\mathbf{s} = \mathbf{H}_{\text{pub}}\mathbf{e}$ multiplication is performed over $\mathbb{F}_2$.

```
Input: H, e
s = [0, ..., 0];
for $r \in [0...n - k]$ do
    for $c \in [0...n - k]$ do
        s[r] ^= H[r][c] & e[c]
    end
end
return syn
```

Attacks on PQC: Classic McEliece – Matrix-vector mult.

Targeting the XOR operation, considering the Thumb instruction set.

bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EORS: Rd = Rm $\oplus$ Rn	0	1	0	0	0	0	0	0	0	1	Rm			Rdn		
EORS: R1 = R0 $\oplus$ R1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Laser fault injection in flash memory : mono-bit, bit-set fault model[70][71].

ADCS: R1 = R0 + R1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1

[70] B. Colombier et al. “Laser-Induced Single-bit Faults in Flash Memory: Instructions Corruption on a 32-Bit Microcontroller”. In: HOST. 2019.

[71] A. Menu et al. “Single-Bit Laser Fault Model in NOR Flash Memories: Analysis and Exploitation”. In: FDTC. 2020.

Attacks on PQC: Classic McEliece – ILP

Consider $\mathbf{H}_{\text{pub}}\mathbf{e} = \mathbf{s}$ as an optimization problem and solve it.

Integer syndrome decoding problem ($\mathbb{N}$-SDP)

Input: a matrix $\mathbf{H}_{\text{pub}} \in \mathcal{M}_{n-k,n}(\mathbb{N})$ with $h_{i,j} \in \{0,1\}$ for all i, j
a vector $\mathbf{s} \in \mathbb{N}^{n-k}$ and a scalar $t \in \mathbb{N}^{+}$

Output: a vector $\mathbf{e}$ in $\mathbb{N}^{n}$ with $x_i \in \{0,1\}$ for all i
and with a Hamming weight HW($\mathbf{x}$) $\leq t$ such that: $\mathbf{H}_{\text{pub}}\mathbf{e} = \mathbf{s}$

ILP problem

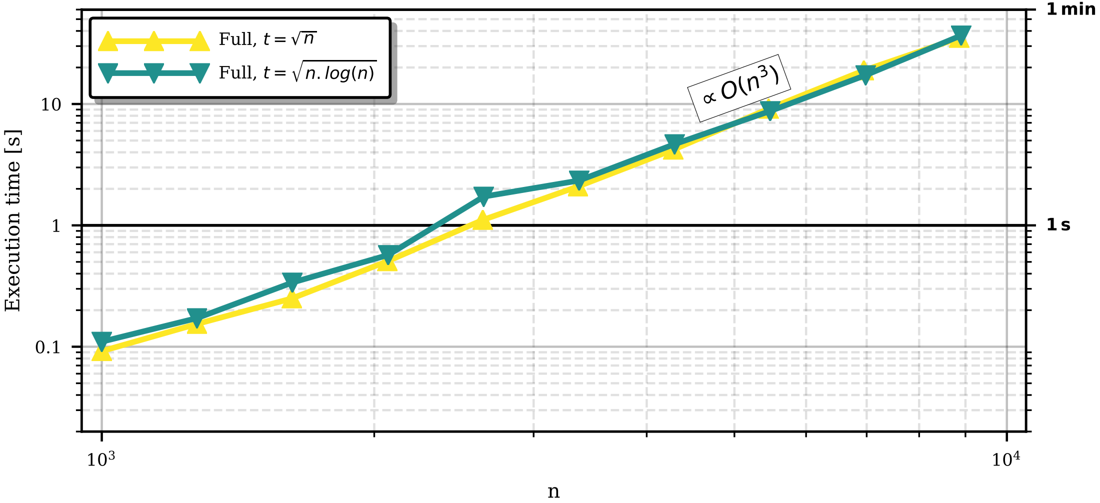
Let $\mathbf{b} \in \mathbb{N}^{n}$, $\mathbf{c} \in \mathbb{N}^{m}$ and $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{N})$:

$$\min\{\mathbf{b}^{T}\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{c}, \mathbf{x} \in \mathbb{N}^{n}, \mathbf{x} \geq 0\}$$

