CRYPTOEXPERTS

WE INNOVATE TO SECURE YOUR BUSINESS

Presentation of the VeriSiCC Project (2018-2022)

Sonia Belaïd

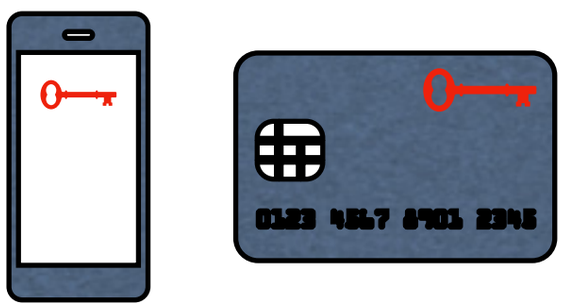
September 22, 2022

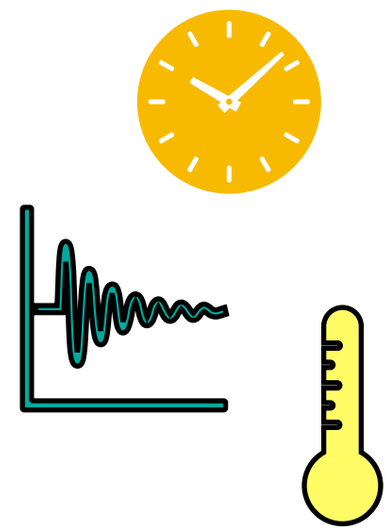
Sincere thanks to LIP6 which graciously makes a room available for this event.

Motivation

Side-Channel Attacks and Masking

Cryptographic implementations



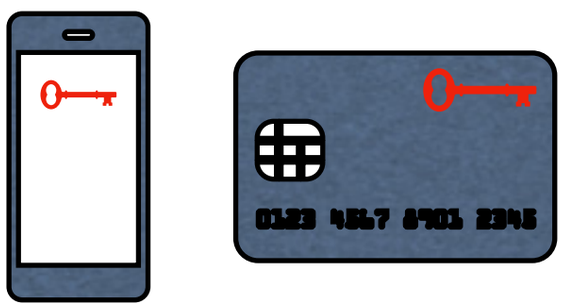


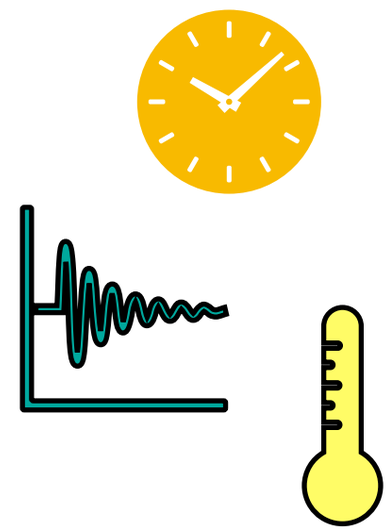
Mathematical tools:

- Probability theory
- Linear algebra
- Statistics

Side-Channel Attacks and Masking

Cryptographic implementations

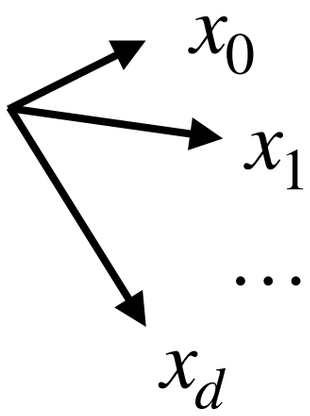




Mathematical tools:

