Application outillée de contre-mesures contre les attaques matérielles

Soutenance d'Habilitation à Diriger des Recherches

Damien Couroussé

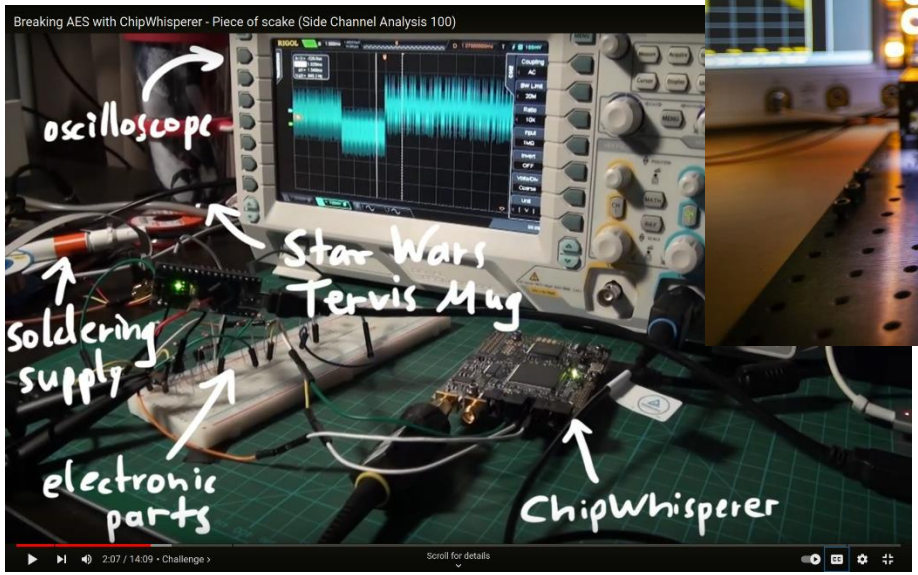
28 août 2024

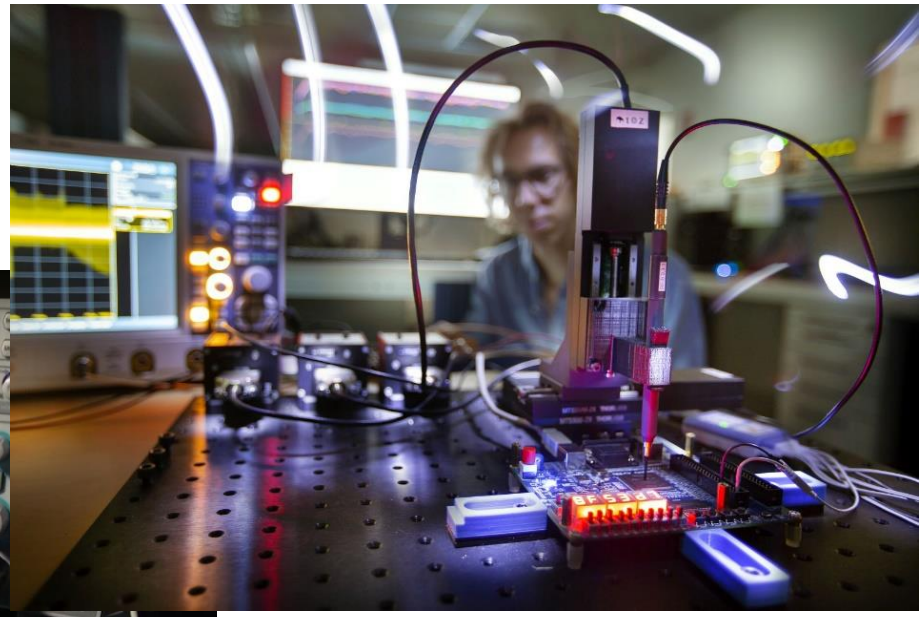
Composition du jury

Rapporteurs	**Guy Gogniat**	professeur à l'Université Bretagne Sud
	Guillaume Hiet	professeur à CentraleSupélec Rennes
	Ingrid Verbauwhede	professeure à KU Leuven
Examinateurs	**Aurélien Francillon**	professeur à EURECOM
	David Hély	professeur à LCIS, Université Grenoble Alpes
	Karine Heydemann	experte sécurité à Thalès DIS, chercheuse associée au LIP6
	Marie-Laure Potet	professeure à Université Grenoble Alpes

Cybersecurity: a challenge for the information society

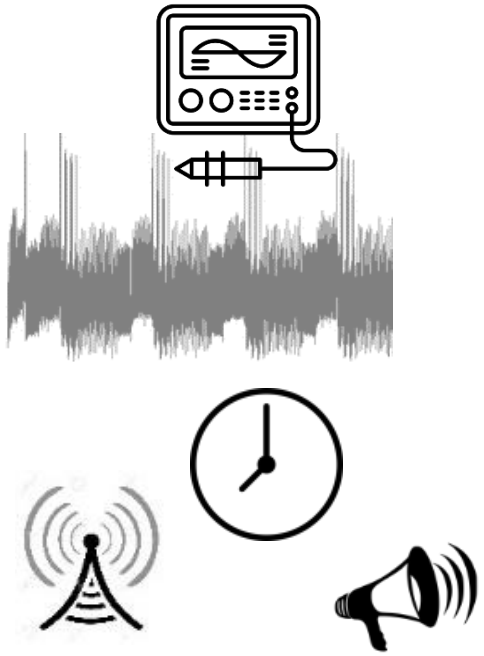
Once there were two "mental chess" experts who had become tired of their pastime. "Let's play 'Mental Poker,' for variety" suggested one. "Sure" said the other. "Just let me deal!"

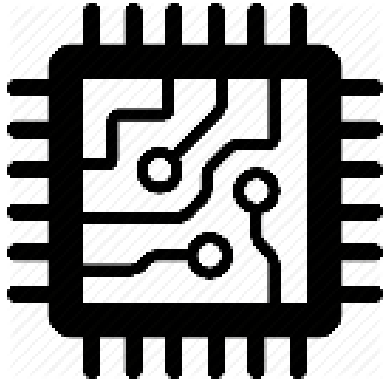




Physical attacks

Side-channel analysis





Fault injection attacks

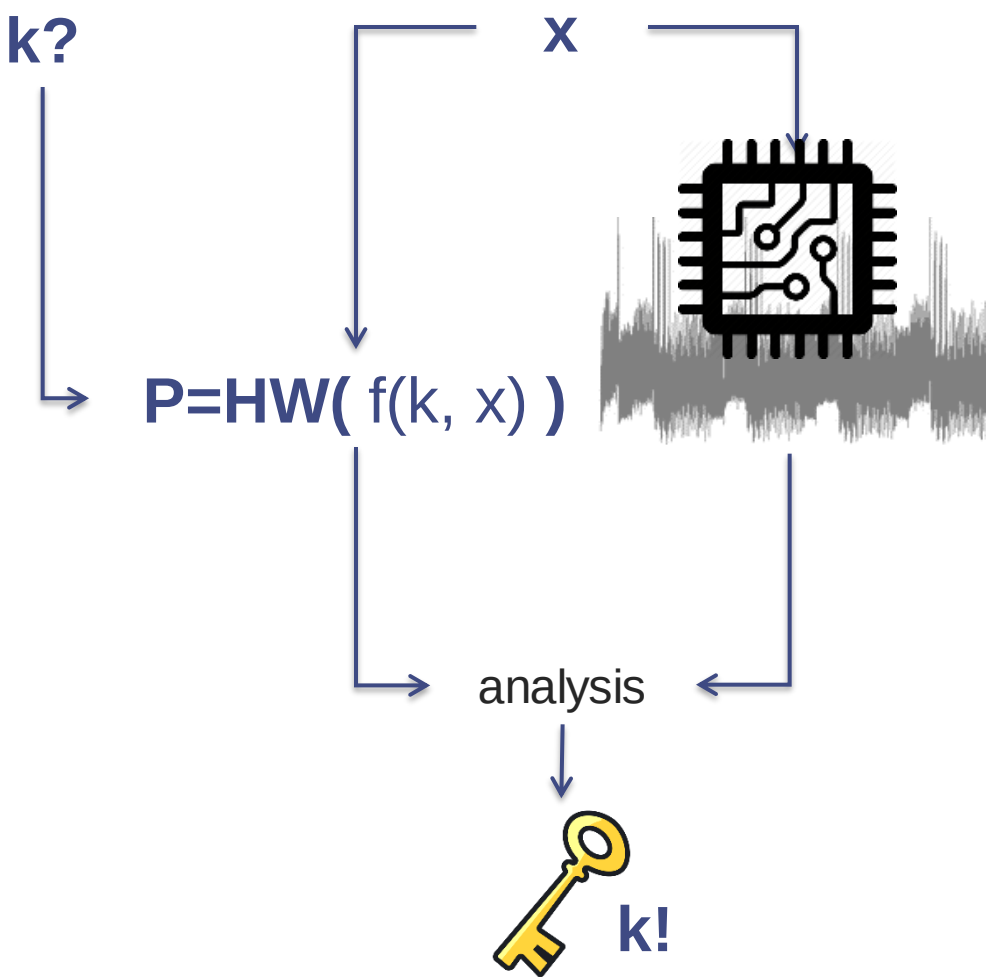
Highly effective against cryptographic implementations

Can leverage software vulnerabilities [Cui & Rousley, 2017]