with $\mathbf{b} = (1, 1, ..., 1)$ and $\mathbf{x} \in \{0,1\}^{n}$

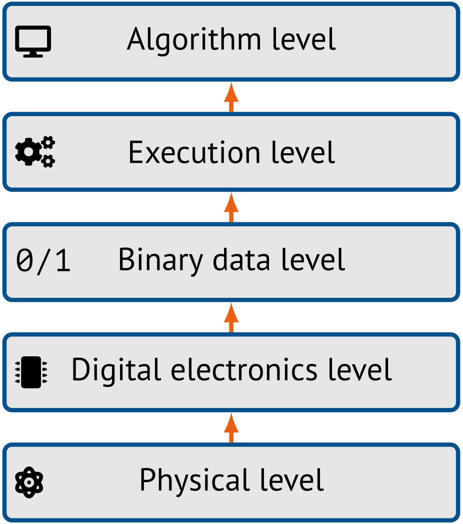
Solved by integer linear programming
(e.g. `Scipy.optimize.linprog`)

Attacks on PQC: Classic McEliece – ILP



For *Classic McEliece*: $3488 < n < 8192$

Attacks on PQC: Classic McEliece– summary



Algorithm level — Compute $\mathbf{H}_{\text{pub}}\mathbf{e}$ over $\mathbb{N}$ instead of $\mathbb{F}_2$

Execution level — Turn an XOR instruction into an ADD

0/1 Binary data level — Single-bit bit-set

Digital electronics level — Alter the readout of a flash memory cell

Physical level — Create charges by photoelectric effect

Attacks:
Secure boot

Attacks on Secure boot

Secure boot: verification of the authenticity of a boot image.

Authenticity: hash the image and compare with a reference.

Hash comparison is performed, and `auth=1` if they match[72]

Skipping the branching instruction allows to load a modified image.

Challenges:

- Complex hardware target
- Complex software target

[72] C. Fanjas et al. "Combined Fault Injection and Real-Time Side-Channel Analysis for Android Secure-Boot Bypassing". In: CARDIS. 2023.

Agenda

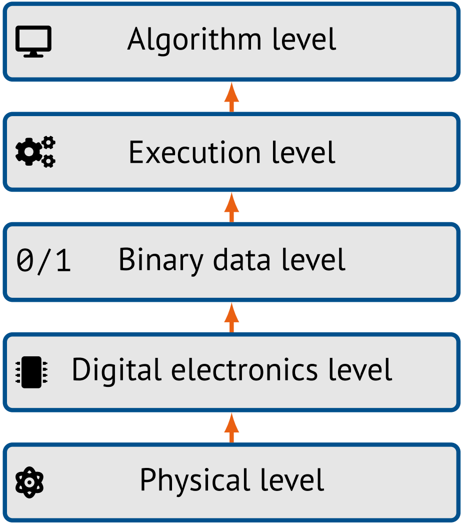
1. History
2. Techniques
3. Attacks
4. Countermeasures
5. Perspectives
6. Conclusion

Countermeasures

Countermeasures aim at preventing faults and/or attacks by:

- **neutralizing** the effect
  - → normal behaviour
- **spreading** the effect
  - → unexploitable behaviour
- **hiding** the effect
  - → no behaviour

A countermeasure has an effect at a given level of abstraction in the fault model.



Countermeasures coming from reliability

"many techniques of reliability have been ported as such to security applications. Nonetheless the objectives of reliability and security do differ"[73]

If an error/fault occurs:

Reliability : detect the error and act accordingly

- raise an alarm, fallback to emergency mode, etc.
- recover

Security : "the computation result, if erroneous, carries no information about secret involved"

- seems very restrictive
- does not need to preserve correct behaviour
- actually an alarm could be exploited (safe-error context)

[73] S. Guilley et al. "Fault Injection Resilience". In: FDTC. 2010.

Countermeasures: physical level

Physical-level countermeasures modify the integrated circuit or its package.

- Make the attack harder (integrity check, fault detection)
  - Add a metallic shield inside the device[74]
  - Add a metallic shield on top of the device[75]
- Detect the physical phenomenon
  - bulk current with a Bulk Built-In Current Sensor (BBICS)[76]
  - electromagnetic field with LC oscillators[77]
  - voltage drop[78]

[74] S. Briais et al. “Random Active Shield”. In: FDTC. 2012.