- Probability theory
- Linear algebra
- Statistics

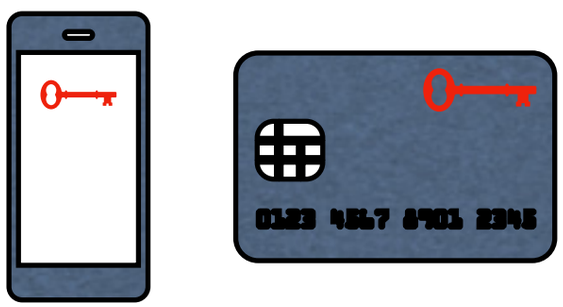
Masking: each sensitive variable x

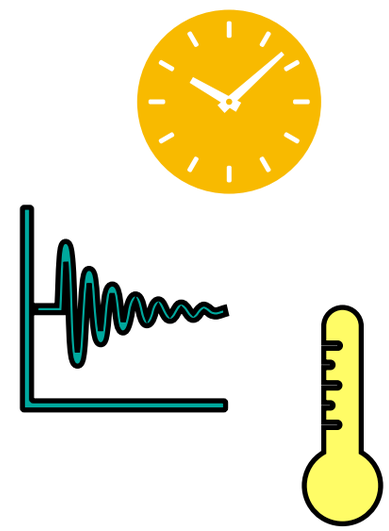


$$x = x_0 \star x_1 \star \ldots \star x_d$$

Side-Channel Attacks and Masking

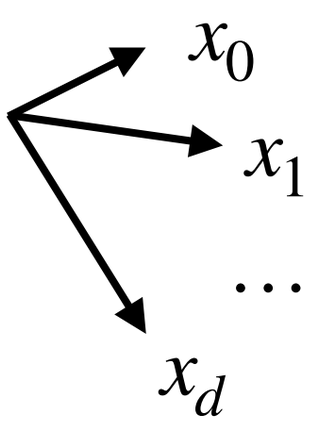
Cryptographic implementations





Mathematical tools:
- Probability theory
- Linear algebra
- Statistics

Masking: each sensitive variable x

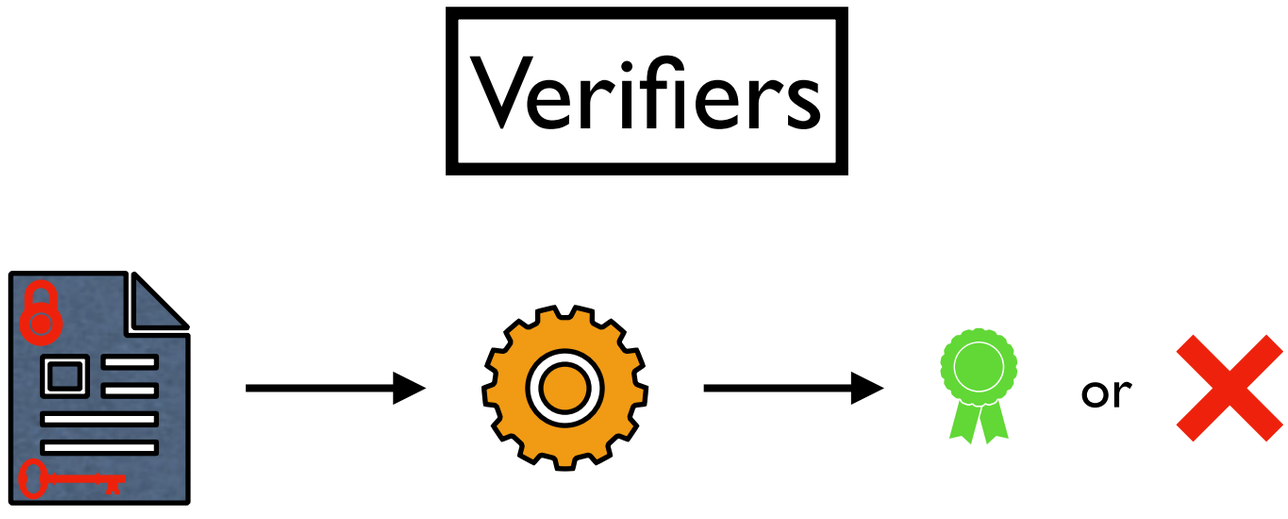


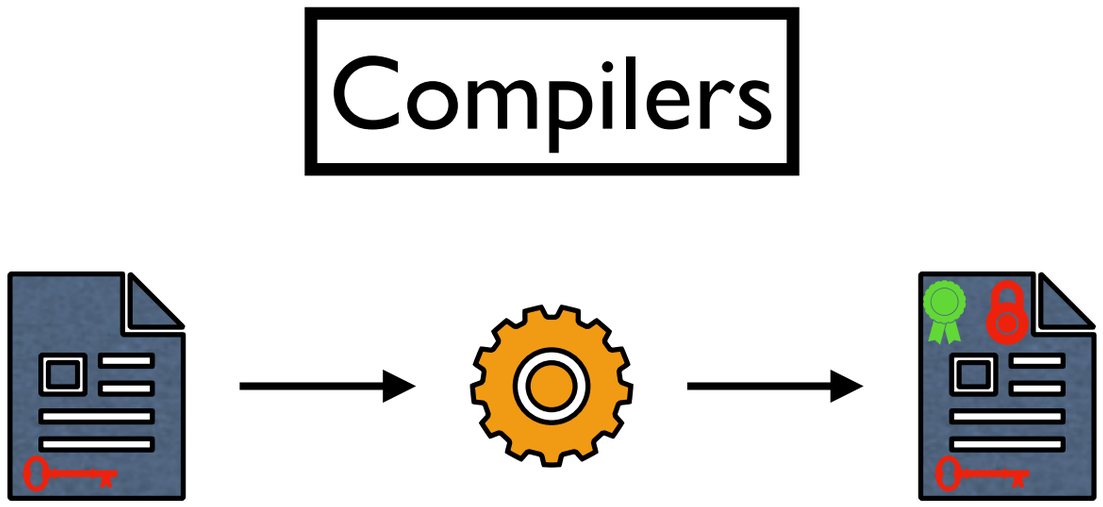
$$x = x_0 \star x_1 \star \ldots \star x_d$$

Attack complexity:
a priori exponential in d

Implementation complexity:
at least quadratic in d

Verifiers and Compilers





Limitations of Current Tools

Verifiers

- Efficiency: exponential in the size of the circuit
- Scope: not dedicated to a final embedded devices, possible errors while adapting the output implementation
- Realism: only dependent on abstract leakage models

Compilers

- Efficiency: inefficient implementations with a large amount of randomness
- Scope: not all the input operations are handled
- Realism: only dependent on abstract leakage models

Actors & Objectives

Actors