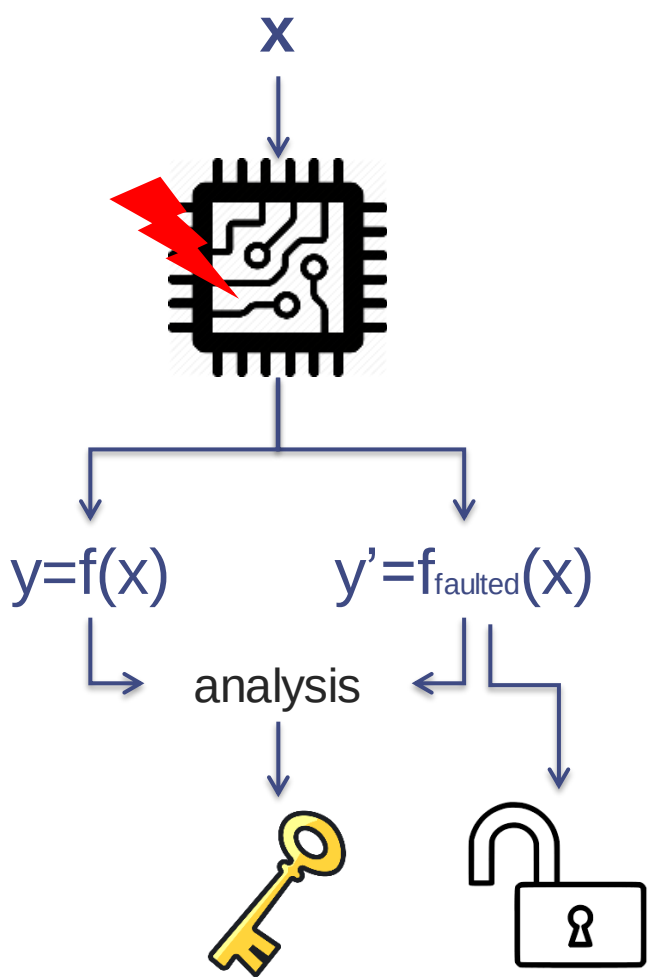
[Cui & Rousley, 2017] Defeating Modern Secure Boot Using Second-Order Pulsed Electromagnetic Fault Injection. 10.5555/3154768.3154771

Physical attacks, conceptually

Side-channel analysis

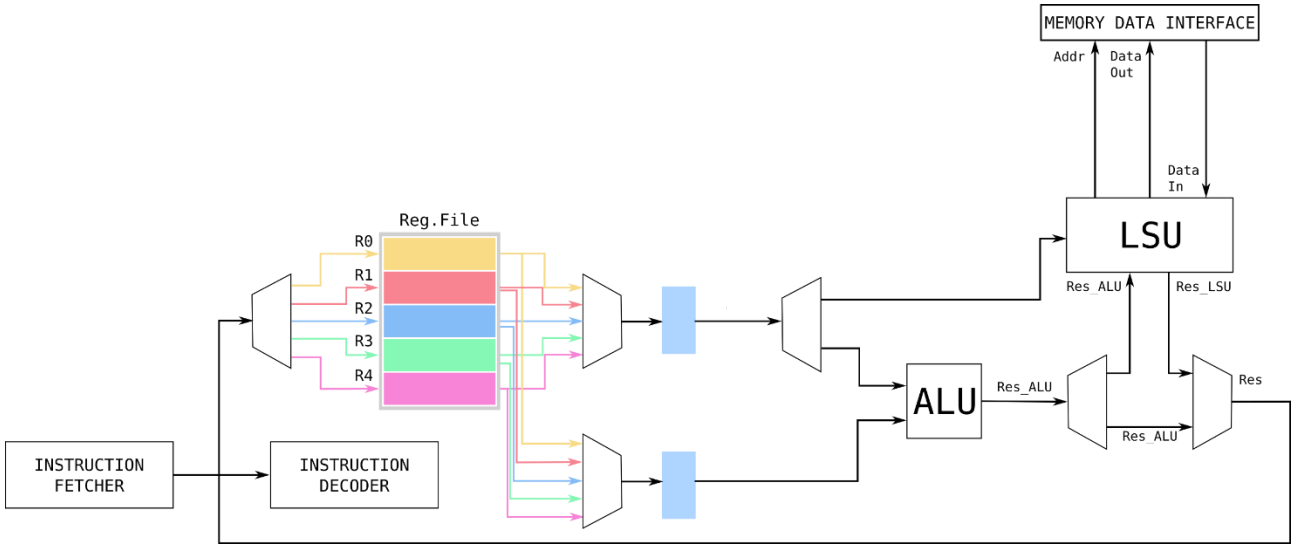


Fault injection attacks



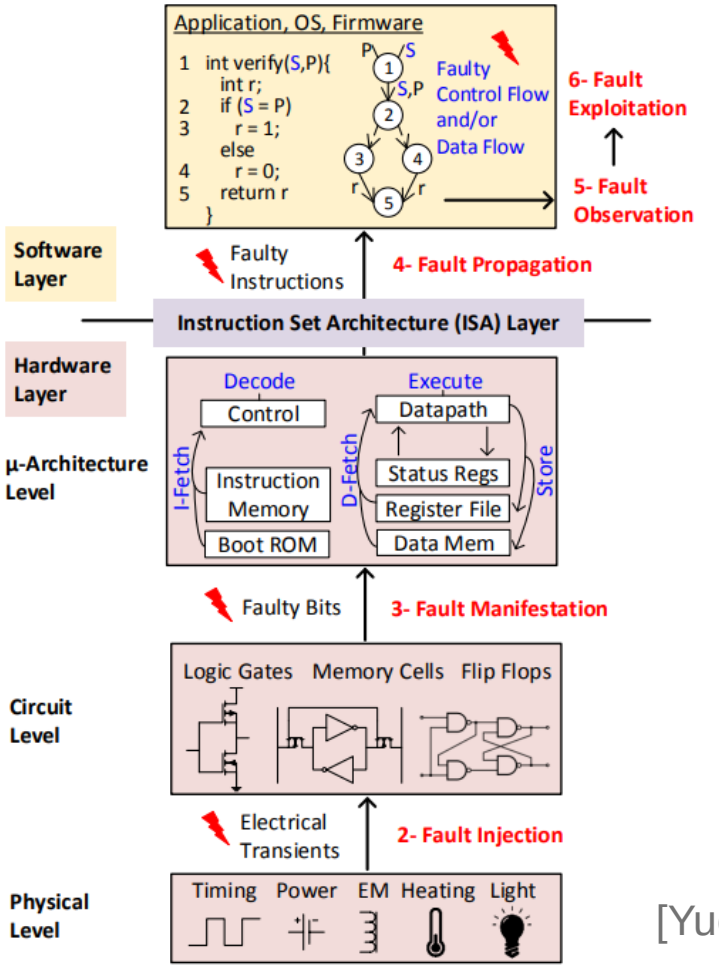
Physical attacks, behind the scenes

Side-channel analysis



[Casalino PhD, 2024]

Fault injection attacks



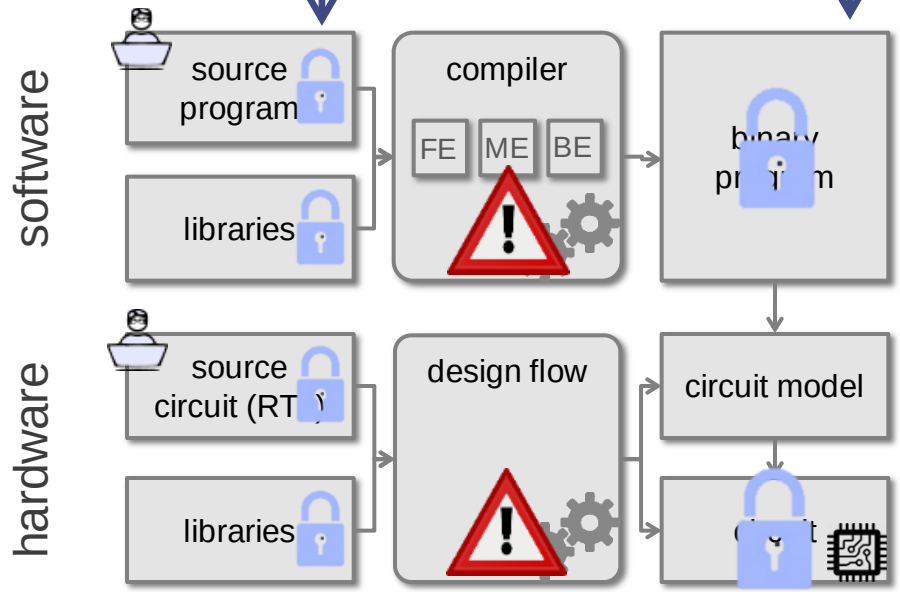
[Yuce & al. 2018]

[Yuce & al. 2018] Fault Attacks on Secure Embedded Software: Threats, Design and Evaluation. 10.1007/s41635-018-0038-1
[Casalino PhD, 2024] (On) The Impact of the Micro-architecture on Countermeasures against Side-Channel Attacks. HAL:tel-04573949

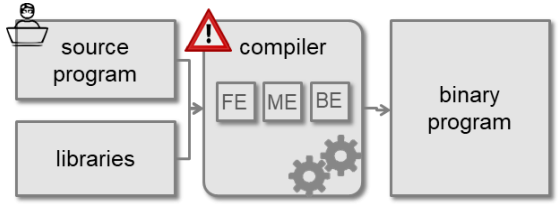
Towards the production of secure digital systems

countermeasures are applied at every abstraction level, from software to silicon

effective security in the implementation



(Legacy) compilers are not designed for security



No support for FIA / SCA countermeasures

- Software security: support of control-flow protections such as stack canaries, etc. (GCC, LLVM)

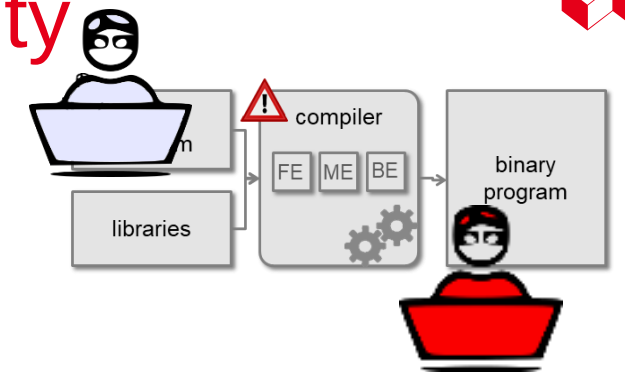
Compiler optimizations are harmful to countermeasures

- Redundancy / dead code elimination (FIA)
- Inversion of commutative operations (masking vs. SCA)

Compile without optimizations (-O0)? Not even better!

