CEA - DRT/DPACA
Secure Architectures and Systems laboratory

Brice Colombier
brice.colombier@cea.fr

Pierre-Alain Moëllic
pierre-alain.moellic@cea.fr

Security evaluation of countermeasures against physical attacks inserted at compilation time

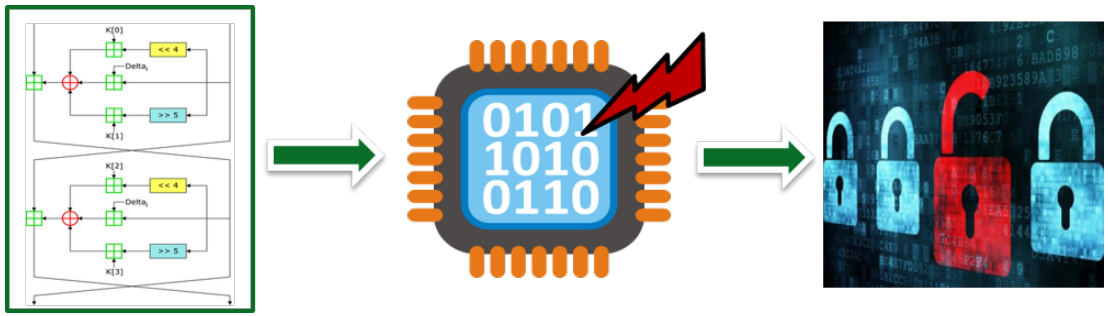
Cryptarchi 2018, Guidel-Plages

19 juin 2018

Context

Embedded systems are vulnerable to **physical attacks** aiming at:

- recovering secret data,
- bypassing protections (PIN, privileges, …),
- preparing/profiling bigger attacks



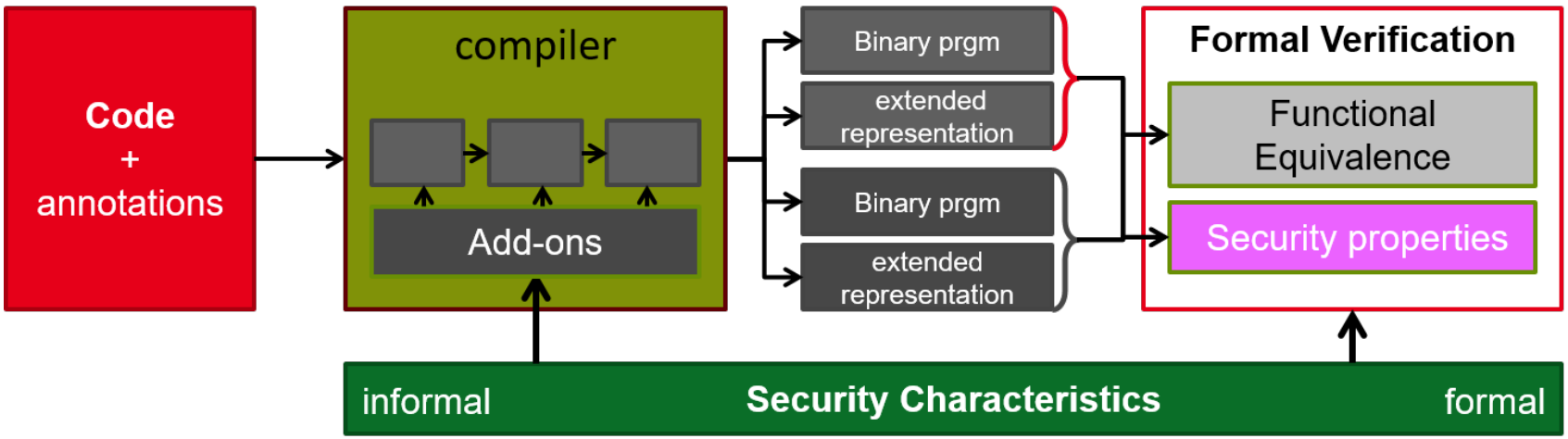
ANR-PROSECCO 2016-2019 [1] project with partners:

- CEA-Tech
- UPMC LIP6

[1] formally proven PROtections for Secured Compiled Code

PROSECCO approach

- Automatically apply the countermeasures against side-channel **and** fault attacks by acting during the **compilation** process:
  - LLVM pass added
  - Expected solutions: redundancy against skip, control flow integrity, masking, hiding…
- **Verification** of the protected code:
  - From a **functional** point of view (same **behaviour**)
  - From a **security** point of view (**evaluation**)



Software protections

Requirements for the protections:

- flexibility (possibility of updates and changes)
- portable to off-the-shelf hardware

Operation of the project

- Build a compiler that understands **annotated source code**: data and control flow
- Evaluate the impact and **robustness** of protections.

- **Formally prove** that the secure and normal codes are **functionally equivalent**.

Goal of the presentation

Goal: Present the first results of the security evaluation we perform at the Secure Architectures and Systems laboratory (joint team CEA Tech, Mines Saint-Etienne).

This evaluation helps to design **efficient countermeasures** by prodiving a **feedback to the designer**.

Evaluation carried out for different:

- Physical threats:
  - Side-channel analysis
  - Fault-attacks
- Hardware targets:
  - 8-bit microcontrollers
  - 32-bit microcontroller ARM Cortex M/A
- Practical use-cases:
  - VerifyPIN
  - AES encryption
  - (Secure boot)

Evaluation method

Two main axes:

- **Leakage** assessment using statistical tools
  - Attack-independent
- **Attack**-based methodology:

Complexity / Cost	Side-channel attacks	Fault attacks
+ / $	Correlation power analysis Template attacks	Clock glitches
+++ / $$$	Machine learning (deep neural networks)	Laser

Outline

1. Side-channel leakage assessment
2. Fault attacks on VerifyPIN
3. Combination of protections
4. Conclusion

Side-channel leakage assessment

Leakage assessment

Aim: conduct a statistical study to evaluate the leakages.

Statistical tests: reject or not a *null hypothesis* (i.e. the means of the target populations are equal)

Two common tools in SCA context:

- ***t*-test** [2]: split the traces in two sets w.r.t an intermediate value, see if they differ statistically.
  - The t statistic follows a Student law. For sufficient number of traces, $|t| > 4.5$ give a confidence of 99.999 % to reject the NH.
  - In our experiments: target at **bit level**.
- ***F*-test** [3], **SNR**: generalization of t-test for multiple sets. Takes the variance into consideration.
  - Ratio of inter-class VS intra-class variance.
  - In our experiments: target at **byte level**.