FUI project funded by bpifrance and région Ile-de-France

6 partners

ANSSI

PARIS

INRIA

SOPHIA ANTIPOLIS

IDEMIA

COURBEVOIE

UNIVERSITÉ DU LUXEMBOURG

LUXEMBOURG

NINJALAB

MONTPELLIER

CRYPTOEXPERTS

PROJECT COORDINATOR

PARIS

VeriSiCC Objectives

Characterization Method

- Determine what leakage to evaluate

Verifiers

- Improve the current verifiers in terms of efficiency, scope and realism

Compilers

- Improve the current verifiers in terms of efficiency, scope and realism

Project Organization

Tasks

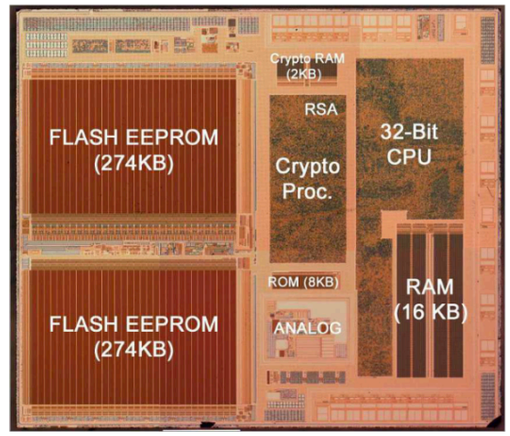
- SP1: Analysis of the state of the art and needs
- SP2: Selection of new techniques
  - Formal language and countermeasures
  - Efficient formal verification of implementations
  - Generation and optimization of secure implementations
- SP3: Prototypes
  - Verification tool
  - Compiler
  - Characterization of devices
- SP4: Demonstrator

Project Organization

- Regular meetings between all the partners
  - Every 3 months with a different partner (when possible)
- 2 public seminars with selected invited speakers
  - September, 25th 2019 at Sorbonne University
  - Septembre, 22nd 2022 at Sorbonne University