- Poor performance
- Large attack surface: code size, high nb of mem accesses

(Legacy) compilers are not designed for security



No support for FIA / SCA countermeasures

- Software security: support of control-flow protections such as stack canaries, etc. (GCC, LLVM)

Compiler optimizations are harmful to countermeasures

- Redundancy / dead code elimination (FIA)
- Inversion of commutative operations (masking vs. SCA)

Compile without optimizations (-O0)? Not even better!

- Poor performance
- Large attack surface: code size, high nb of mem accesses

Consequences

- **High manual effort** for hardening implementations
- **Fragile implementations** (compiler tricks, recipes, workarounds…)
- **Requires careful inspection** of compiler outputs

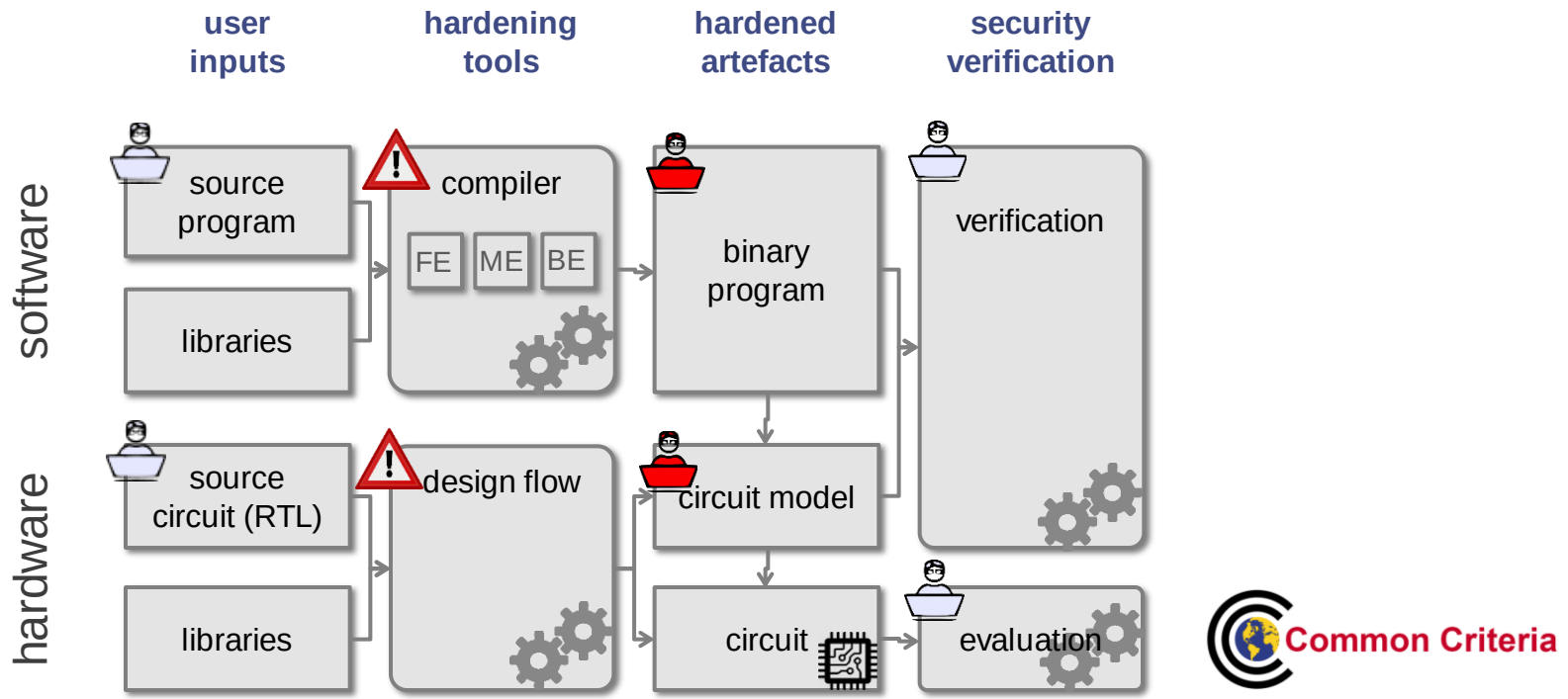
Research question:
Provide tools to help secure digital systems

→ **Application of countermeasures**

- Automation
- Flexibility: security/cost trade-off
- Effective at the binary level

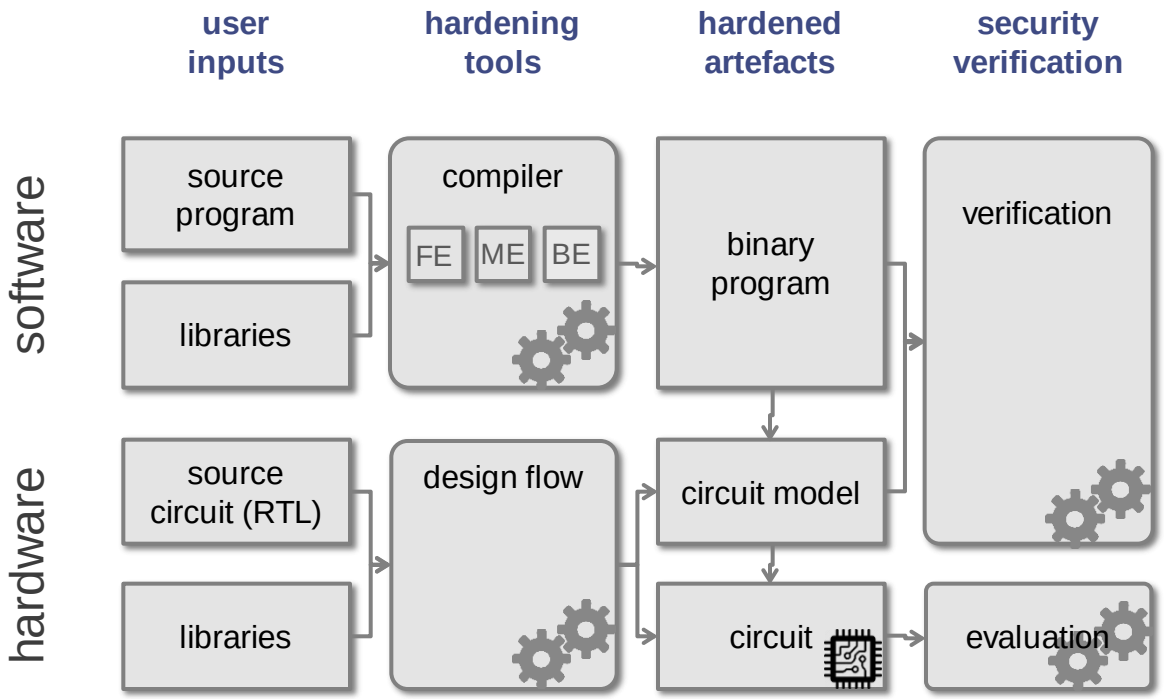
Same bottlenecks with synthesis tools, e.g. [Nasahl & al., 2022]

Towards the production of secure digital systems



Towards the production of secure digital systems

Research question:
Provide tools to help secure digital systems



→ **Application of countermeasures**

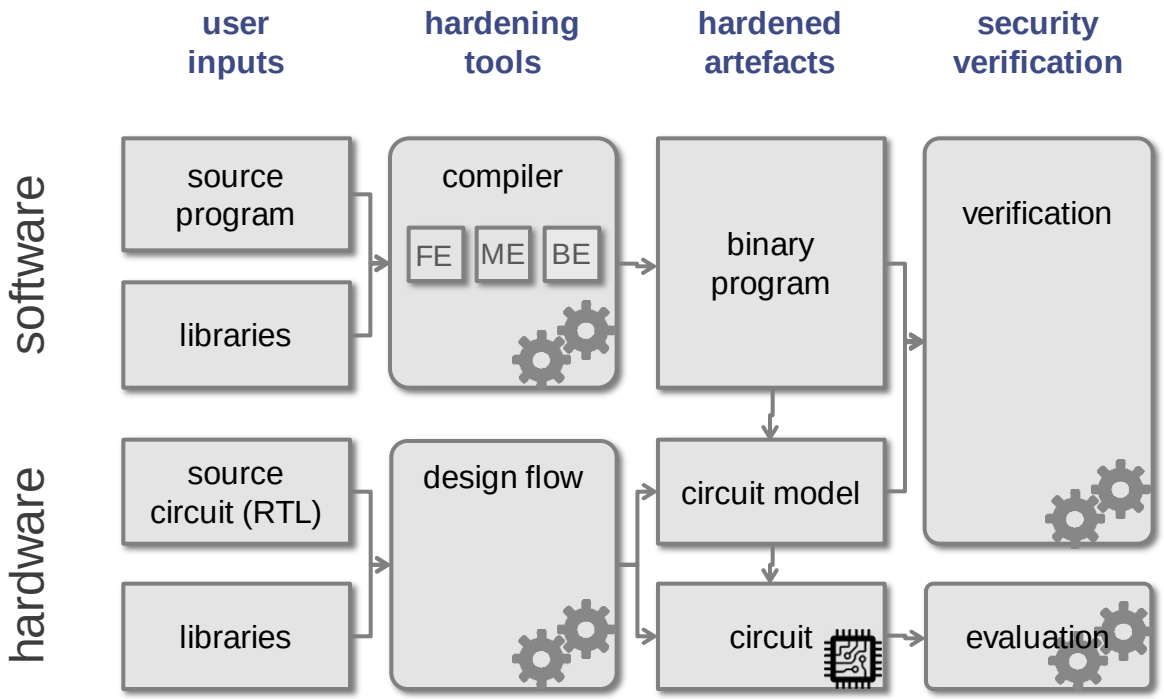
- Automation
- Flexibility: security/cost trade-off
- Effective at the binary level