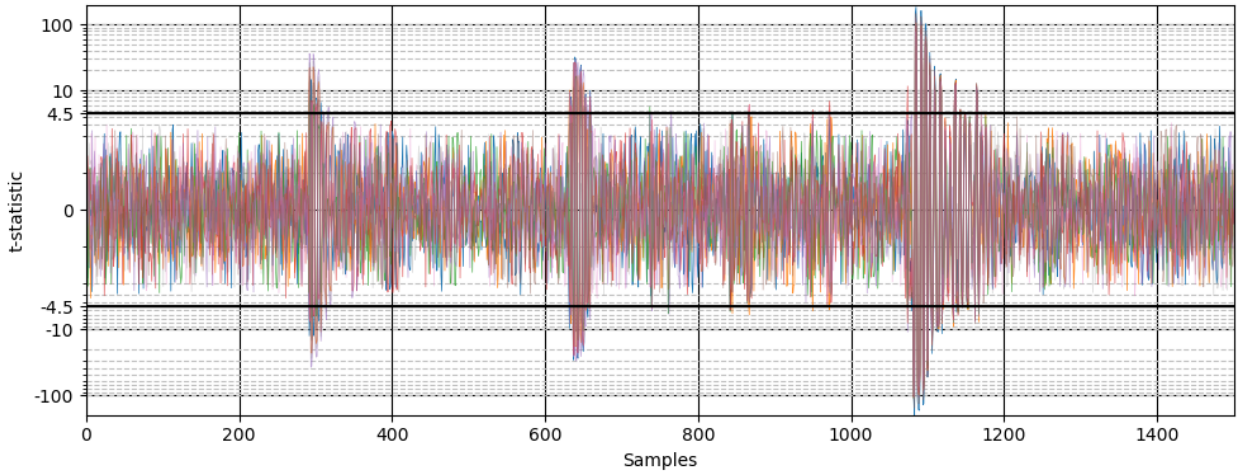
---

[2] Tobias Schneider and Amir Moradi. "Leakage Assessment Methodology - a clear roadmap for sidechannel evaluations". IACR ePrint 2015.

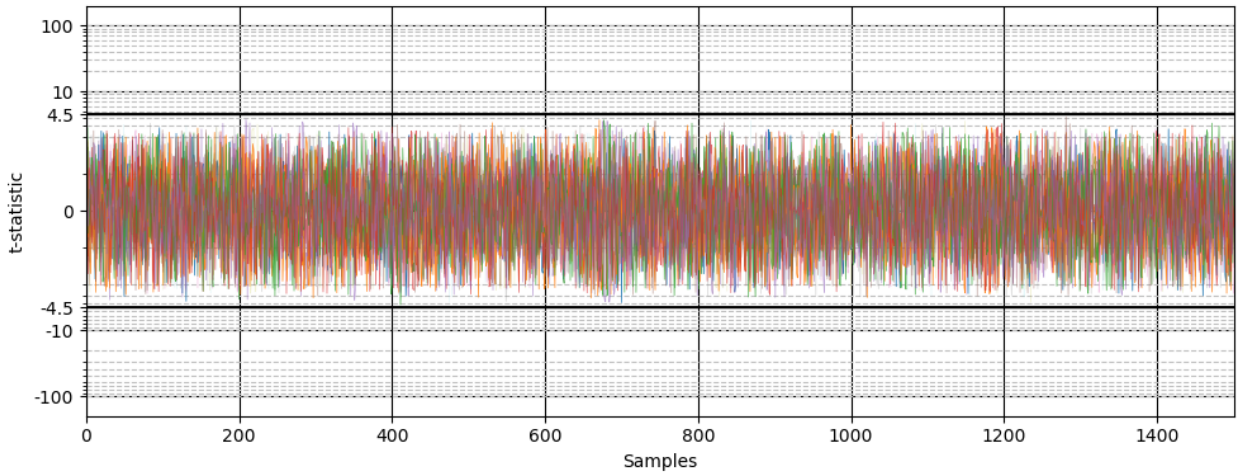
[3] Omar Choudary and Markus G. Kuhn. "Efficient template attacks." International Conference on Smart Card Research and Advanced Applications. 2013.

Comparison of unmasked and masked S-boxes

Splitting according to the value of the 8 bits at the 1st S-box output.
20000 traces of 128-bit AES encryption.



unmasked

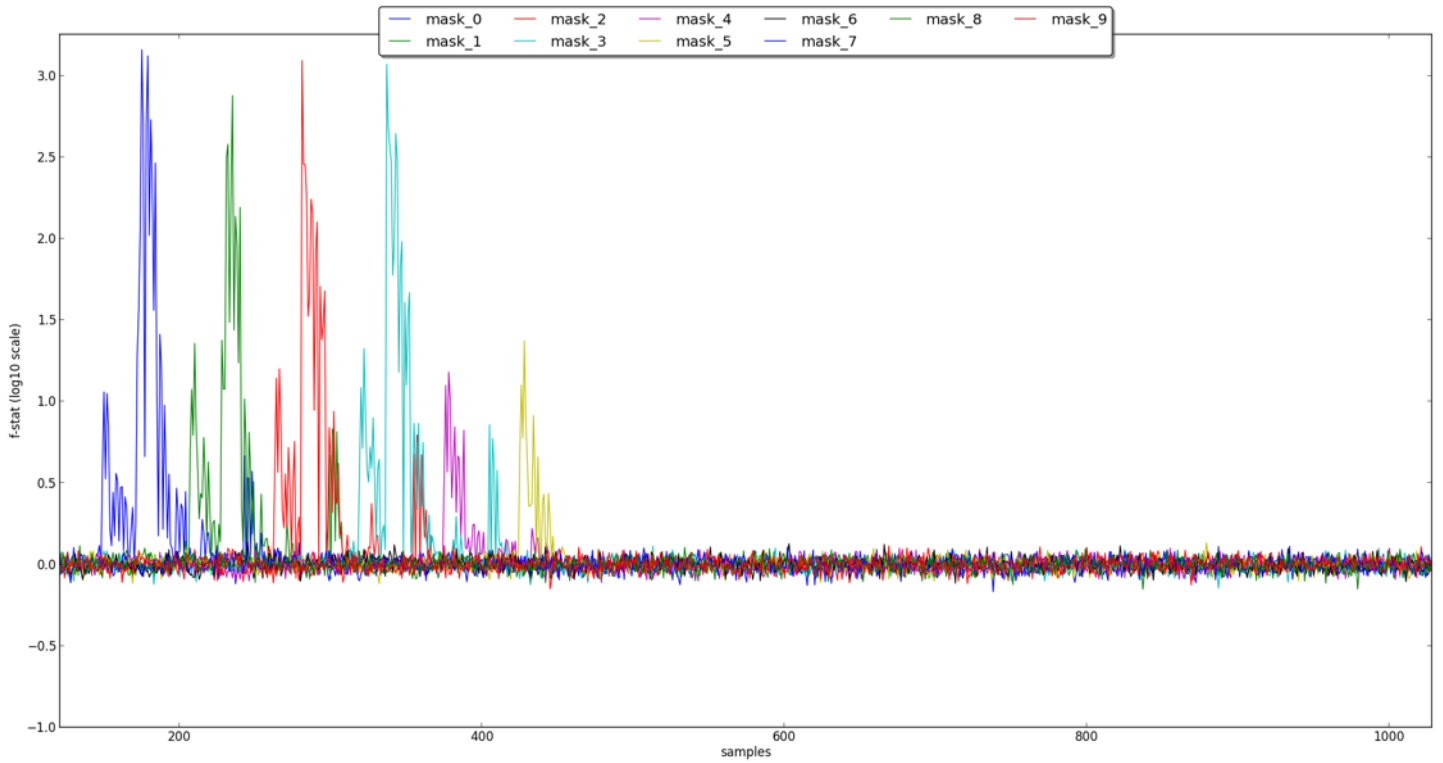


masked

➔ No more 1st order leakage with this masking scheme.