Outputs

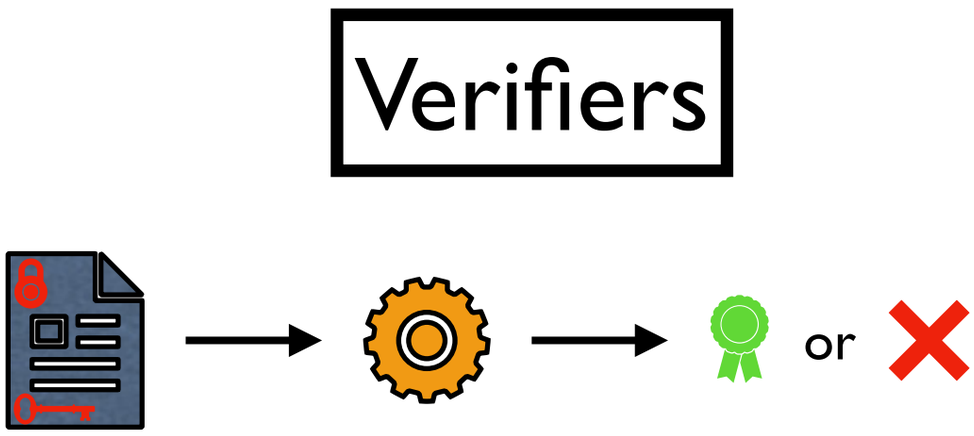
Characterization

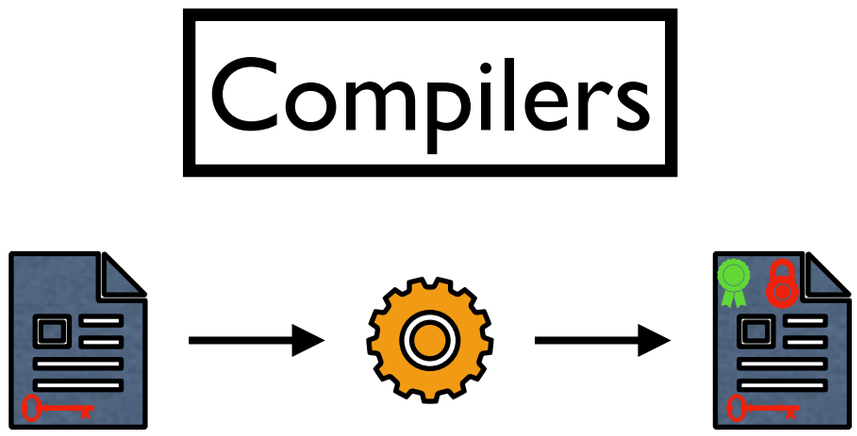


- One device: OpenCard (ARM core SC 100)

- Study of single instructions on 32-bit registers: *adds*, *eors* and *ands*
  - significant leakage of the least significant byte
  - observed leakages are well modelled by the Hamming Weight
  - the register choice shows to have an impact on the side-channel leakage
  - transition leakages from two consecutive instructions over two sets of independent registers
- Study of a masked bit-wise AND gadget with 4 shares
  - strong leakage related to all 4 bytes of the manipulated registers
  - no leakage on combination of shares

Verifiers and Compilers





* maskVerif: hardware implementations
* Tornado: register-probing model
* VRAPS: random probing model
* scverif: device features
* IronMask: completeness

* Tornado: least number of refresh from three main gadgets
* Random probing compiler

Practical Evaluation

Target cryptographic primitive: PRESENT S-box

- 3 implementations:
  - Unprotected
  - 2-share from [BGG+21]
  - 3-share from [BGG+21]

- Main results
  - Leakage similar to what was obtained in the characterization phase
  - First-order leakage on the 2-share implementation on the OpenCard
  - No first- or second-order leakage on the 3-share implementation on the OpenCard

What's Next?

- Converge towards a generic characterization
- Include this automatic characterization into our verifiers and compilers
- Verify on concrete devices again