[75] C. Gaine et al. “Active Shielding Against Physical Attacks by Observation and Fault Injection: ChaXa”. In: JHSS (2023).

[76] R. P. Bastos et al. “A Bulk Built-in Sensor for Detection of Fault Attacks”. In: HOST. 2013.

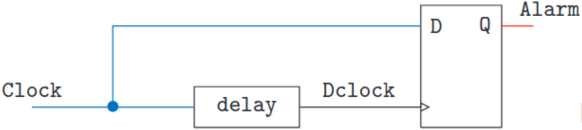
[77] N. Homma et al. “Design Methodology and Validity Verification for a Reactive Countermeasure Against EM Attacks”. In: Journal of Cryptology (2017).

[78] L. Zussa et al. “Analysis of the Fault Injection Mechanism Related to Negative and Positive Power Supply Glitches Using an On-Chip Voltmeter”. In: HOST. 2014.

Countermeasures: digital level

Digital-level countermeasures modify the way logic gates interact.

- Specific logic styles (dual-rail with precharge, etc)[79][80]
- Digital sensors:
  - Sampling faults aginst EM fault injection



[81]

  - IR drop against laser fault injection[82]

[79] K. Tiri et al. “A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation”. In: DATE. 2004.
[80] S. Guilley et al. “Fault Injection Resilience”. In: FDTC. 2010.
[81] D. El-Baze et al. “A Fully-Digital EM Pulse Detector”. In: DATE. 2016.
[82] M. Ebrahimabadi et al. “DELFINES: Detecting Laser Fault Injection Attacks via Digital Sensors”. In: TCAD (2024).

Countermeasures: binary level

Binary-level countermeasures make sure 0s and 1s are correct:

- Error-detection/correction codes
  - Parity bits[83]
  - codes[84]
- Hashes[85]

Often not enough, especially against instruction skip

[83] G. Bertoni et al. “Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard”. In: IEEE Transactions on Computers (2003).

[84] M. Karpovsky et al. “Robust Protection against Fault-Injection Attacks on Smart Cards Implementing the Advanced Encryption Standard”. In: DSN. 2004.

[85] J.-L. Danger et al. “CCFI-Cache: A Transparent and Flexible Hardware Protection for Code and Control-Flow Integrity”. In: DSD. 2018.

Countermeasures: execution level

Execution-level countermeasures checks that the program execution is correct.

Code and Control flow integrity: security properties

- Code integrity
- Code authenticity
- Control flow integrity
- Control signals integrity

Add metadata alongside the instructions[86][87]

Hardware + software (compiler) support is the key: many RISC-V based proposals.

Extra properties can be added: code confidentiality, data confidentiality, etc

[86] O. Savry et al. “Confidaent: Control FLow Protection with Instruction and Data Authenticated Encryption”. In: DSD. 2020.

[87] T. Chamelot et al. “MAFIA: Protecting the Microarchitecture of Embedded Systems Against Fault Injection Attacks”. In: TCAD (2023).

Countermeasures: application level

Application-level countermeasures involve dealing with the algorithm:

- Encrypt+decrypt to check for correctness (cf. FO transform)
- Make the ciphertext impossible to exploit (infective countermeasures)[88][89]

[88] B. Gierlichs et al. “Infective Computation and Dummy Rounds: Fault Protection for Block Ciphers without Check-before-Output”. In: LATINCRYPT. 2012.

[89] S. Patranabis et al. “Fault Tolerant Infective Countermeasure for AES”. In: SPACE. 2015.

Countermeasures: perspectives

Combined countermeasures against SCA and FIA:

- Merge masking and error detection[90]
- Combined attacks exist too!

Countermeasures against fault injection can help side-channel analysis[91]

Countermeasures can be attacked too:

- Do not assume that the countermeasure part is protected
- Use randomness to make analysis harder[92]

[90] T. Schneider et al. “ParTI – Towards Combined Hardware Countermeasures Against Side-Channel and Fault-Injection Attacks”. In: CRYPTO. 2016.

