

SIMPLE-Crypto Association

Open-Source & Non-Profit Hand in Hand

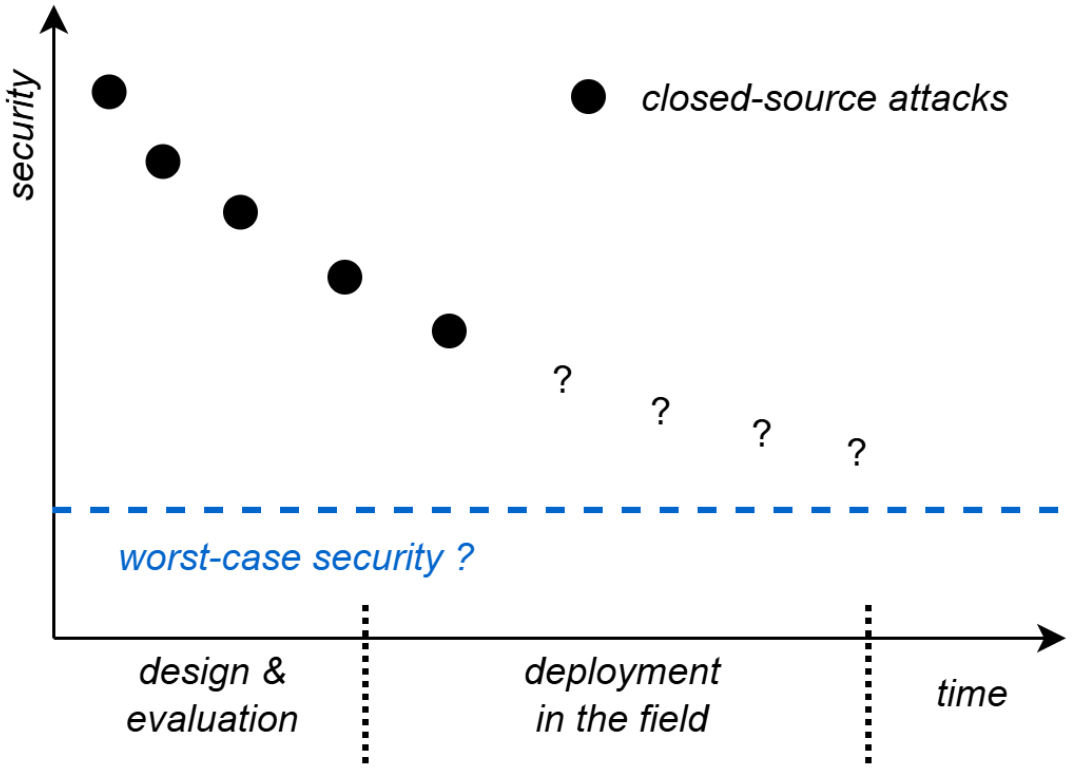
F.-X. Standaert

POpSI Workshop, Bochum, Germany
September 2026

Credits: G. Cassiers, C. Momin, T. Moos, B. Udvahelyi, O. Bronchain, ….

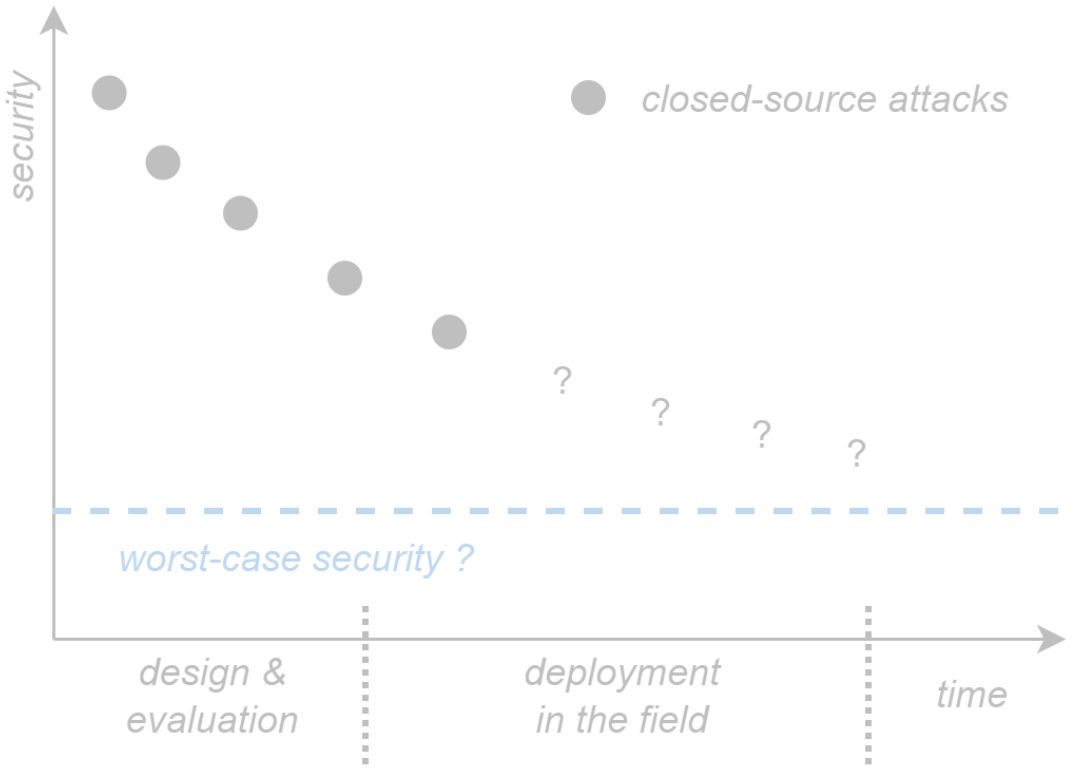
Cryptographic specificities (I)

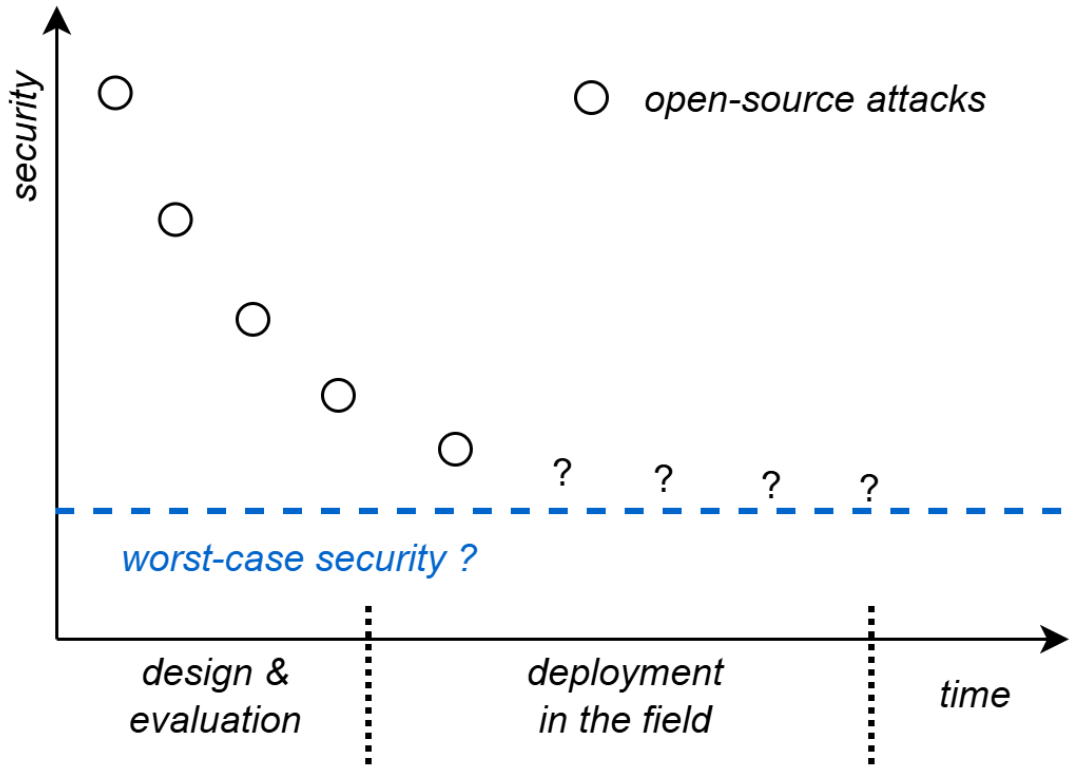
- Long-term (worst-case) security is hard to anticipate / predict