Identification of new leakage points

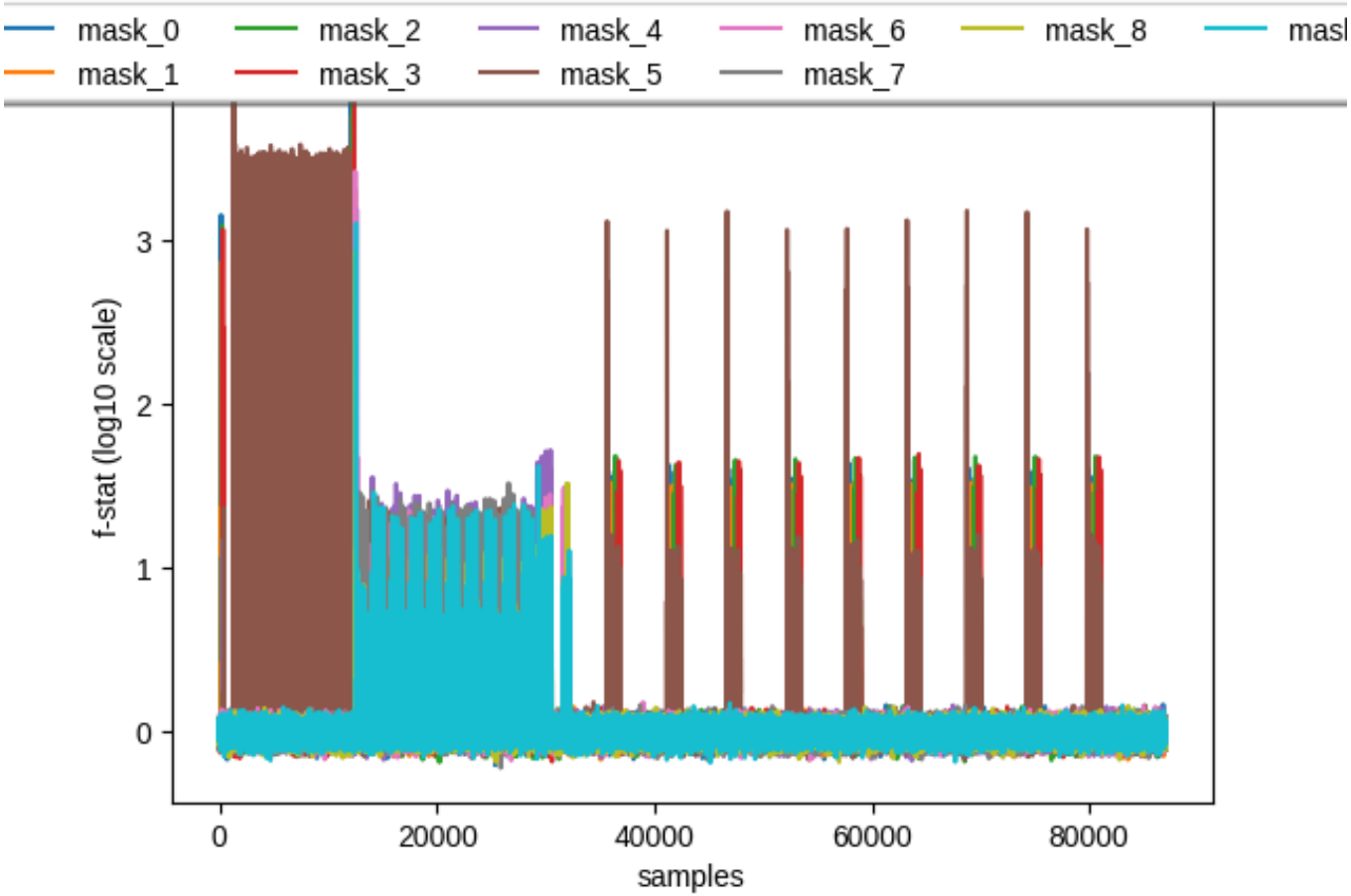
The **masks generation process** leaks information as well (*F*-test).
Generation of the 6 random masks (4 for MixColumn, 2 for SubBytes):



In the **worst case scenario** (profiled attacks), these can be **combined** with other leakage points later to perform a **second order attack**.
(M ; SBOX(P⊕K)⊕M) ➔ SBOX(P⊕K)