→ **Security verification**

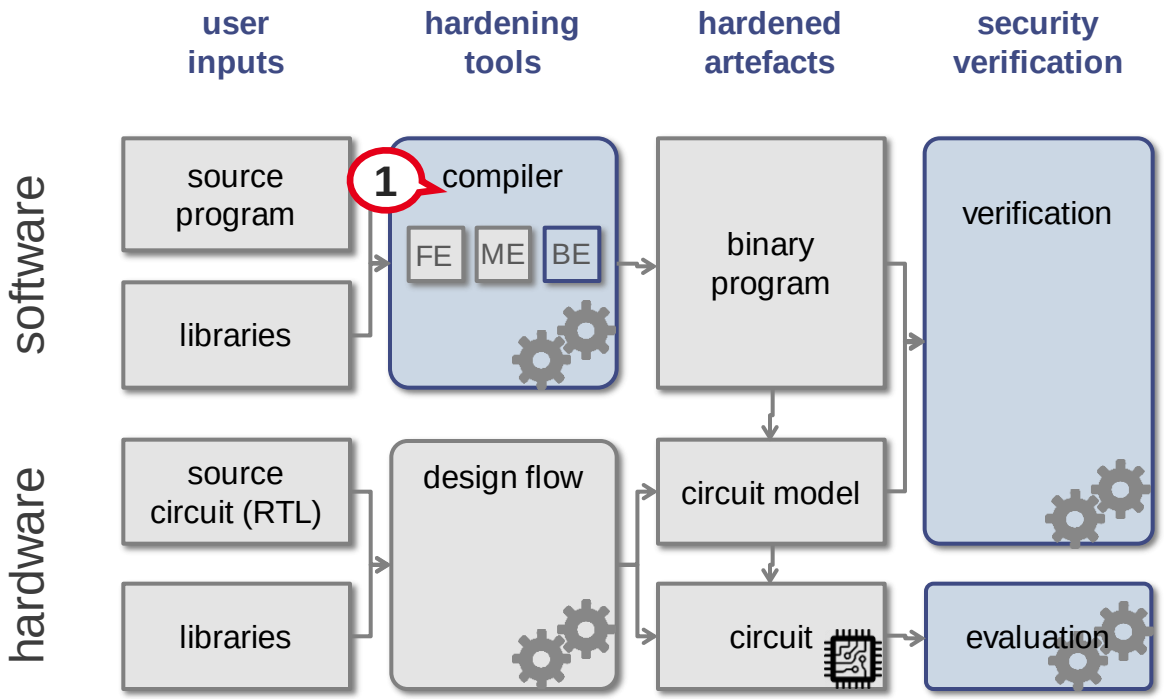
- Automation
- Flexibility: variable threat models
- SW level, HW level, **SW+HW**

Research overview

Research overview



Research overview

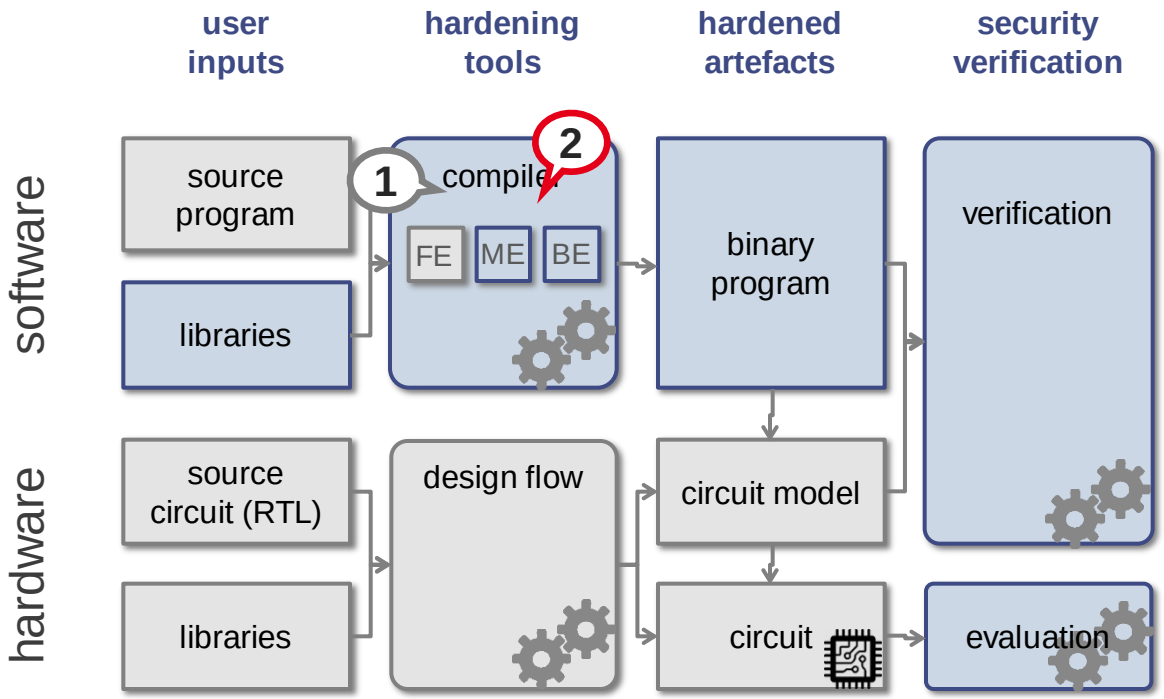


Compilation of software countermeasures

1. Against fault injection attacks
   - Fault tolerance against instruction-skip fault attacks [CS2, 2016]
   - Control-flow integrity [Barry PhD, 2017]

[CS2, 2016] Compilation of a Countermeasure against Instruction-Skip Fault Attacks. 10.1145/2858930.2858931
[Barry PhD, 2017] Sécurisation à la compilation de logiciels contre les attaques en fautes. HAL:tel-01783995v2

Research overview

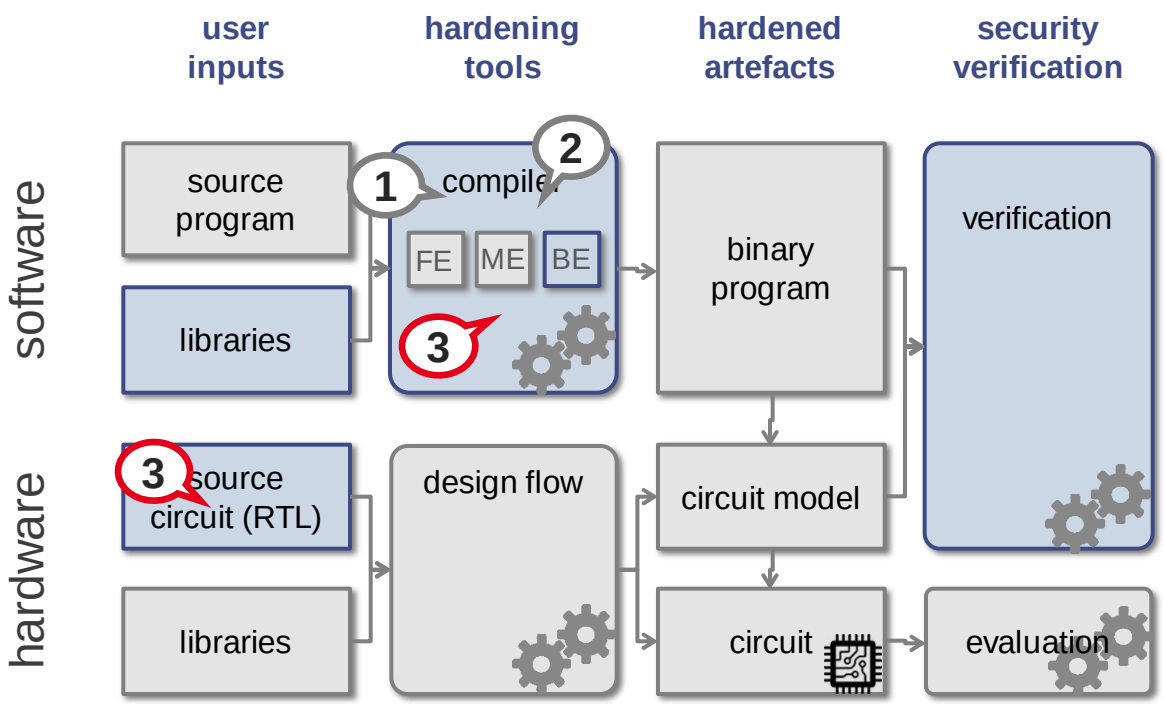


Compilation of software countermeasures

1. Against fault injection attacks
   - Fault tolerance against instruction-skip fault attacks [CS2, 2016]
   - Control-flow integrity against fault injection attacks [Barry PhD, 2017]
2. Against side-channel attacks
   - Code polymorphism as a hiding countermeasure [WISTP, 2016] [TACO, 2019]
   - First-order Boolean masking [TCAD, 2020]

[WISTP, 2016] Runtime Code Polymorphism as a Protection against Side Channel Attacks. 10.1007/978-3-319-45931-8_9
[TACO, 2019] Automated Software Protection for the Masses against Side-Channel Attacks. 10.1145/3281662
[TCAD, 2020] Maskara: Compilation of a Masking Countermeasure with Optimised Polynomial Interpolation. 10.1109/TCAD.2020.3012237