Cryptographic specificities (I)

- Long-term (worst-case) security is hard to anticipate / predict

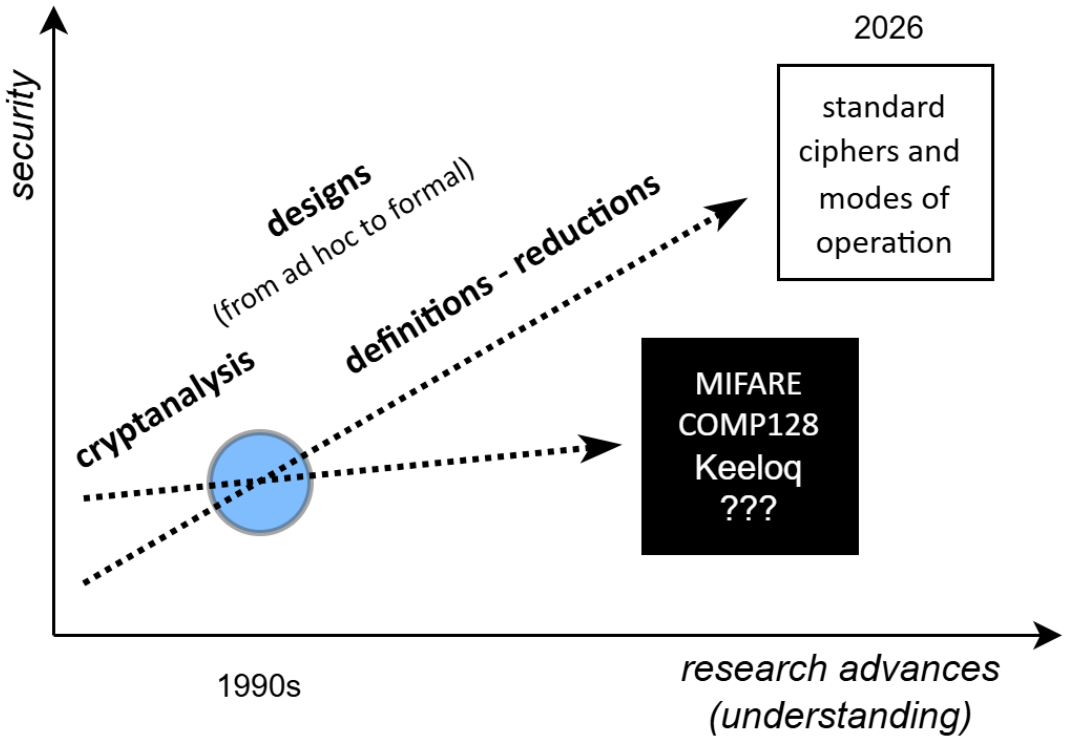




- Anticipation is easier in an open setting (more public scrutiny)

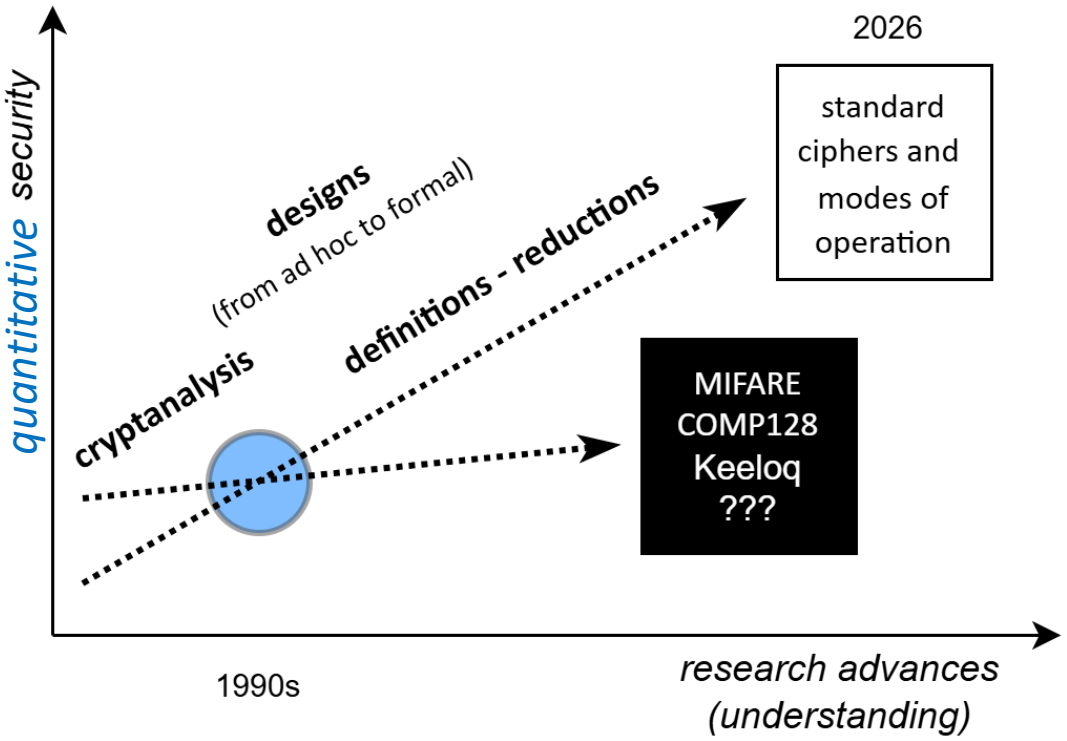
Cryptographic specificities (II)

- As research advances, the advantages of security by obscurity vanish and the interest of open-source solutions increases
  - Well known for algorithms