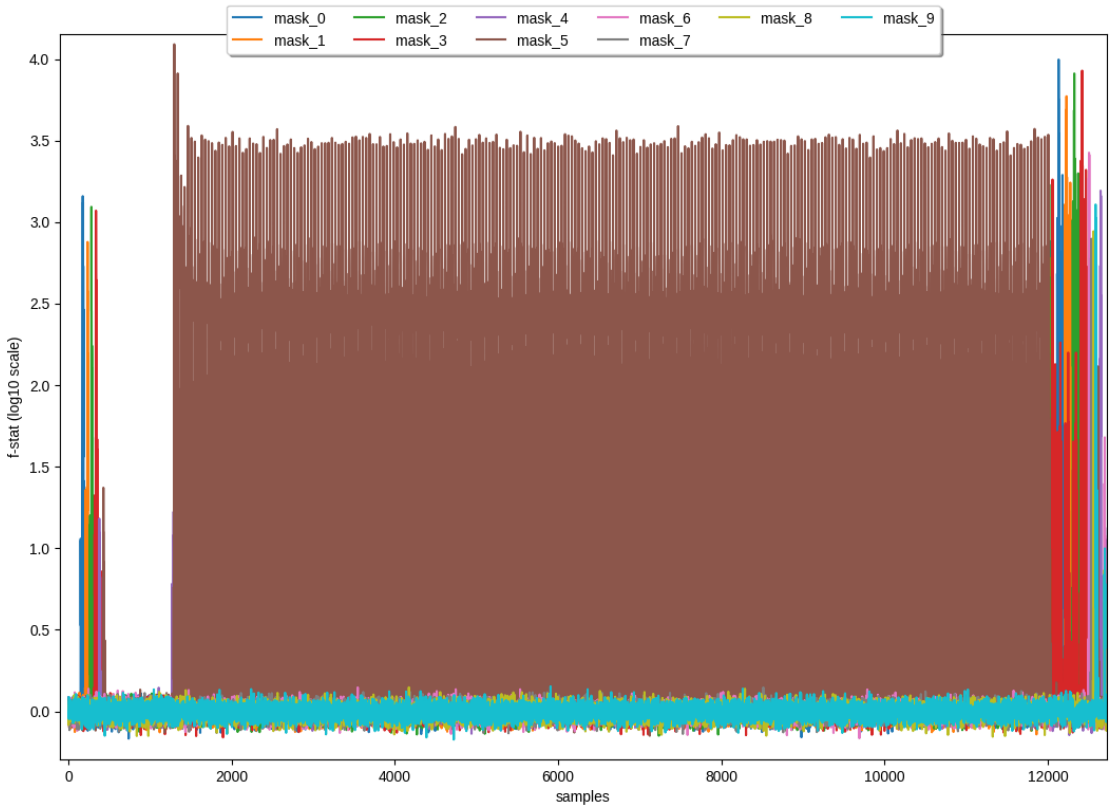
Identification of new leakage points

Interestingly, we can **see the masks manipulation** during the encryption process. The initial (masked) key schedule can also leak information or be profiled for efficient differential fault attack (DFA):



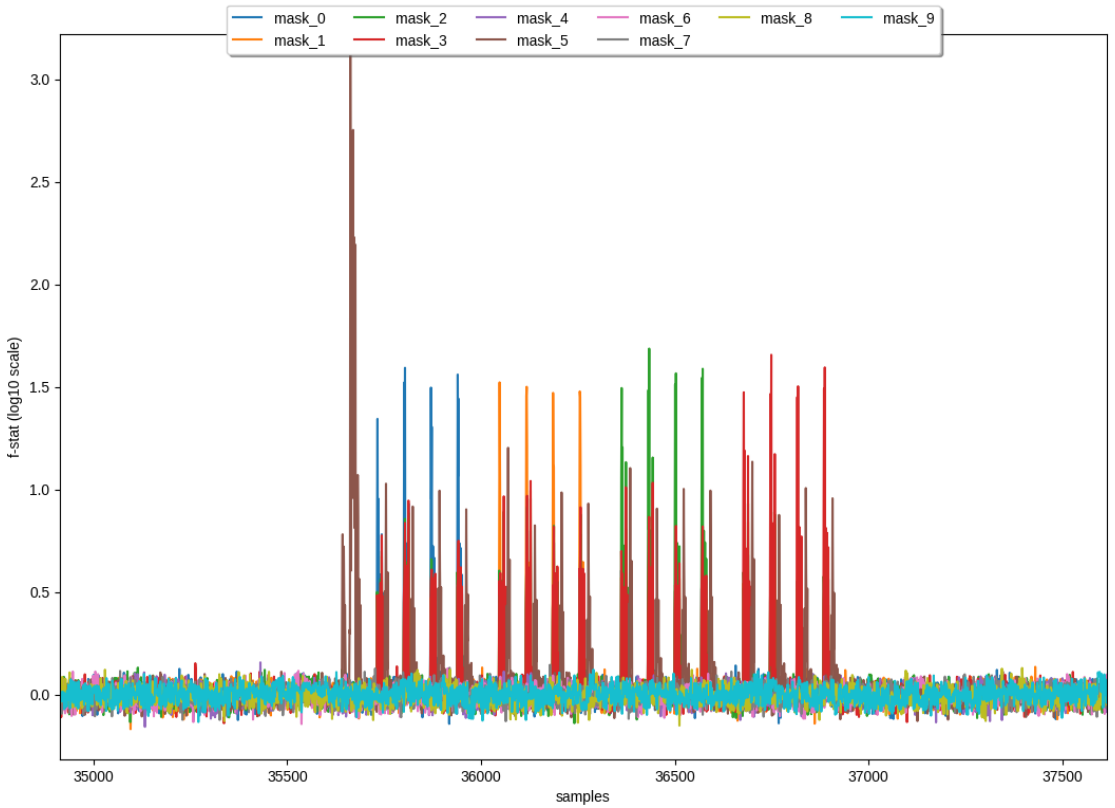
Identification of new leakage points

Interestingly, we can **see the masks manipulation** during the encryption process. The initial (masked) key schedule can also leak information or be profiled for efficient differential fault attack (DFA):



Identification of new leakage points

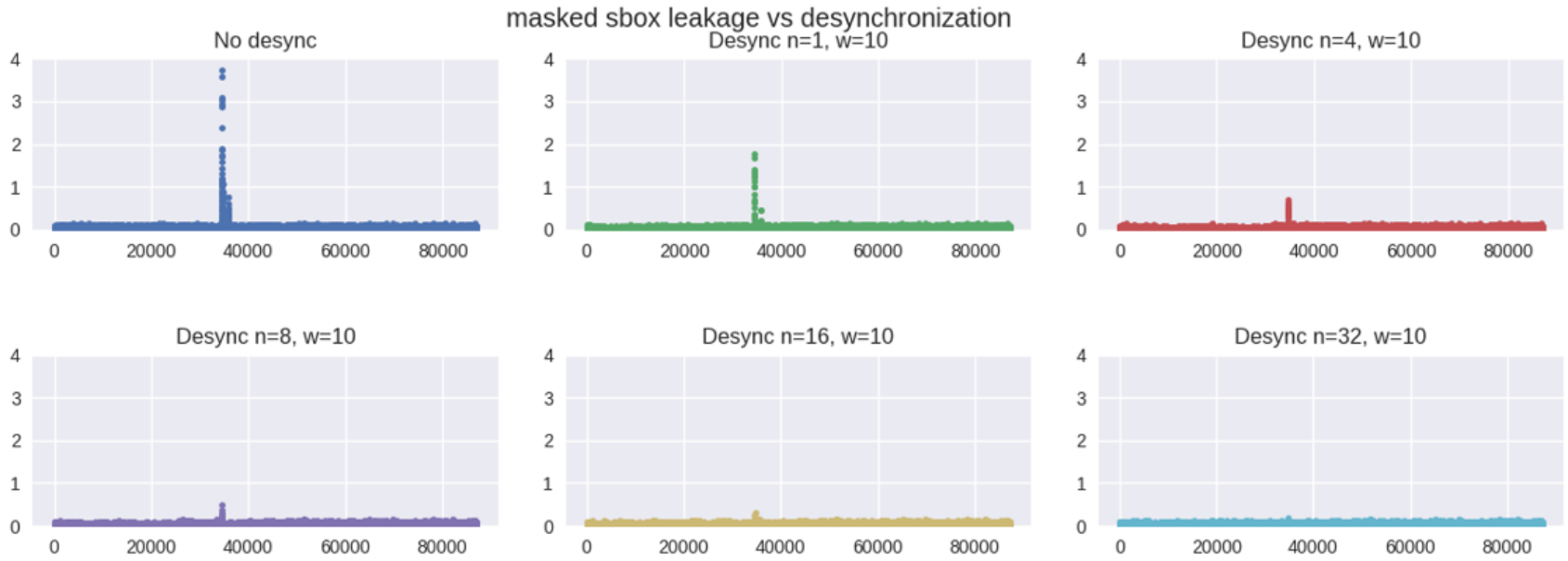
Interestingly, we can **see the masks manipulation** during the encryption process. The initial (masked) key schedule can also leak information or be profiled for efficient differential fault attack (DFA):



F-test on desynchronised traces

A second order CPA can target – jointly – the **two shares**.
Desynchronization-based protections can **reduce this exploitability**.