Research overview



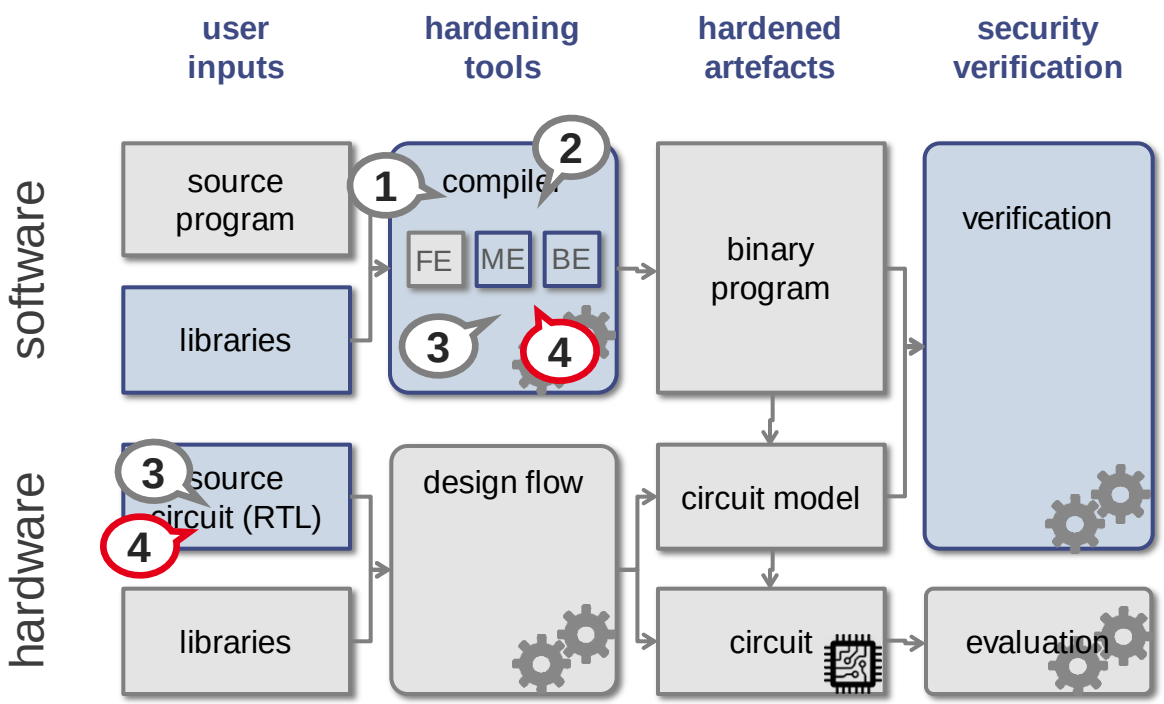
Compilation of software countermeasures

1. Against fault injection attacks [CS2, 2016] [Barry PhD, 2017]
2. Against side-channel attacks [WISTP, 2016] [TACO, 2019] [TCAD, 2020]

Hardware-software co-design of secure systems

3. Confidentiality and side-channel protection [DTRAP, 2022]

Research overview



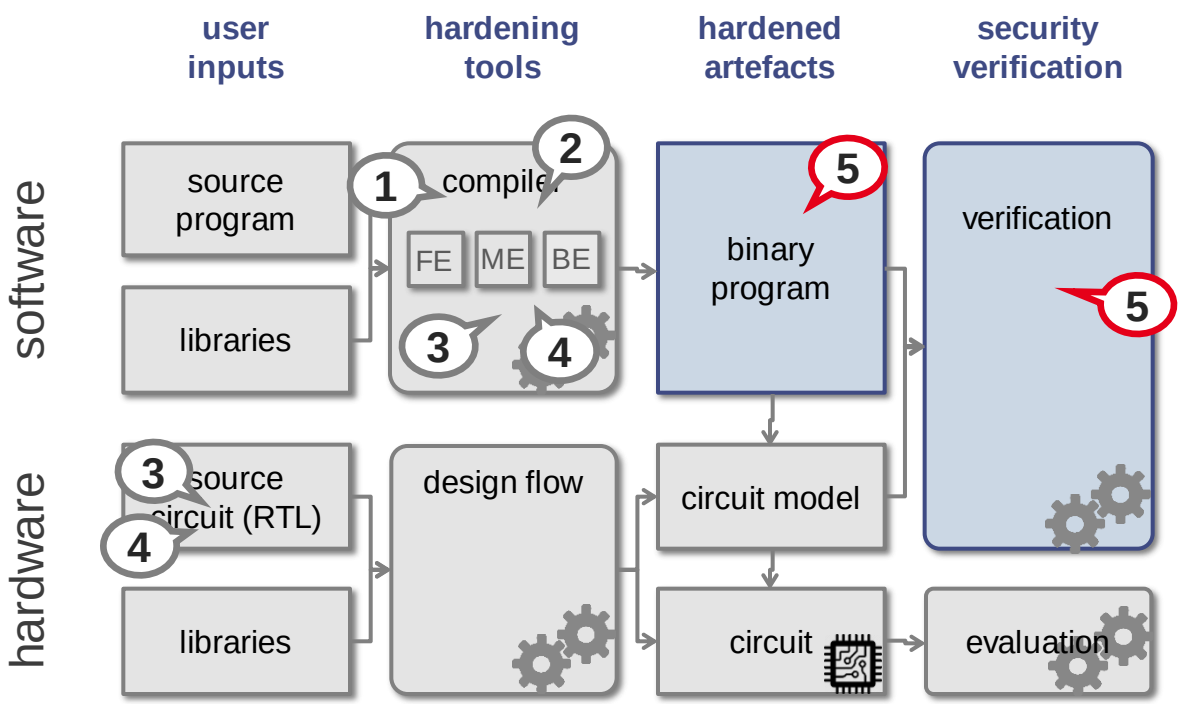
Compilation of software countermeasures

1. Against fault injection attacks [CS2, 2016] [Barry PhD, 2017]
2. Against side-channel attacks [WISTP, 2016] [TACO, 2019] [TCAD, 2020]

Hardware-software co-design of secure systems

3. Confidentiality and side-channel protection [DTRAP, 2022]
4. Control-signal integrity of a processor microarchitecture [TCAD, 2023]

Research overview



Compilation of software countermeasures

1. Against fault injection attacks [CS2, 2016] [Barry PhD, 2017]
2. Against side-channel attacks [WISTP, 2016] [TACO, 2019] [TCAD, 2020]

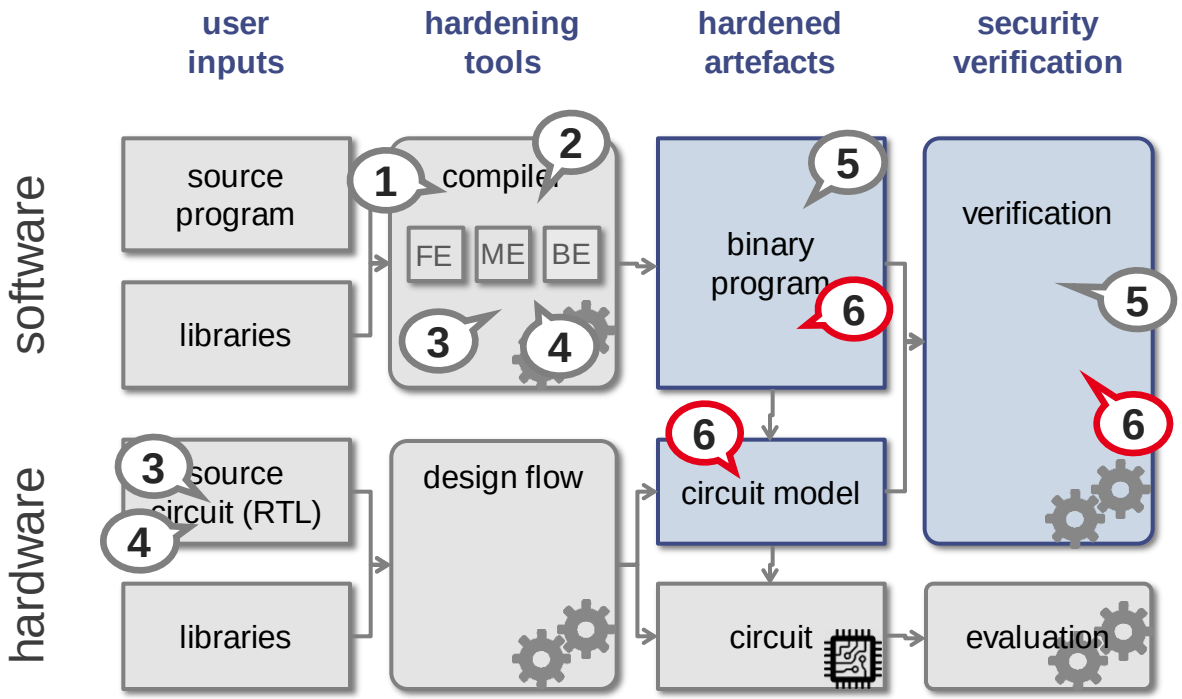
Hardware-software co-design of secure systems

3. Confidentiality and side-channel protection [DTRAP, 2022]
4. Control-signal integrity of a processor microarchitecture [TCAD, 2023]

Security formal verification

5. Characterization of software vulnerabilities [POPL, 2024]
   - Application to fault injection attacks

Research overview



Compilation of software countermeasures

1. Against fault injection attacks [CS2, 2016] [Barry PhD, 2017]
2. Against side-channel attacks [WISTP, 2016] [TACO, 2019] [TCAD, 2020]

Hardware-software co-design of secure systems

3. Confidentiality and side-channel protection [DTRAP, 2022]
4. Control-signal integrity of a processor microarchitecture [TCAD, 2023]

Security formal verification

5. Characterization of software vulnerabilities [POPL, 2024]
6. Verification of processor microarchitecture for fault robustness HW+SW analysis [FDTC, 2022] [TCHES, 2024]