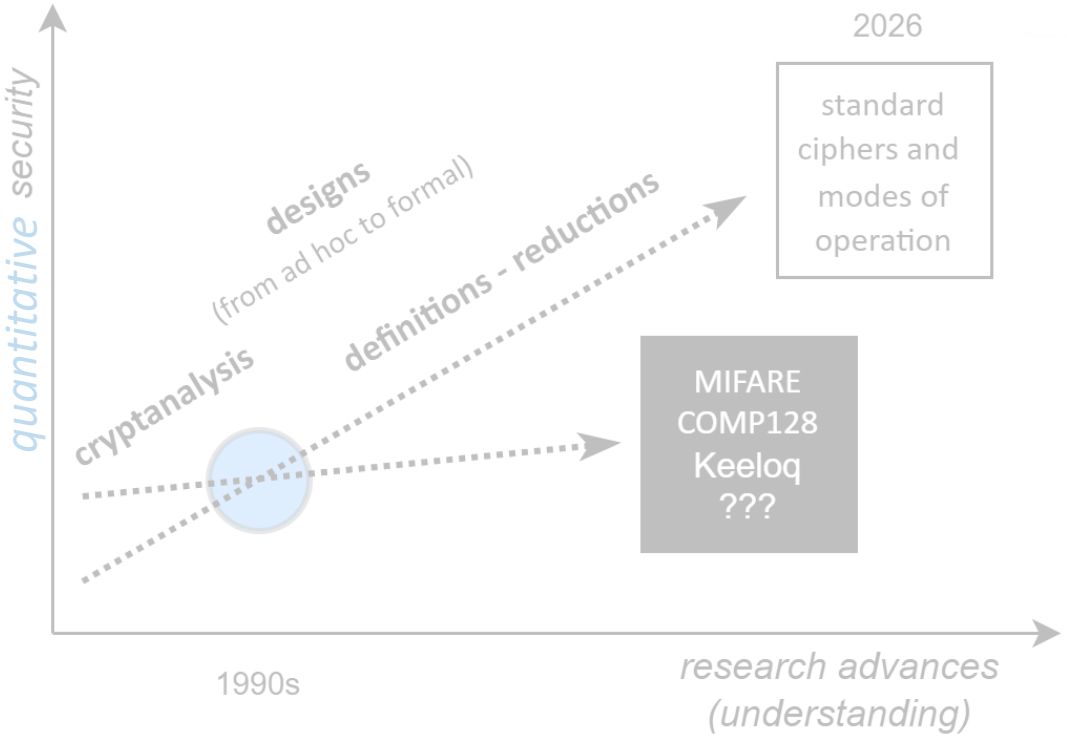
Cryptographic specificities (II)

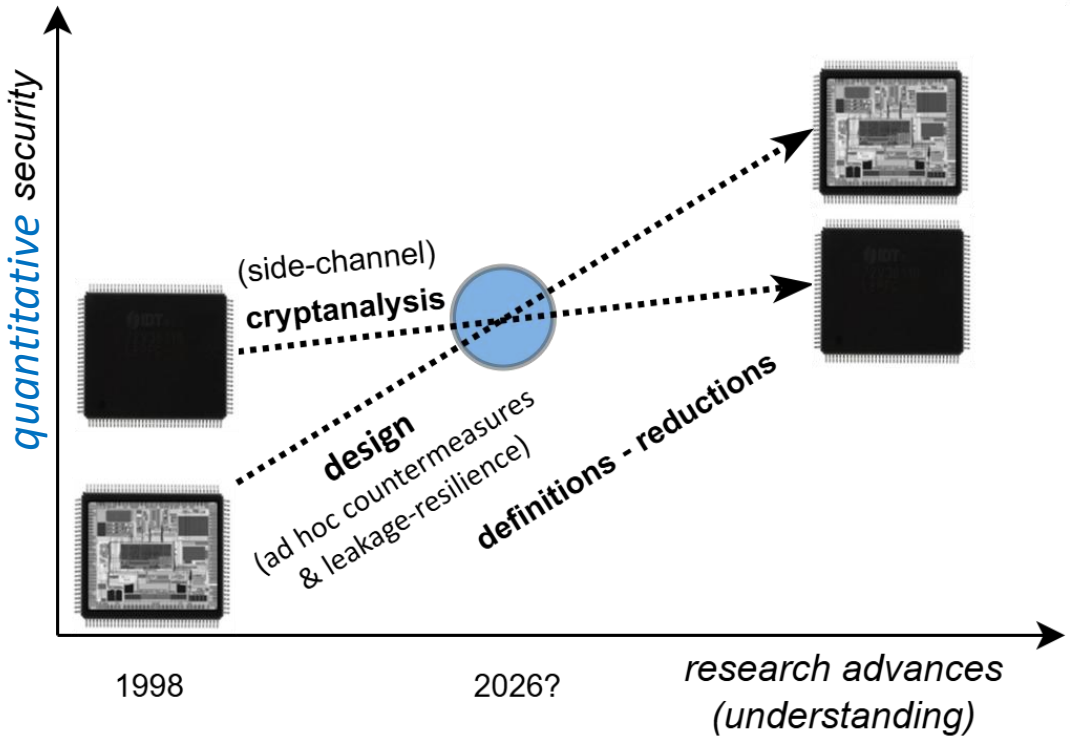
- As research advances, the advantages of security by obscurity vanish and the interest of open-source solutions increases
  - Well known for algorithms



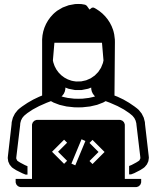
Cryptographic specificities (II)

- As research advances, the advantages of security by obscurity vanish and the interest of open-source solutions increases
  - Well known for algorithms, what about implementations?





Research generalities



paper & artifacts
(code, datasets, …)

? ?

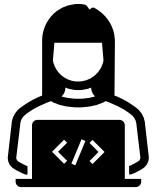


well-documented
implementation

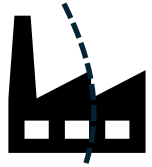


Code is rarely used / maintained

Research generalities



paper & artifacts
(code, datasets, …)

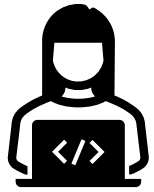




well-documented
implementation

- Code is used / maintained
- Becomes proprietary
- Redundant efforts

SIMPLE-Crypto goals



paper & artifacts
(code, datasets, …)

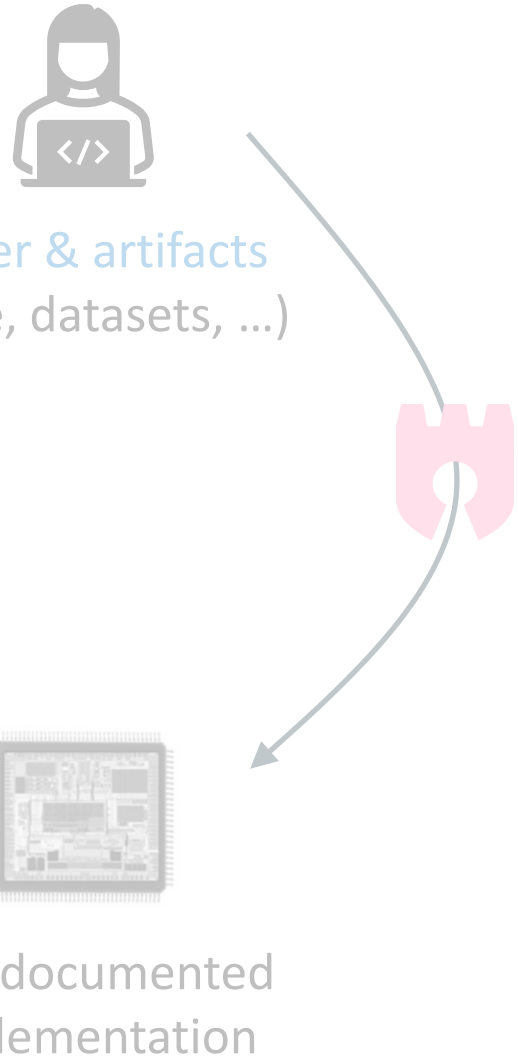




well-documented
implementation

- Code is used / maintained
- Remains public (open-source)
- Community efforts (non-profit)