Leakage evaluation when **simulating desynchronisation** by randomly inserting *n* blocks of *w* NOPs during the execution:



➔ Leakage shrinks and becomes **unexploitable** (20000 traces here).

➔ Provide hints for protecting the design.

Ongoing works

On protected AES (masking, hiding), powerful **template** attacks need:

- Strong **information compression** (PCA, LDA) or
- Detection of **points of interest**
- **Resynchronization** techniques

➔ can become rapidly difficult in practice.

Machine Learning-based analysis can be helpful here [4] [5]

- **Deep learning**-based attacks against **masking**
- **Denoising** and **resynchronization** with **autoencoder**
- …

[4] Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, François-Xavier Standaert
Template Attacks vs. Machine Learning Revisited (and the Curse of Dimensionality in Side-Channel Analysis). COSADE 2015

[5] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, Cécile Dumas
Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database. IACR ePrint 2018

Fault attacks on VerifyPIN

Clock glitches

Different **hardened** VerifyPIN have been successfully bypassed:

- ✔ Constant-time
- ✔ Constant-time and inlined functions
- ✔ Constant-time and inlined functions and loop counter
- ✖ Constant-time and inlined functions and double call

Limitations

The ChipWhisperer platform **cannot** glitch at two different times.

Plan to overcome

We shall shoot with the **laser**!

Laser faults

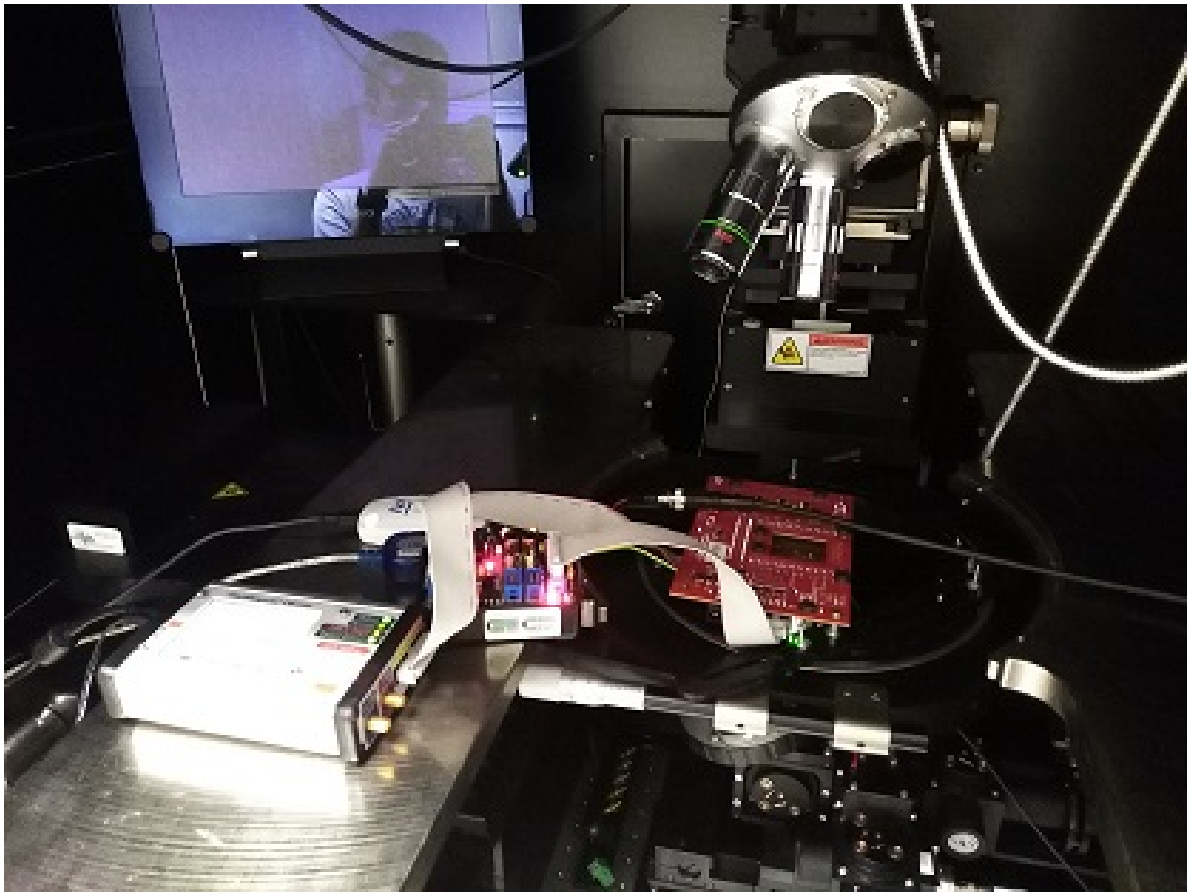
Preparatory work

- ✔ Design a **custom** ChipWhisperer target board:
  - ✔ Front-side access
  - ✔ Back-side access
- ✔ Prepare the target: **decapsulate** the chip to access the die
- ✔ **Mechanical setup** of the target on the bench
- … Mapping out the faults:
  - x-y position,
  - power,
  - duration,
  - delay,
  - type of fault (skip, set, reset, flip, …)