[FDTC, 2022] Exploration of Fault Effects on Formal RISC-V Microarchitecture Models. 10.1109/FDTC57191.2022.00017
[TCHES, 2024] Fault-Resistant Partitioning of Secure CPUs for System Co-Verification against Faults. eprint.iacr:2024/247

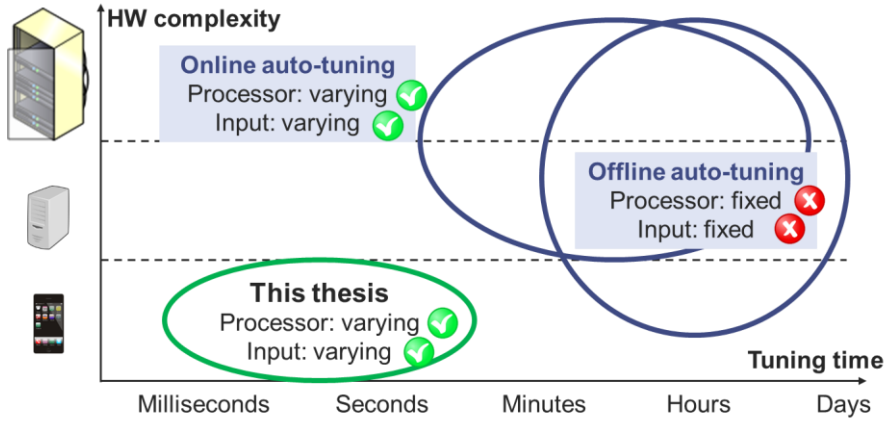
Compilation of Software Countermeasures

Detour: Runtime Code Generation for Embedded Systems

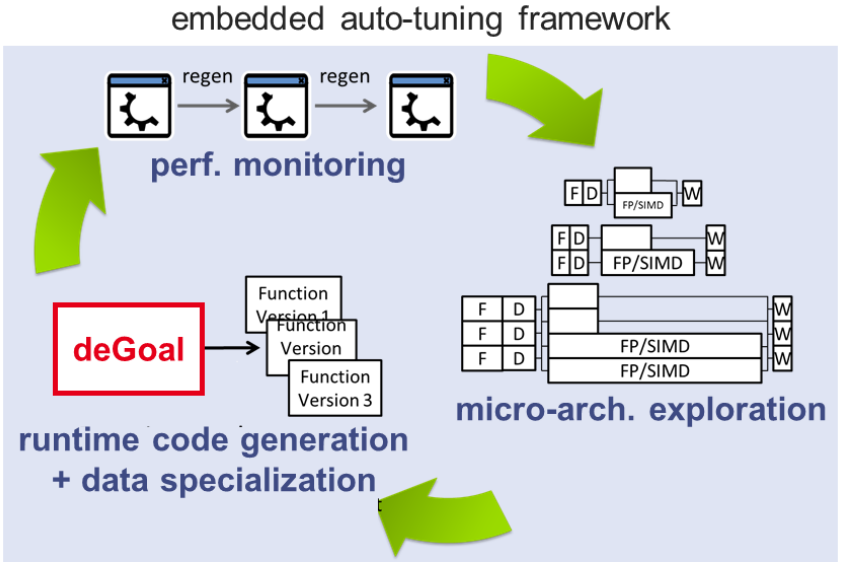
PhD. Fernando Endo, 2015

Architecture- and data-driven code specialization for time and energy efficiency

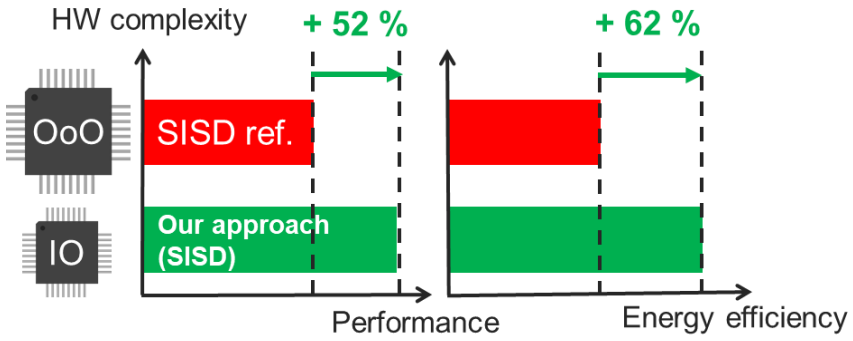
Challenge

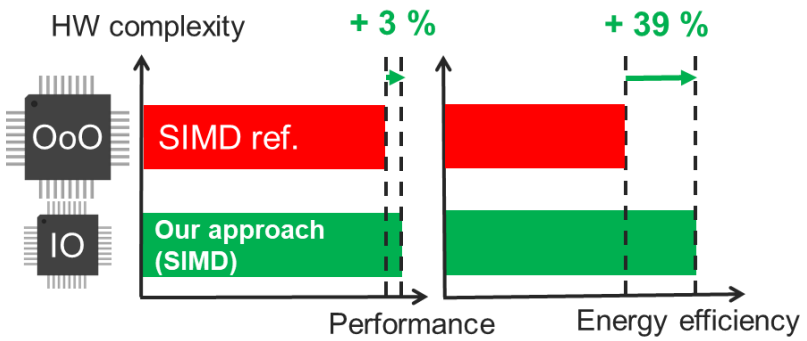


Approach



Results





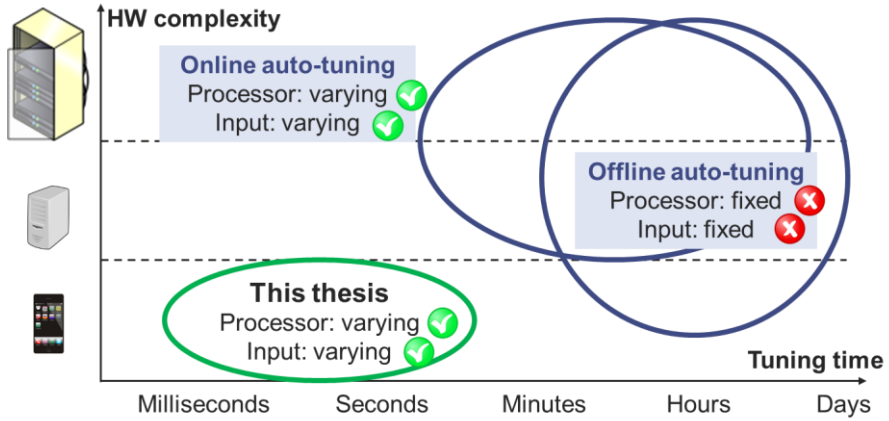
[Endo PhD, 2015] Génération dynamique de code pour l'optimisation énergétique. HAL:tel-01285964
[MCSoC, 2016] Pushing the Limits of Online Auto-Tuning: Machine Code Optimization in Short-Running Kernels. arXiv:1707.04566

Detour: Runtime Code Generation for Embedded Systems

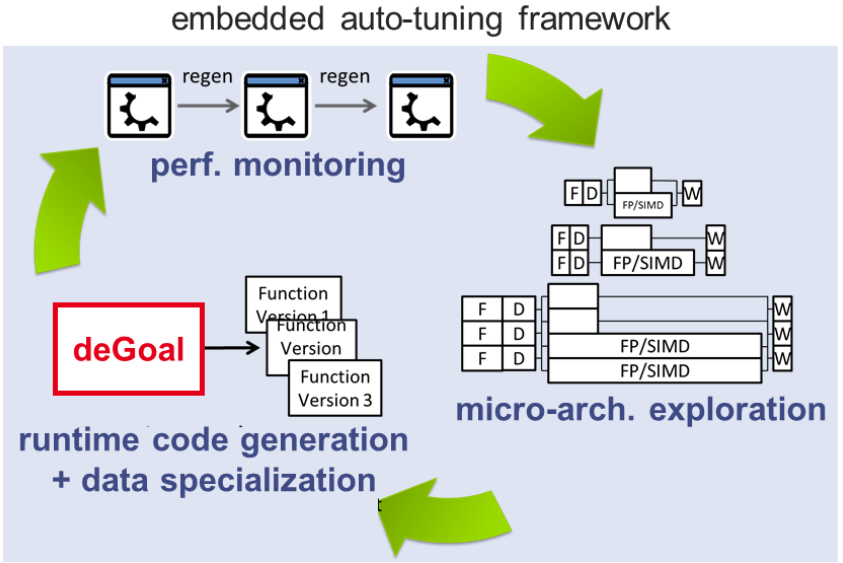
PhD. Fernando Endo, 2015

Architecture- and data-driven code specialization for time and energy efficiency

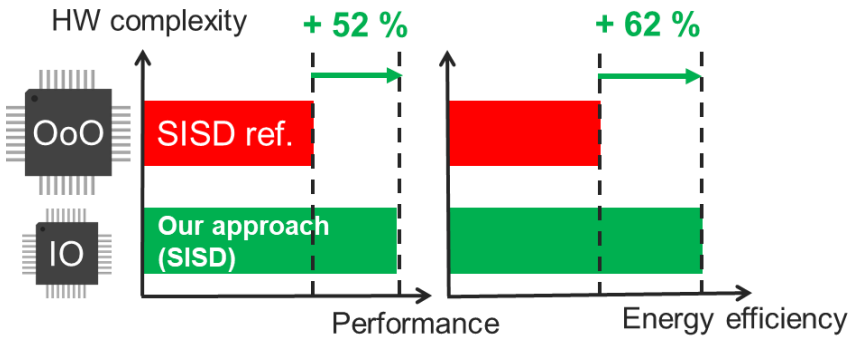
Challenge

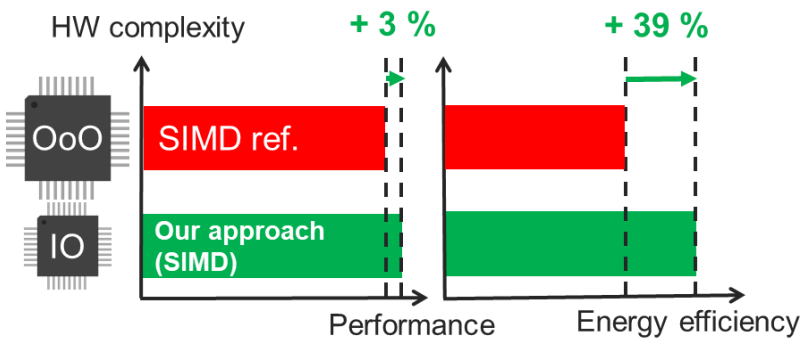


Approach



Results





[Endo PhD, 2015] Génération dynamique de code pour l'optimisation énergétique. HAL:tel-01285964