SIMPLE-Crypto goals

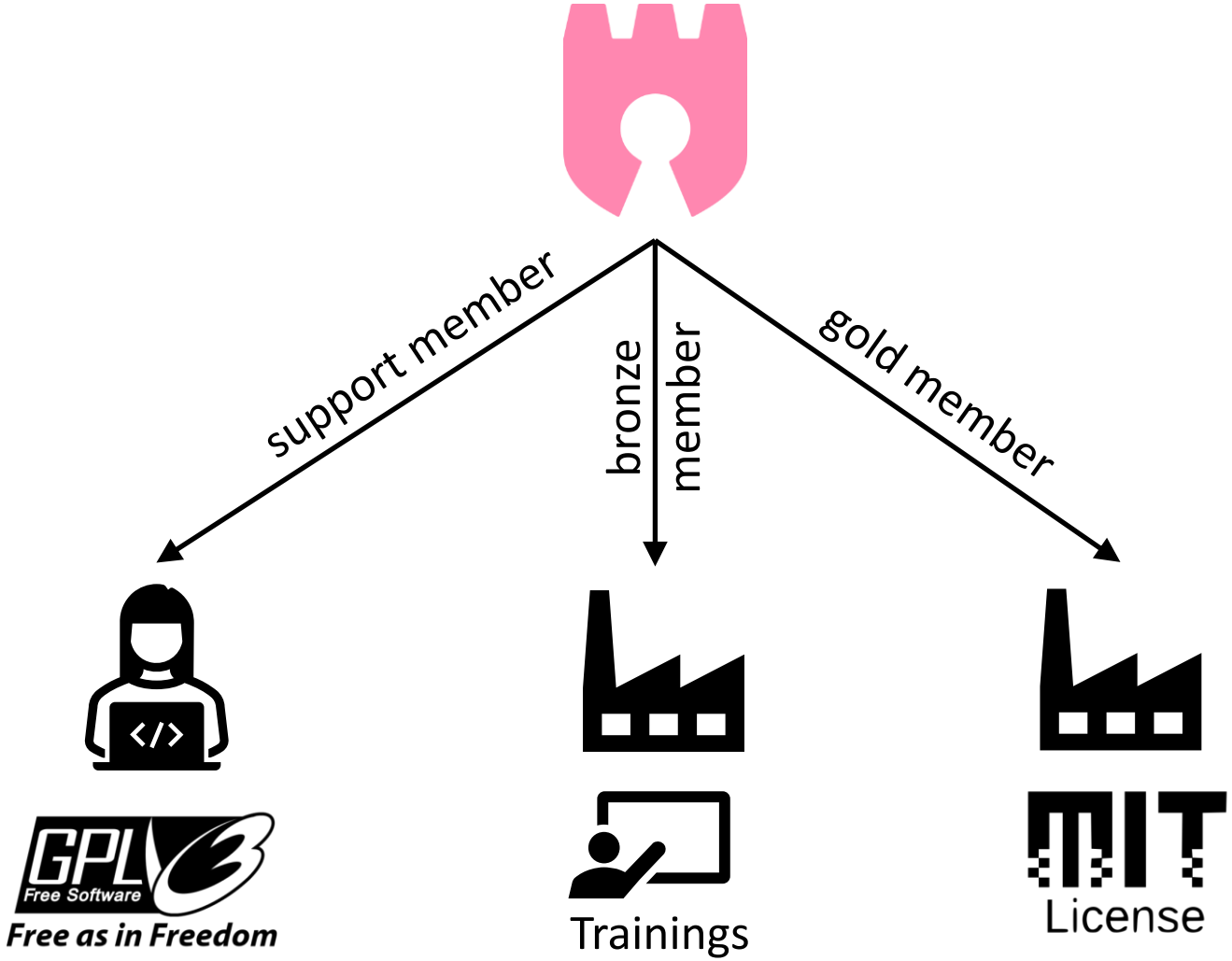


⇒ Improve the long-term security of cryptographic Implementations

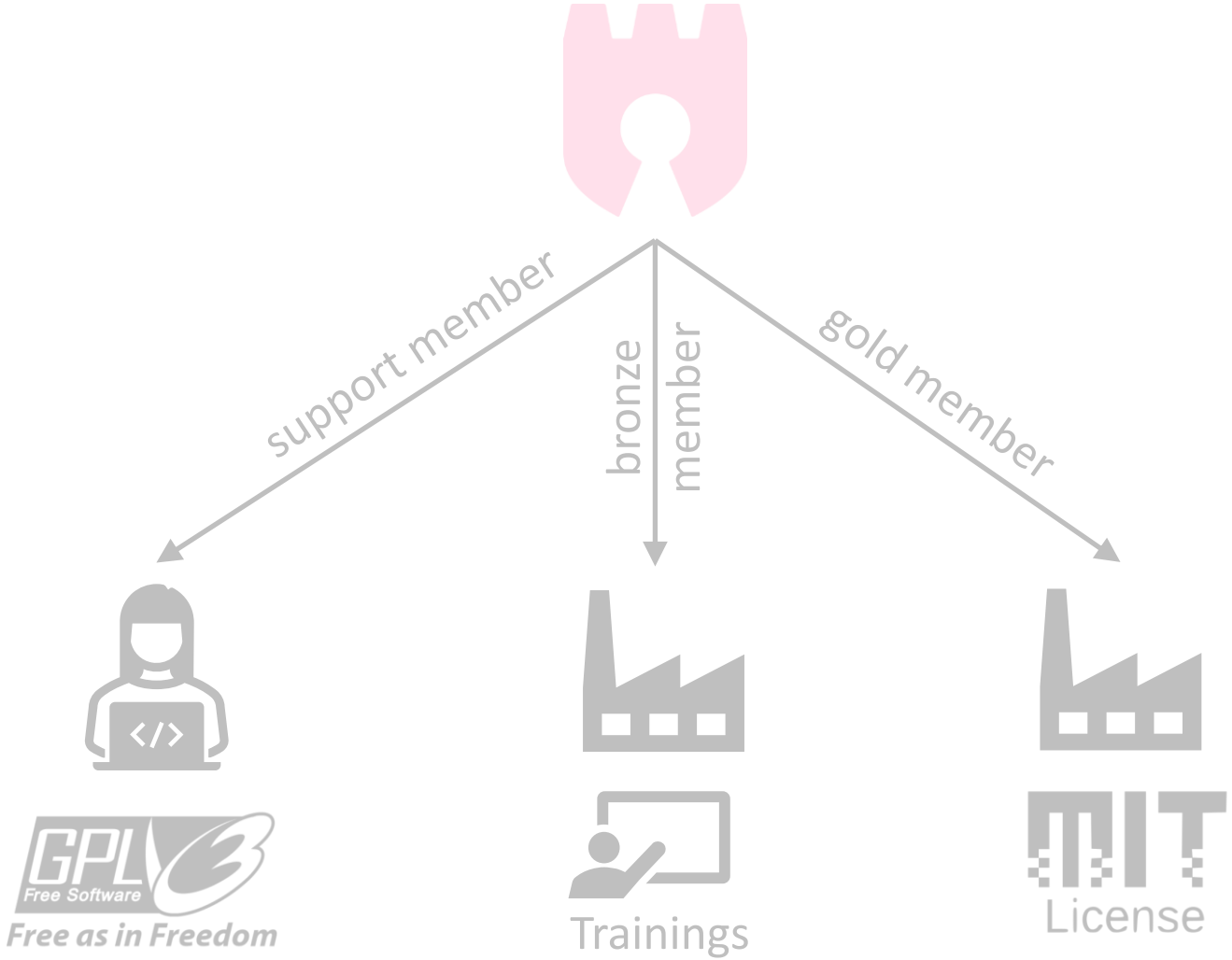
- Code is used / maintained
- Remains public (open-source)
- Community efforts (non-profit)