Laser setup

Characteristics

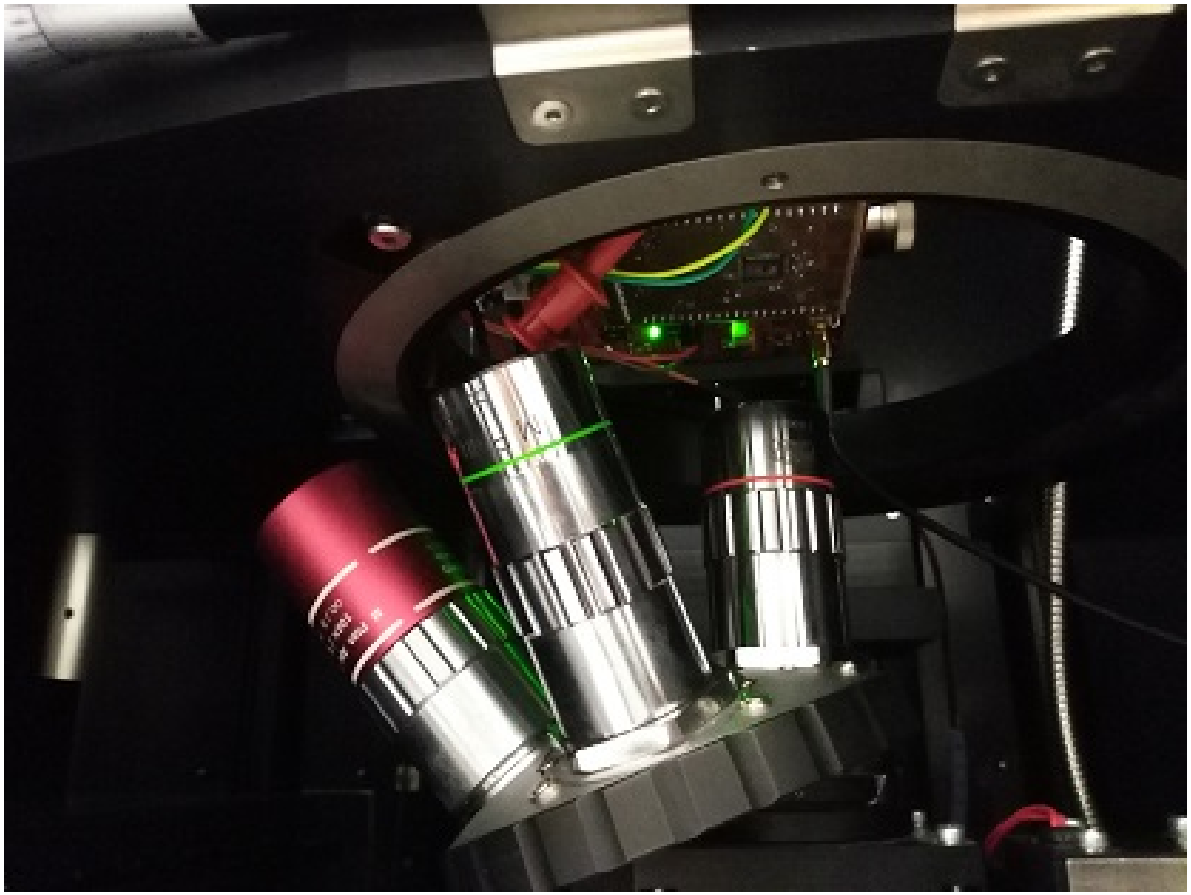
- IR (1064nm)
- >30ps
- 0-3W
- 3 objective lenses:
  - x5 (20µm)
  - x20 (5µm)
  - x100 (1µm)



Laser setup

Characteristics

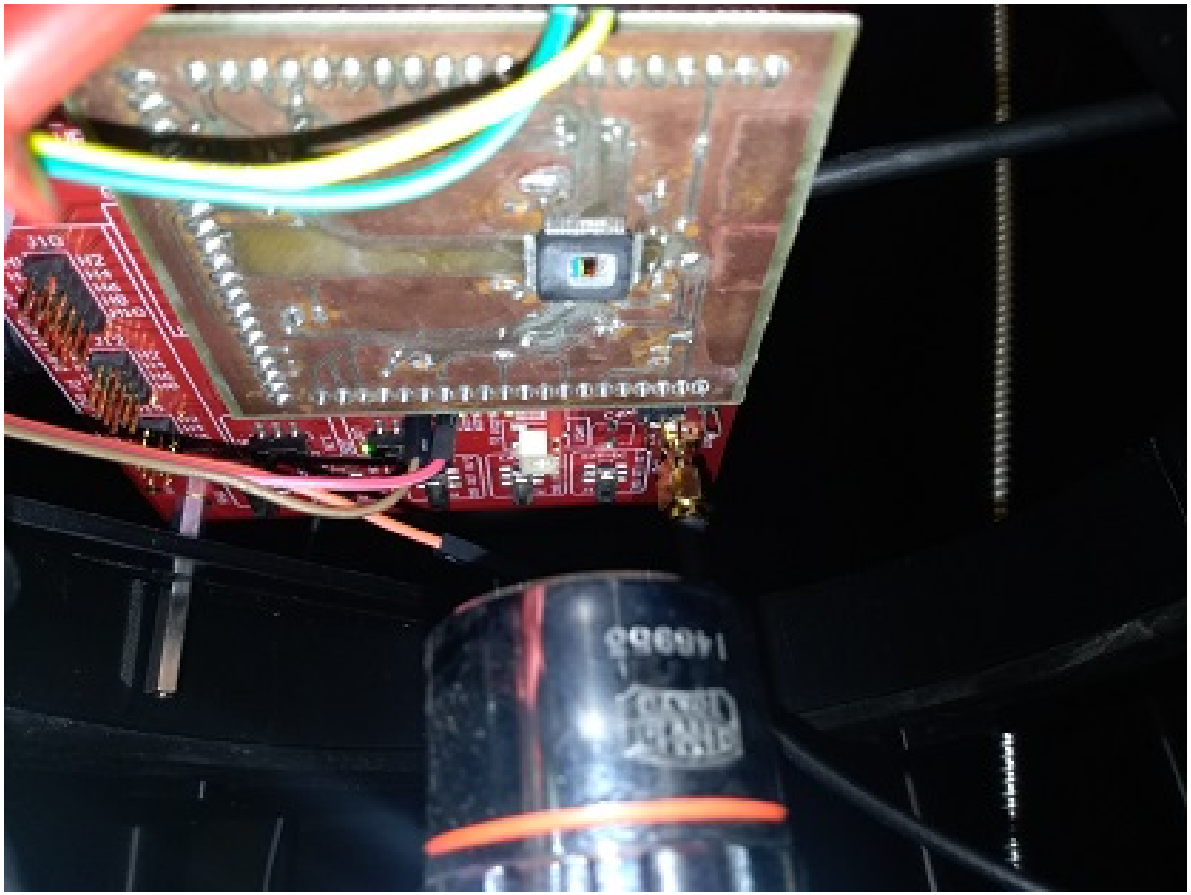
- IR (1064nm)
- >30ps
- 0-3W
- 3 objective lenses:
  - x5 (20µm)
  - x20 (5µm)
  - x100 (1µm)