[91] L. Cojocar et al. “Instruction Duplication: Leaky and Not Too Fault-Tolerant!” In: CARDIS. 2018.

[92] V. Lomné et al. “On the Need of Randomness in Fault Attack Countermeasures - Application to AES”. In: FDTC. 2012.

Agenda

1. History
2. Techniques
3. Attacks
4. Countermeasures
5. Perspectives
6. Conclusion

Perspectives

Perspective #1 fault injection attacks are marginal

- Software attacks are still the vast majority
- Physical access to the device is very restrictive

However...

- Computing devices are more and more physically accessible
- More and more sensitive data is being handled by them
- Attack equipment can be expensive (but it is getting cheaper)

Perspectives

Perspective #2 no countermeasure is perfect:

- they all come at a cost
  - area / logic resources
  - execution time
- no silver bullet
- they can be attacked too[93]

[93] A. Askeland et al. “Who Watches the Watchers: Attacking Glitch Detection Circuits”. In: TCHES (2024).

Perspectives

Perspective #3 Other targets exist (besides crypto):

- neural networks[94][95]: misclassification, interesting fault models (memory effect)
- analog parts (RO[96], PLL[97], etc)
- random number generators[98][99]
- anything that is part of the security system / handling sensitive data

Domain knowledge is the key to efficient attacks.

[94] J. Breier et al. “Practical Fault Attack on Deep Neural Networks”. In: CCS. 2018.

[95] C. Gaine et al. “Fault Injection on Embedded Neural Networks: Impact of a Single Instruction Skip”. In: DSD. 2023.

[96] P. Bayon et al. “Contactless Electromagnetic Active Attack on Ring Oscillator Based True Random Number Generator”. In: COSADE. 2012.

[97] L. Dubois et al. “PLL Over-Clocking Through Repeated Fault Injections”. In: IOLTS. 2025.

[98] A. T. Markettos et al. “The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators”. In: CHES. 2009.

[99] M. Madau et al. “The Impact of Pulsed Electromagnetic Fault Injection on True Random Number Generators”. In: FDTC. 2018.

Perspectives

Perspective #4 The physical access requirement may not be relevant after all[100]:

- Rowhammer[101]
- Heat generators in FPGAs[102]
- Reliability/performance interfaces in complex systems
  - Delay lines calibration[103]
  - Processor frequency and voltage[104]

[100] A. M. Shuvo et al. A Comprehensive Survey on Non-Invasive Fault Injection Attacks. 2023. URL: https://eprint.iacr.org/2023/1769 (visited on 10/29/2025). Pre-published.

[101] Y. Kim et al. “Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors”. In: ISCA. 2014.

[102] M. Happe et al. “Eight Ways to Put Your FPGA on Fire – A Systematic Study of Heat Generators”. In: ReConFig. 2012.

[103] J. Gravellier et al. “FaultLine: Software-Based Fault Injection on Memory Transfers”. In: HOST. 2021.

[104] K. Murdock et al. “Plundervolt: Software-based Fault Injection Attacks against Intel SGX”. In: IEEE Symposium on Security and Privacy. 2020.

Perspectives

Perspective #5 The evaluation should be done as early as possible (pre-silicon)

- Formal verification to the rescue
- Large blocks can be efficiently checked:[105]
  - 3 faults on AES
  - bit-flip on the OpenTitan[106] secure element

[105] S. Tollec et al. "Fault-Resistant Partitioning of Secure CPUs for System Co-Verification against Faults". In: TCHES (2024).

[106] `https://opentitan.org/`

Agenda

1. History
2. Techniques
3. Attacks
4. Countermeasures
5. Perspectives
6. Conclusion

Conclusion

Conferences / journals / workshops to follow:

- FDTC (Workshop on Fault Detection and Tolerance in Cryptography)[107]
- TCHES (IACR Transactions on Cryptographic Hardware and Embedded Systems)[108]
- JAIF (Journée thématique sur les attaques par injection de fautes)[109]

[107] https://fdtc-workshop.eu/FDTC/

[108] https://tches.iacr.org/

[109] https://jaif.io/

Questions

– Questions ? –

How could that apply to your research topic?