⇒ Serve as a constructive interface between academia & industry

Valorization model



Valorization model



Google

NXP

poetic technologies

NinjaLab

intel

life.augmented

lowRISC

NIST

PQSHIELD

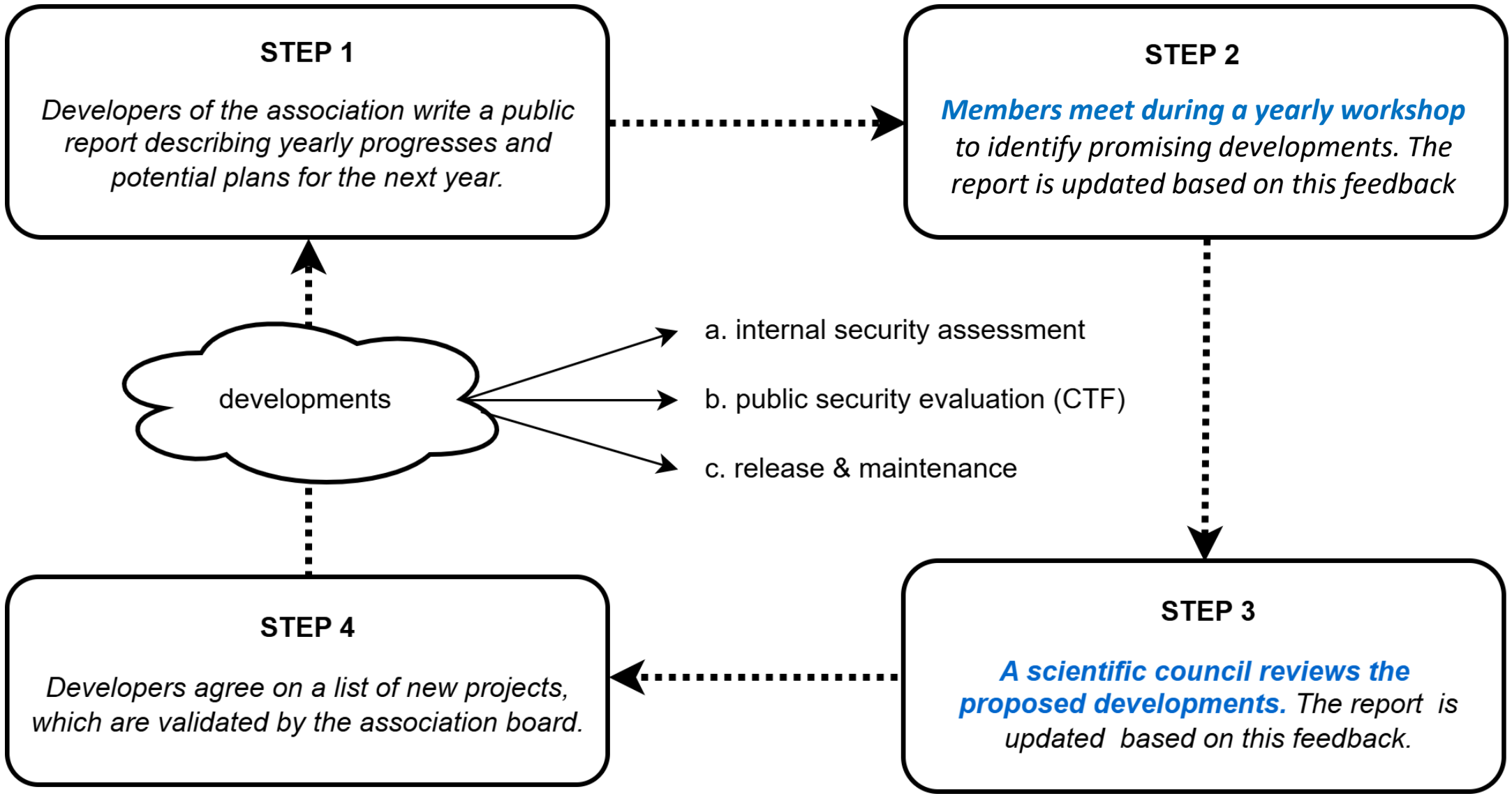
THALES
Building a future we can all trust