Laser setup

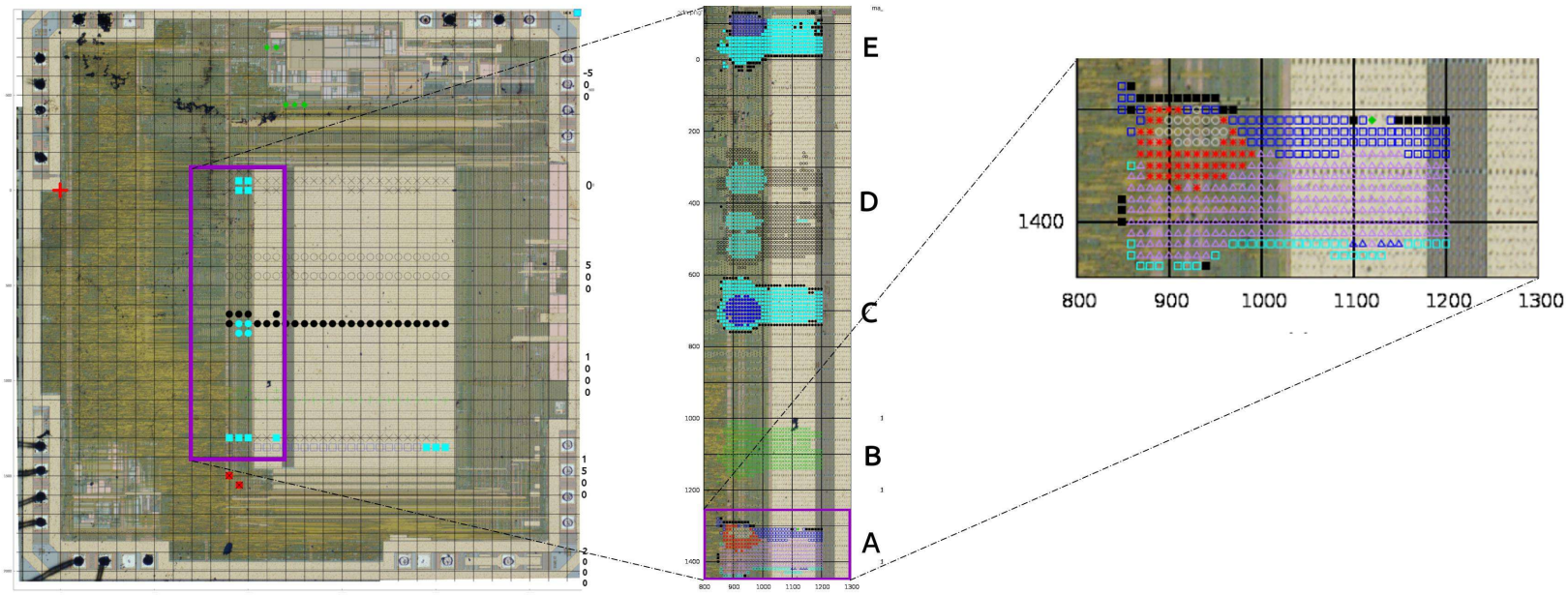
Characteristics

- IR (1064nm)
- >30ps
- 0-3W
- 3 objective lenses:
  - x5 (20µm)
  - x20 (5µm)
  - x100 (1µm)



8-bit microcontroller results

Instruction skip fault model previously validated experimentally [6]



[6] Practical results on laser-induced instruction-skip attacks into microcontrollers.
T. Riom, J.-M. Dutertre, O. Potin, J.-B. Rigaud, TRUDEVICE Workshop 2016, Barcelona

8-bit microcontroller results

This time, **all implementations are vulnerable**.

- ✔ Constant-time
- ✔ Constant-time and inlined functions
- ✔ Constant-time and inlined functions and loop counter
- ✔ Constant-time and inlined functions and double call
- ✔ Constant-time and inlined functions and control-flow integrity

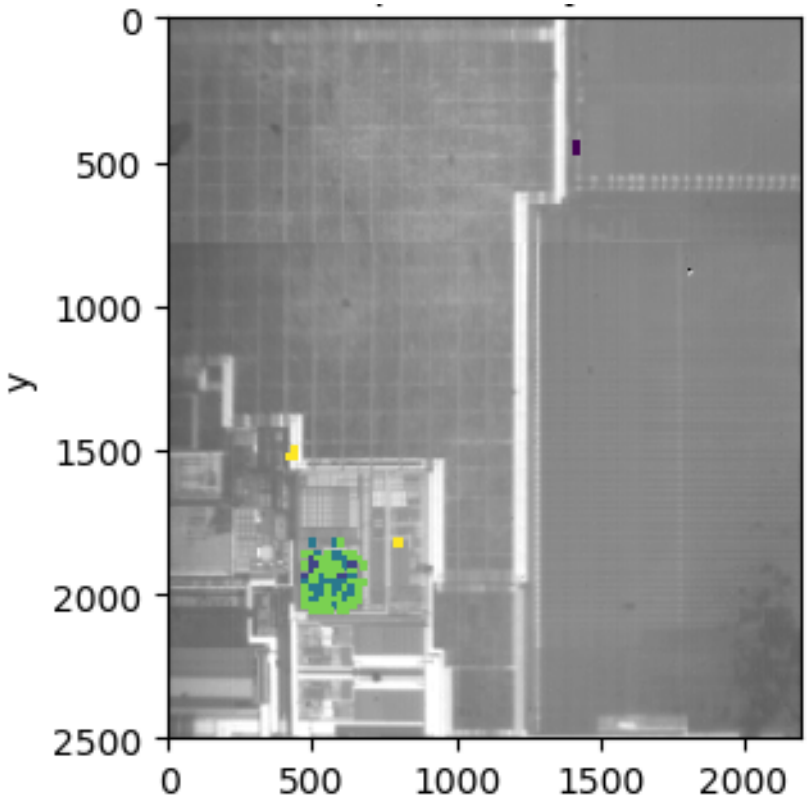
Paradox

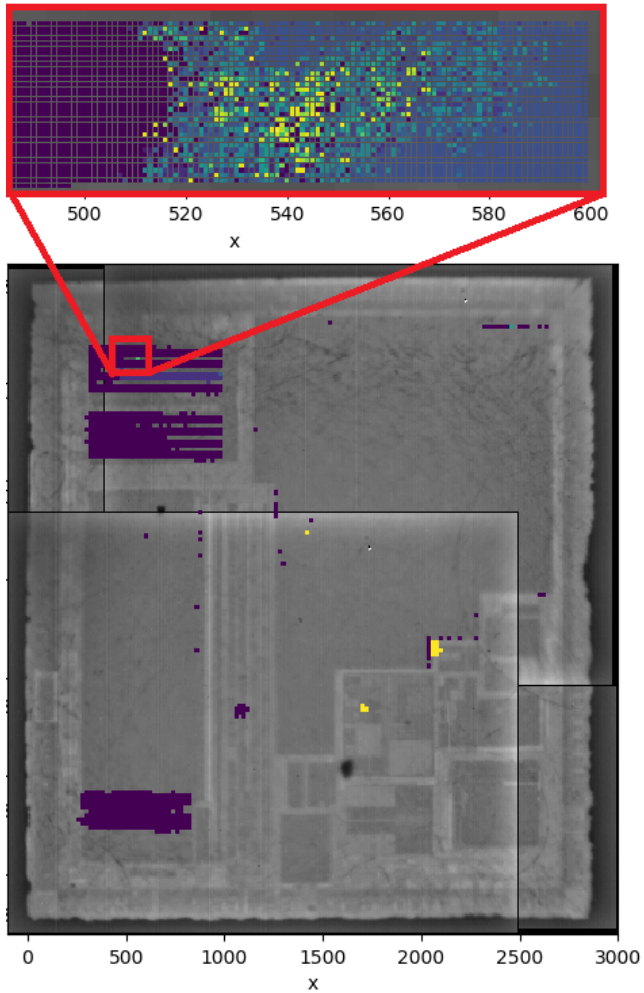
Constant-time implementation makes laser attacks much **easier**