[MCSoC, 2016] Pushing the Limits of Online Auto-Tuning: Machine Code Optimization in Short-Running Kernels. arXiv:1707.04566

Code Polymorphism

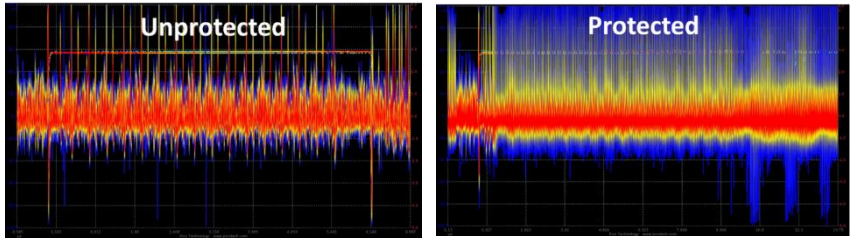
COGITO, ANR 2013-2017

PhD. Nicolas Belleville, 2019

Regularly changing the behavior of a software component, at runtime, while preserving functional properties

Challenges:

High observational variability



Applicability to constrained embedded systems

Code Polymorphism

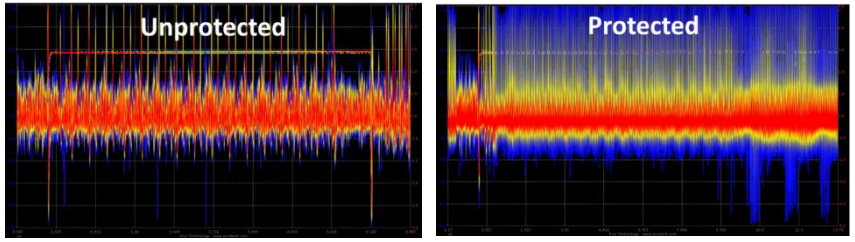
COGITO, ANR 2013-2017

PhD. Nicolas Belleville, 2019

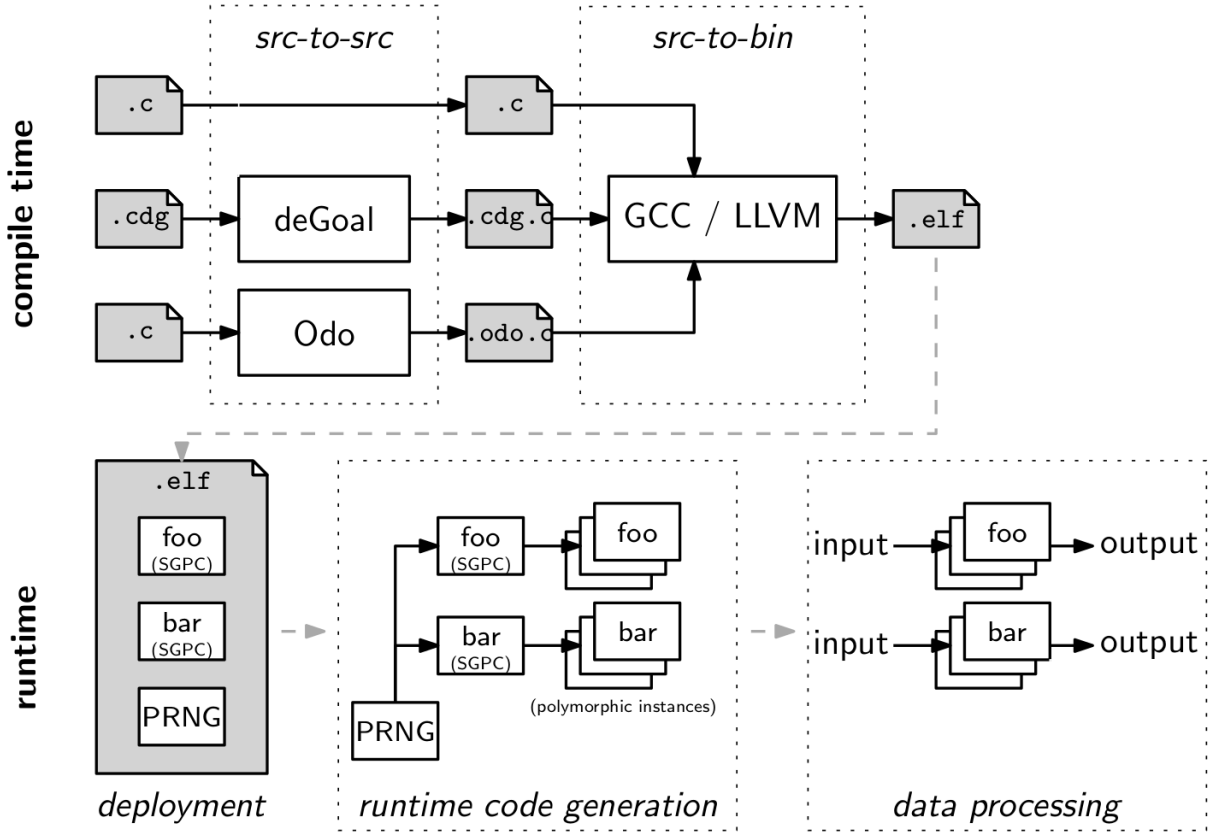
Regularly changing the behavior of a software component, at runtime, while preserving functional properties

Challenges:

High observational variability



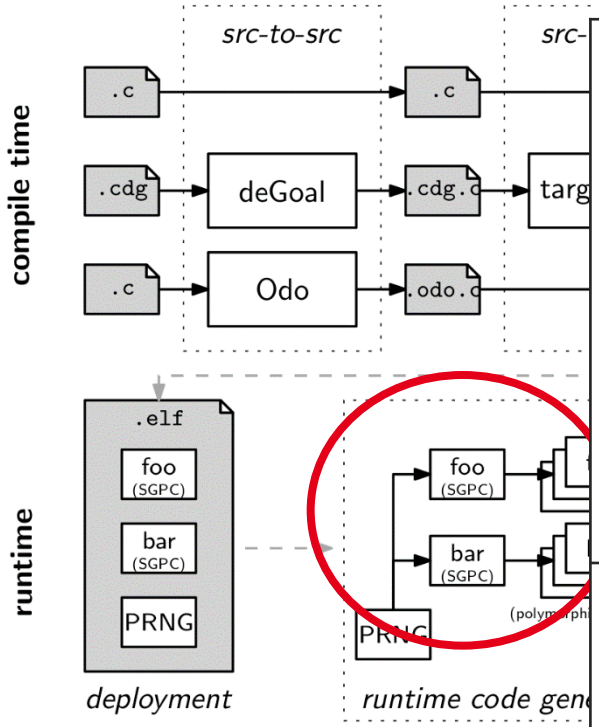
Applicability to constrained embedded systems



Runtime Code Transformations

COGITO, ANR 2013-2017

PhD. Nicolas Belleville, 2019




```
add r4, r4, r5
xor r6, r5, r8
```

Register shuffling

random general purpose register permutation

r4 → **r11**

...

```
add r11, r11, r7
xor r8, r7, r5
```

Instruction shuffling

independent instructions are emitted in a **random** order

```
xor r6, r5, r8
add r4, r4, r5
```

Semantic variants

replacement of an instruction by a **randomly** selected semantic variant

```
add r4, r4, r5
xor r6, r5, #12348
xor r6, r6, r8
xor r6, r6, #12348
```

Noise instructions

insertion of a **random** number of **randomly** chosen useless instructions

```
add r4, r4, r5
sub r7, r6, r2
load r3, r10, #53
xor r6, r5, r8
```

Dynamic noise

random execution of noise instructions

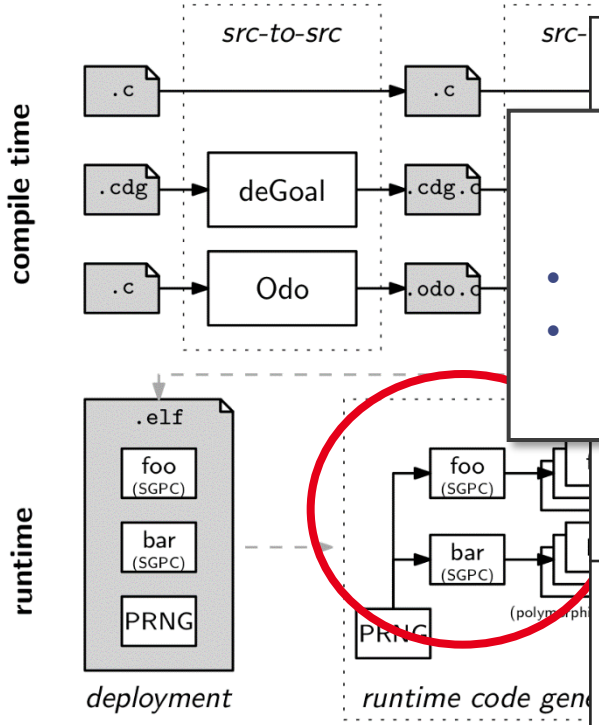
```
add r4, r4, r5
jump ?
sub r7, r6, r2
load r3, r10, #53
xor r6, r5, r8
```