NATIONALE VEILIGHEIDSOVERHEID
AUTORITÉ NATIONALE DE SÉCURITÉ
NATIONAL SECURITY AUTHORITY

CRYPTO EXPERTS

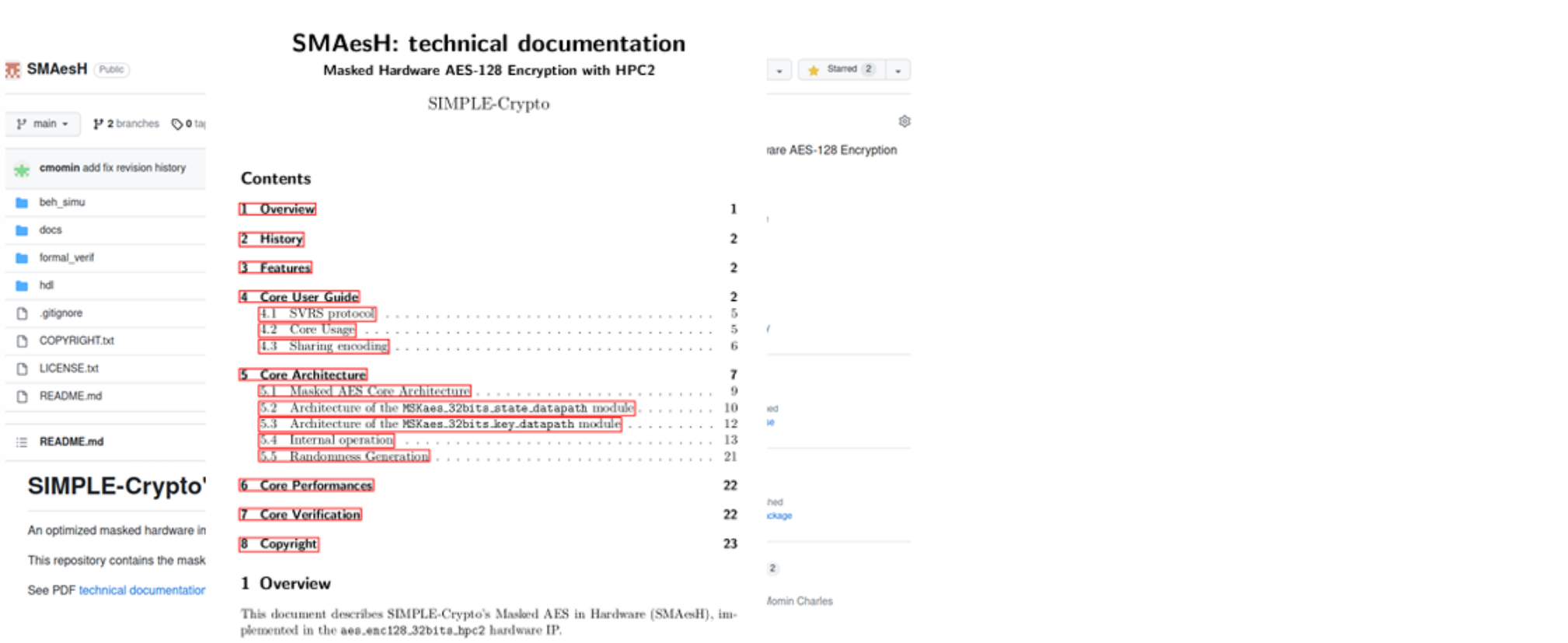
ANSSI

Yearly organization



Contributions (I): SMAEsH & Challenge

- Hardware AES implementation masked at arbitrary orders



SMAesH Public

main · 2 branches · 0 tag

cmomin add fix revision history

beh_simu
docs
formal_verif
hdl
.gitignore
COPYRIGHT.txt
LICENSE.txt
README.md

README.md

SIMPLE-Crypto'

An optimized masked hardware in

This repository contains the mask

See PDF technical documentatior

SMAesH: technical documentation

Masked Hardware AES-128 Encryption with HPC2

SIMPLE-Crypto

Contents

1 Overview 1
2 History 2
3 Features 2
4 Core User Guide 2
4.1 SVRS protocol 5
4.2 Core Usage 5
4.3 Sharing encoding 6
5 Core Architecture 7
5.1 Masked AES Core Architecture 9
5.2 Architecture of the MSKaes_32bits_state_datapath module 10
5.3 Architecture of the MSKaes_32bits_key_datapath module 12
5.4 Internal operation 13
5.5 Randomness Generation 21
6 Core Performances 22
7 Core Verification 22
8 Copyright 23

1 Overview

This document describes SIMPLE-Crypto's Masked AES in Hardware (SMAesH), implemented in the aes_emc128_32bits_hpc2 hardware IP.