32-bit microcontroller ongoing works

A **more complex** target (32 bits) implies:

- **Larger area** to cover for cartography,
- More time variability





32-bit microcontroller ongoing works

A **more complex** target (32 bits) implies:

- **Larger area** to cover for cartography,
- More time variability

TO DO:

- Experimentally validate the various fault models,
- Reproduce the attacks on VerifyPIN (skip instruction)
- Specific attacks on AES:
  - Differential fault attack [7]
  - Combined attacks (Fault analysis + Side-channel) [8]

[7] Christophe Giraud
DFA on AES. AES Conference 2004

[8] Thomas Roche, Victor Lomné, Karim Khalfallah
Combined Fault and Side-Channel Attack on Protected Implementations of AES. CARDIS 2011

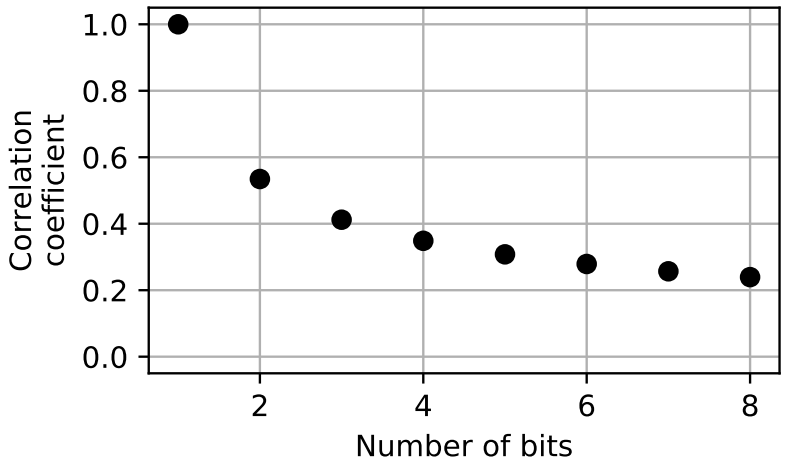
Combination of protections

For the best: 2nd order CPA made harder

Principle of 2nd order CPA: attack two S-box output bytes.
Traditionally, target the two shares (mask + masked value) but two consecutive bytes work well:

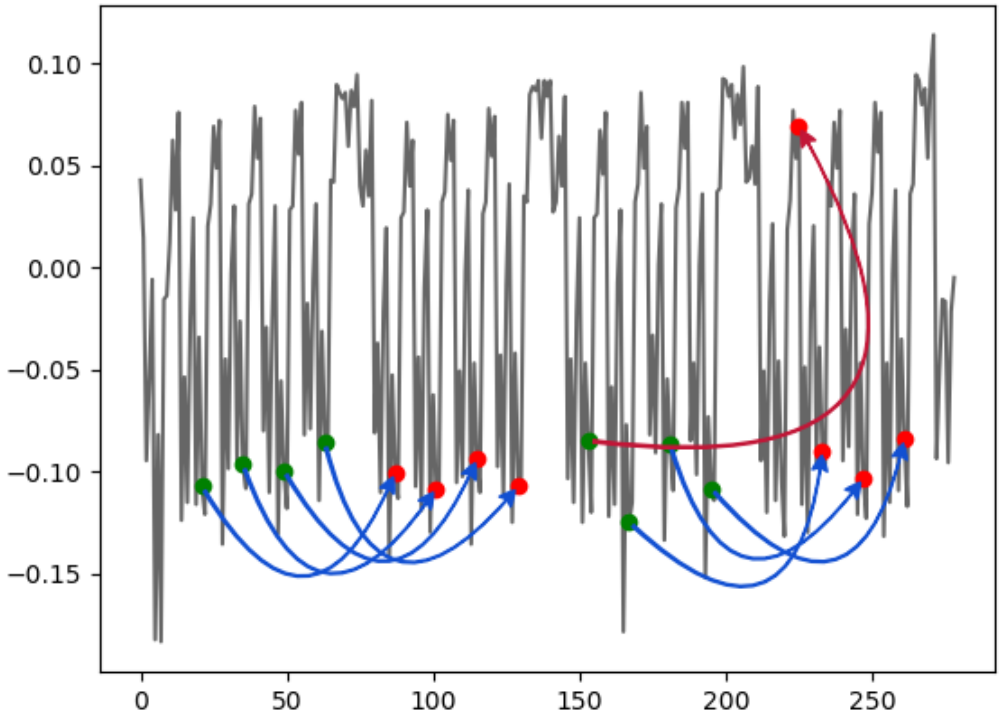
- $|\mathrm{Leak}(\mathrm{Sbox}(P_i \oplus K_i) \oplus M') - \mathrm{Leak}(\mathrm{Sbox}(P_j \oplus K_j) \oplus M')|$
- $\mathrm{HW}(\mathrm{Sbox}(P_i \oplus K_i) \oplus M' \oplus \mathrm{Sbox}(P_j \oplus K_j) \oplus M')$

$= \mathrm{HW}(\mathrm{Sbox}(P_i \oplus K_i) \oplus \mathrm{Sbox}(P_j \oplus K_j))$ ➔ no more mask !



For the best: 2nd order CPA made harder

Combining leakages is **easy** when traces are **perfectly synchronised**.



800 traces required to break 1st-roder masked AES on STM32.

A **desynchronising countermeasure** is very **powerful** here!

For the worst

Countermeasure against FA **or** SCA are usually compatible.

Countermeasure against FA **and** SCA can be incompatible.

Example

Redundancy-based protection against Fault Injection Analysis can **enhance** side-channel leakages…

Side-Channel Analysis is not only for key recovering purpose, it also helps in temporaly **profiling** fault injection (bypassing secure boot [9])

Each case must be evaluated **separately**.

[9] Niek Timmers, Albert Spruyt, Bypassing Secure Boot using Fault Injection, Black Hat Europe 2016

Conclusion

Conclusion

- Inserting protections at **software level** is powerful
- Leakage assessment is a great tool to design protections
  - Provides **metrics** of leakage reduction efficiency
- **Combinations** of protections is a **double-edged** sword

Conclusion

- Inserting protections at **software level** is powerful
- Leakage assessment is a great tool to design protections
  - Provides **metrics** of leakage reduction efficiency
- **Combinations** of protections is a **double-edged** sword

— Questions ? —

Contacts: Brice Colombier
brice.colombier@cea.fr

Pierre-Alain Moëllic
pierre-alain.moellic@cea.fr

Commissariat à l'énergie atomique et aux énergies alternatives
17 rue des Martyrs | 38054 Grenoble Cedex
www.cea-tech.fr

Établissement public à caractère industriel et commercial | RCS Paris B 775 685 019