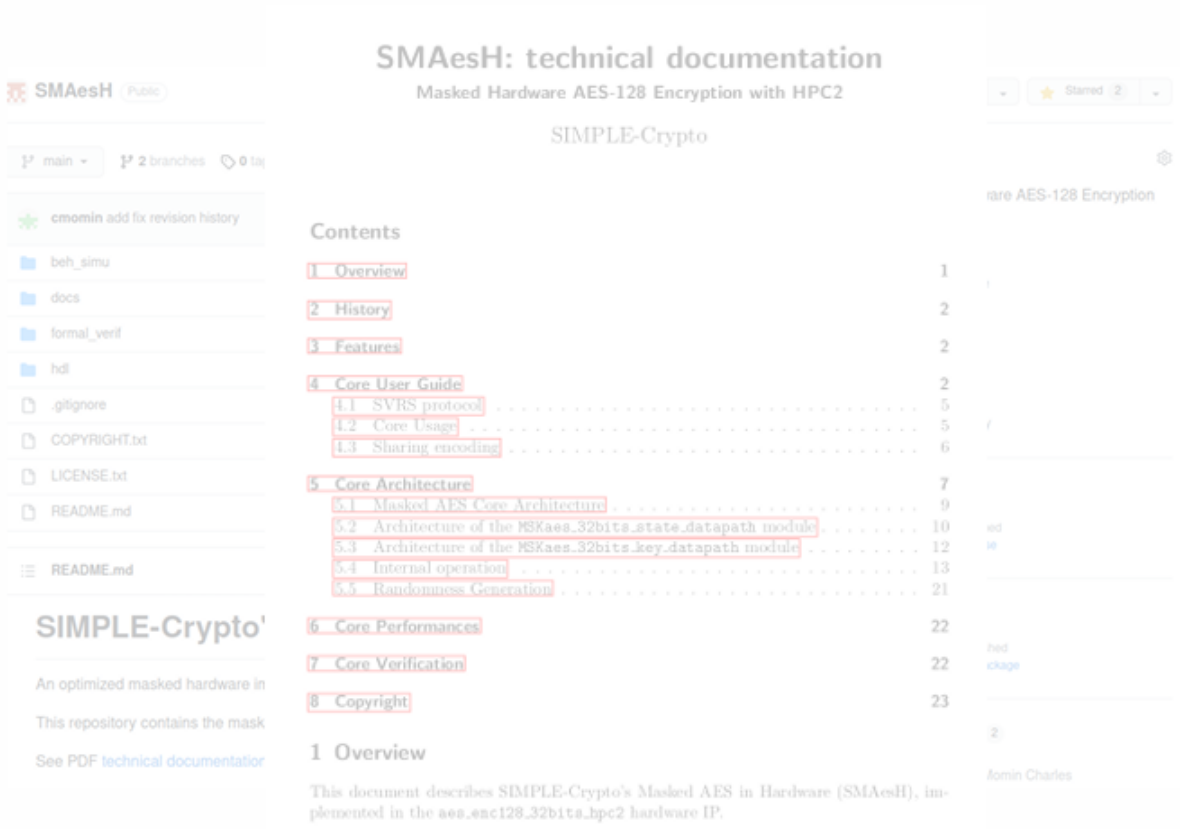
Starred 2

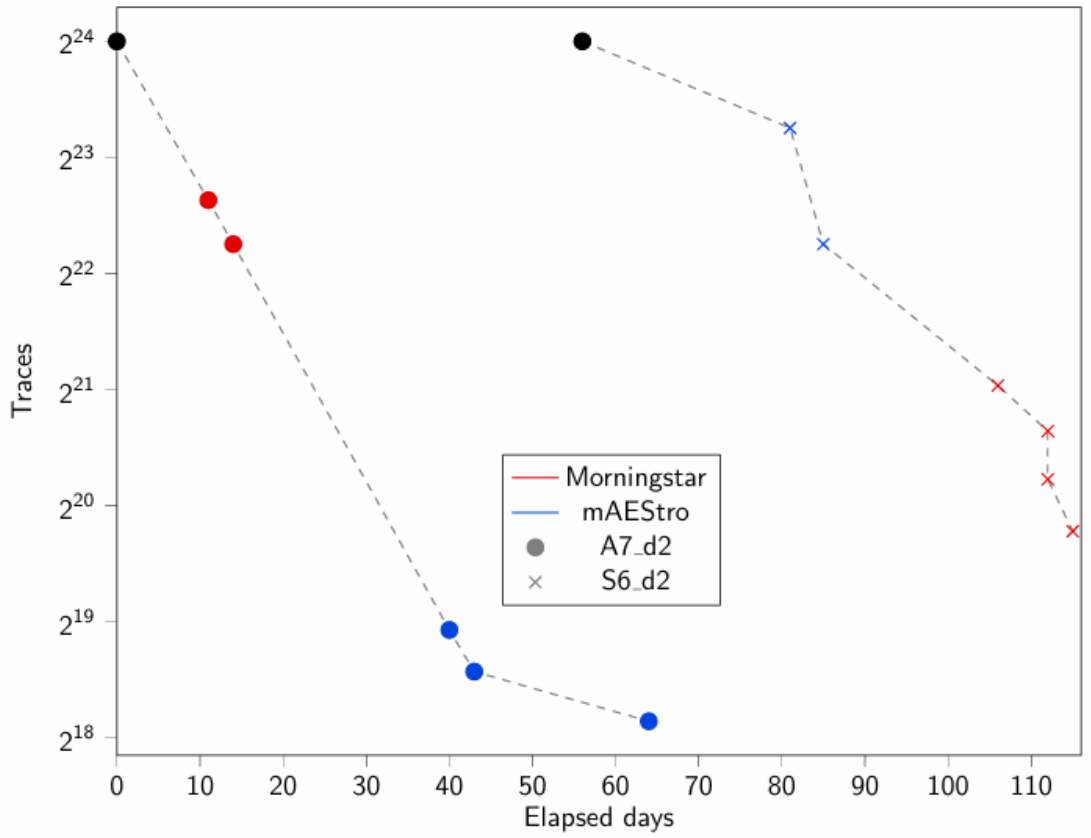
rare AES-128 Encryption

Romin Charles

Contributions (I): SMAEsH & Challenge

- Hardware AES implementation masked at arbitrary orders



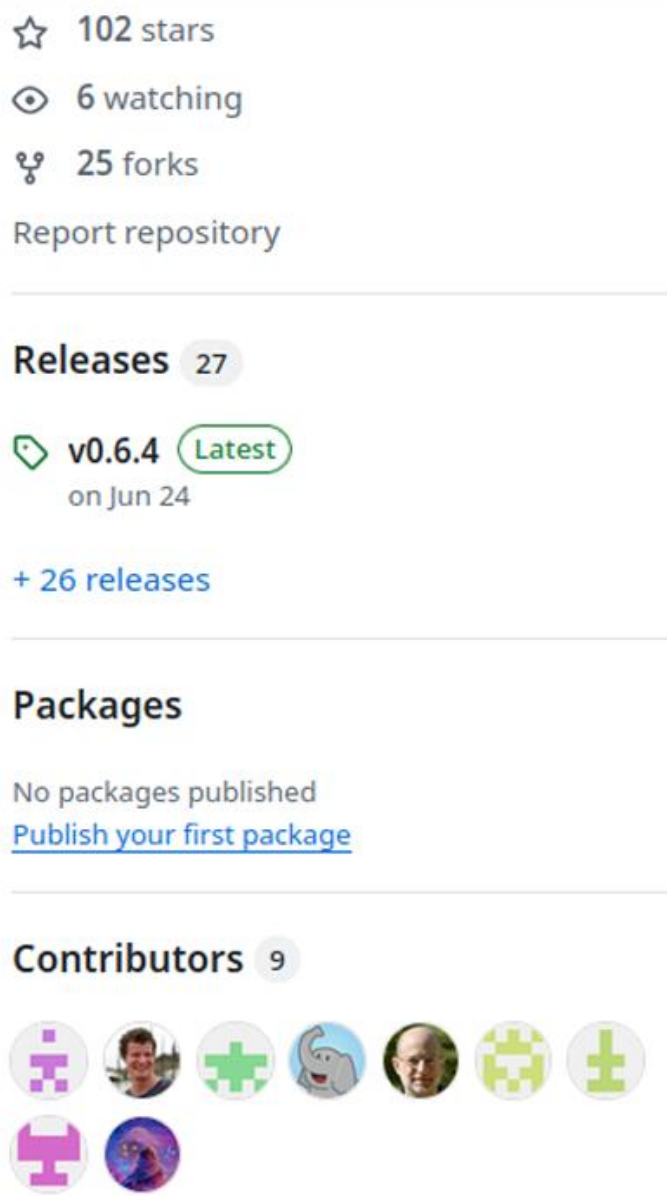


- Comparing worst-case profiling vs. unknown randomness

Contributions (II): SCALib evaluation library

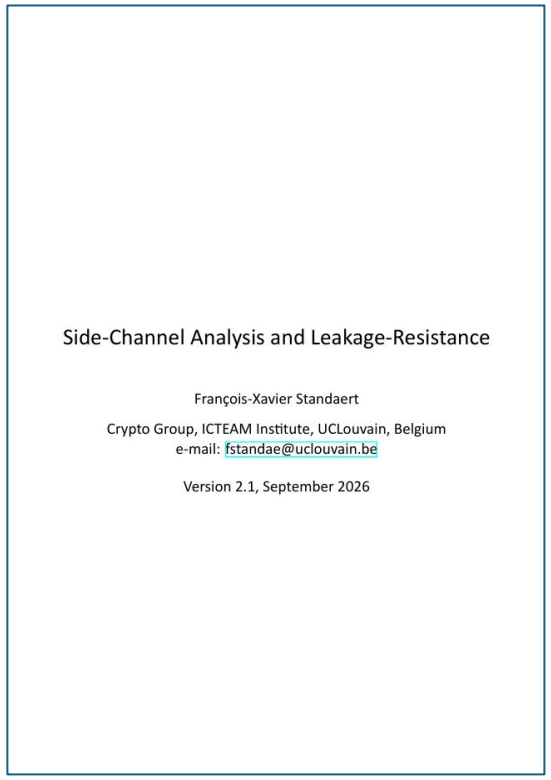
- T-test, SNR, CPA, TA, (R)LDA, SASCA, BP, histogram-based rank-estimation, …
  - Back-end: optimized Rust
  - Font-end: Python
- Competitive with all libraries we know

- Regularly expanding (new functions)
- Fast increasing number of users ☺
- Slower increase of # of contributors ☺



Contributions (III): SCALE trainings

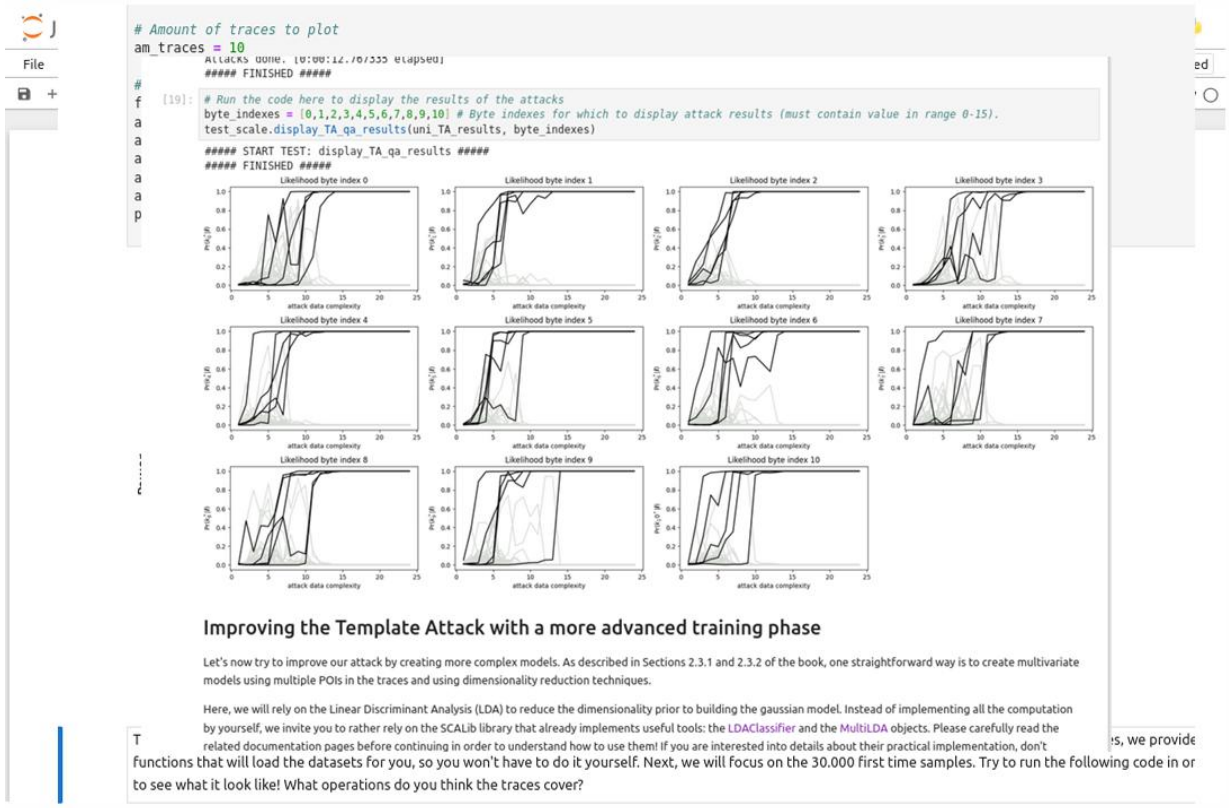
- Book: https://perso.uclouvain.be/fstandae/book.html


Side-Channel Analysis and Leakage-Resistance

François-Xavier Standaert

Crypto Group, ICTEAM Institute, UCLouvain, Belgium
e-mail: fstandae@uclouvain.be

Version 2.1, September 2026



bridging abstraction levels

- Hands on: https://github.com/simple-crypto/scale-one

Coming next

- Hardware SHA3 implem. masked at arbitrary orders (ML-DSA)

Coming next

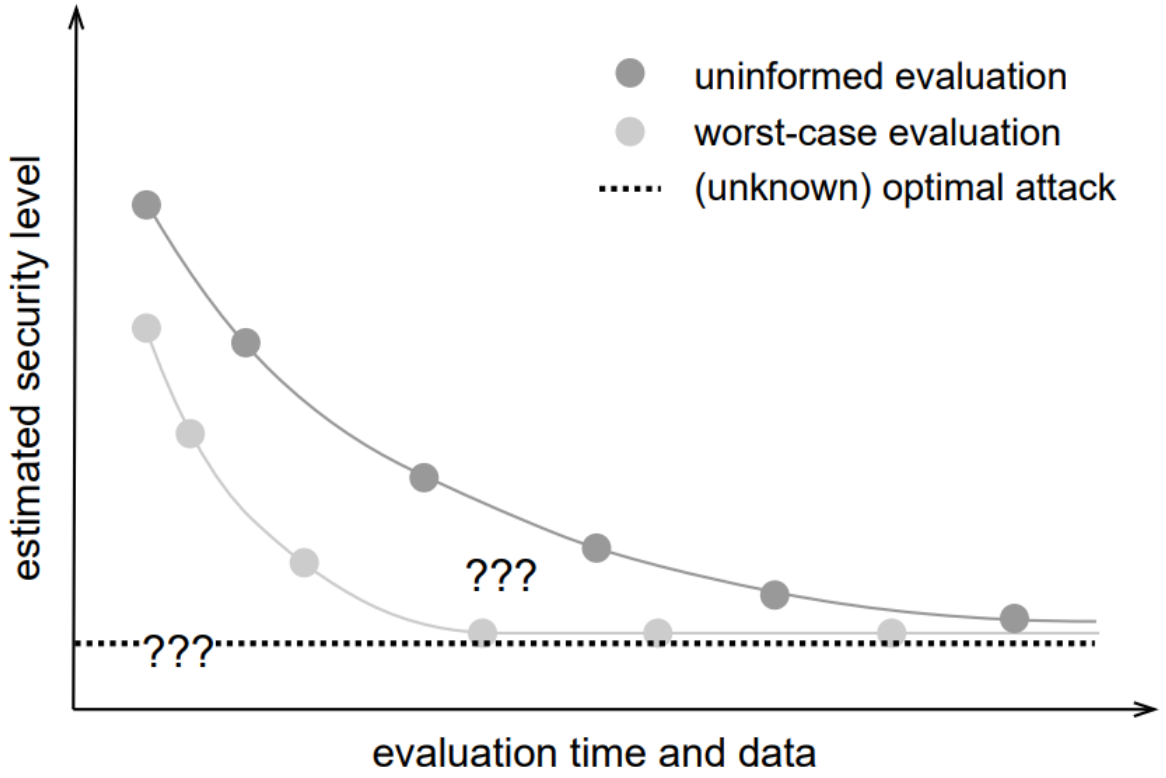
- Hardware SHA3 implem. masked at arbitrary orders (ML-DSA)
-

Level	Transparency	Description	Representative Examples
0	Fully proprietary	Hardware, firmware, and software are closed source. Public security evaluation relies on black-box testing and the vendor's assurances only.	Commercial secure elements or microcontrollers from NXP, Infineon, STMicroelectronics.
1	Open soft-/firmware	Some software or firmware components are open source, while the underlying hardware and microarchitecture remain proprietary.	Hardware wallets from Ledger[1] or Trezor[2], the Raspberry Pi RP2350[3].
2	Open front-end design	Hardware and software source codes are public. Back-end implementation artifacts (e.g., physical layouts) remain confidential.	OpenTitan[4], Caliptra[5], Tropic Square[6], Baochip[7], Pavona[8], open hardware IP on FPGAs (e.g., SMAeSH)[9].
3	Open front- & back-end design	Source codes and physical implementation artifacts are public or can be reproduced, including the netlists, the layouts and the flow scripts.	HEP-Alliance[10], Cloneless1[11].

[1] https://github.com/LedgerHQ
[2] https://github.com/trezor
[3] https://github.com/raspberrypi/rp2350_hacking_challenge
https://github.com/raspberrypi/rp2350_hacking_challenge_2
[4] https://github.com/lowRISC/opentitan
[5] https://github.com/chipsalliance/Caliptra
[6] https://github.com/tropicsquare
[7] https://github.com/baochip/baochip-1x
[8] https://github.com/pavona/pavona
[9] https://github.com/simple-crypto/SMAesH
[10] https://github.com/HEP-Alliance
[11] https://github.com/ThorbenMoos/Cloneless1

Conclusions – lessons learned

- Open-source helps evaluation (no reverse engineering, legal issues, …)
  - Does it lead to the “objective” security level?
  - Is there a (theoretical) gap with closed-source attacks



Crypto analogy: precomputation in security reductions?

ML analogy: gap btw supervised and unsupervised learning?

Conclusions – lessons learned

- Open-source helps evaluation (no reverse engineering, legal issues, …)
  - Does it lead to the “objective” security level?
  - Is there a (theoretical) gap with closed-source attacks
- Even if answer is Y/N, not instantaneously solving everything
  - Needs scientifically-mature solutions (PQ crypto?)
  - Open-source code ≠ open silicon (main challenge?)
- Today (opinion): it already helps security more than it hurts it
- Not UCLouvain-specific (open to / can fund external developers)

https://www.simple-crypto.org/