[TACO, 2019] Automated Software Protection for the Masses against Side-Channel Attacks. 10.1145/3281662

Runtime Code Transformations

COGITO, ANR 2013-2017

PhD. Nicolas Belleville, 2019



Huge number of possible combinations:

- 10 original machine instructions → 6.10^{22}
- **AES with 278 machine instructions** → $> \mathbf{10^{274}}$

[TACO, 2019]

	Register shuffling	**Instruction shuffling**
	ndent instructions are d in a **random** order	
	`add r11, r11, r7` `xor r8, r7, r5`	`xor r6, r5, r8` `add r4, r4, r5`
Semantic variants	**Noise instructions**	**Dynamic noise**
replacement of an instruction by a **randomly** selected semantic variant	insertion of a **random** number of **randomly** chos useless instructions	

`xor r6, r5, #12348`
`add r4, r4, r5`
`xor r6, r6, r8`
`xor r6, r6, #12348`
	`add r4, r4, r5`
	`sub r7, r6, r2`
	`xor r6, r5, r8`
	`load r3, r10, #53`
		`sub r7, r6, r2`
		`load r3, r10, #53`
		`xor r6, r5, r8`

Many possible interactions between runtime code transformations

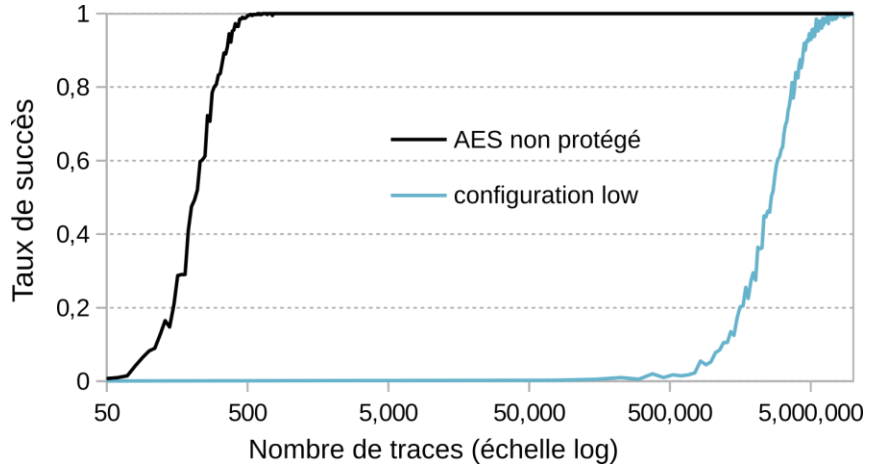
[TACO, 2019] Automated Software Protection for the Masses against Side-Channel Attacks. 10.1145/3281662

Security Evaluation

GreyBox Evaluation, unprotected ARM Cortex-M3

CPA attack, EM, without alignment

- ~ A_{auto} [ESORICS, 2020]



[TACO, 2019]

WhiteBox Evaluation, unprotected ARM Cortex-M4

Several attacks with attacker models of increasing power:

- Automated attack **without alignment** →
- CPA with **manual alignment** →
- Gaussian template with **manual alignment** →
- CNN **designed by expert** →

 # traces for successful leakage exploitation

Scenario	mbedTLS	AES 8-bit
$\mathcal{A}_{\text{auto}}$	$>10^5$	$>\mathbf{10^5}$
$\mathcal{A}_{CPA_{aligned}}$	$3.10^3 - 10^5$	$>\mathbf{10^5}$
$\mathcal{A}_{gT_{aligned}}$	$20 - 10^3$	$>\mathbf{10^5}$
$\mathcal{A}_{\text{CNN}}$	<20	<10

[ESORICS, 2020]

Software hiding with higher noise can be effective against deep learning attacks [Belleville & Masure, 2024]

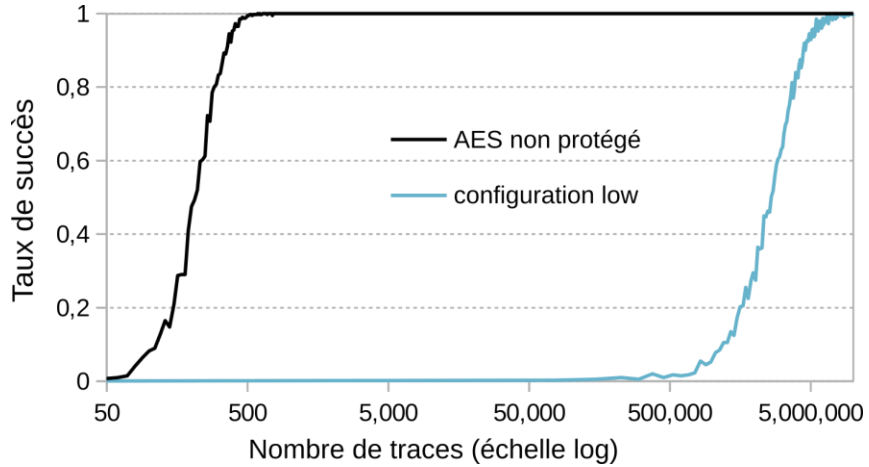
[TACO, 2019] Automated Software Protection for the Masses against Side-Channel Attacks. 10.1145/3281662
[ESORICS, 2020] Deep Learning Side-Channel Analysis on Large-Scale Traces – a Case Study on a Polymorphic AES. 10.1007/978-3-030-58951-6_22
[Belleville & Masure, 2024] Combining Loop Shuffling and Code PolyMorphism for Enhanced AES Side-Channel Security. 10.1007/978-3-031-57543-3_14

Security Evaluation

GreyBox Evaluation, unprotected ARM Cortex-M3

CPA attack, EM, without alignment

- ~ A_{auto} [ESORICS, 2020]



[TACO, 2019]

WhiteBox Evaluation, unprotected ARM Cortex-M4

Several attacks with attacker models of increasing power:

- Automated attack **without alignment** →
- CPA with **manual alignment** →
- Gaussian template with **manual alignment** →
- CNN **designed by expert** →

 # traces for successful leakage exploitation

Scenario	mbedTLS	AES 8-bit
$\mathcal{A}_{\text{auto}}$	$>10^5$	$>\mathbf{10^5}$
$\mathcal{A}_{CPA_{aligned}}$	$3.10^3 - 10^5$	$>\mathbf{10^5}$
$\mathcal{A}_{gT_{aligned}}$	$20 - 10^3$	$>\mathbf{10^5}$
$\mathcal{A}_{\text{CNN}}$	<20	<10

[ESORICS, 2020]

Software hiding with higher noise can be effective against deep learning attacks [Belleville & Masure, 2024]

[TACO, 2019] Automated Software Protection for the Masses against Side-Channel Attacks. 10.1145/3281662
[ESORICS, 2020] Deep Learning Side-Channel Analysis on Large-Scale Traces – a Case Study on a Polymorphic AES. 10.1007/978-3-030-58951-6_22
[Belleville & Masure, 2024] Combining Loop Shuffling and Code PolyMorphism for Enhanced AES Side-Channel Security. 10.1007/978-3-031-57543-3_14

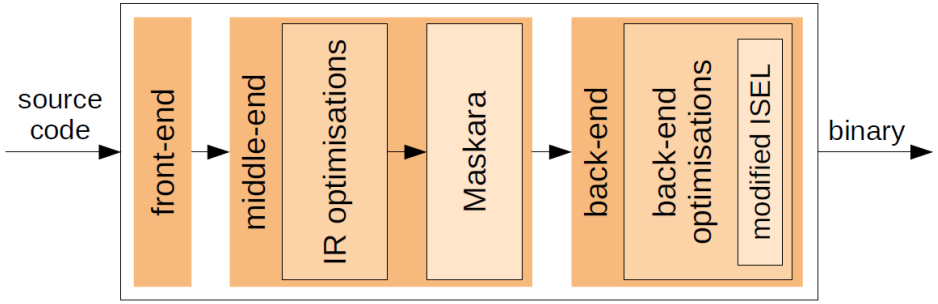
Compilation of First-Order Boolean Masking

PROSECCO, ANR 2015-2019

PhD. Nicolas Belleville, 2019

Masking of secret variables in control-flow structures

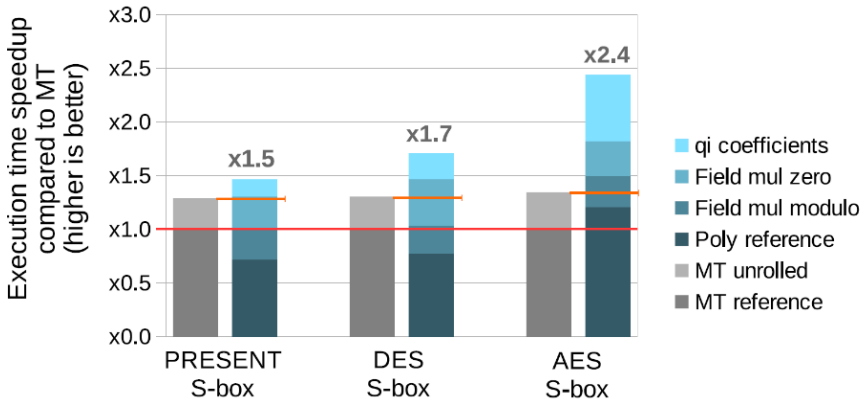
➔ **Confidentiality analysis in CFG**, masking propagation, remasking analysis



Masking of non-linear operations, typically AES/SubBytes:

- usually by *ad hoc* masking schemes
- mostly implemented as Look-Up Tables (LUT)

➔ Masked evaluation of a **polynomial interpolation** of the target LUT



Compilation of First-Order Boolean Masking

PROSECCO, ANR 2015-2019

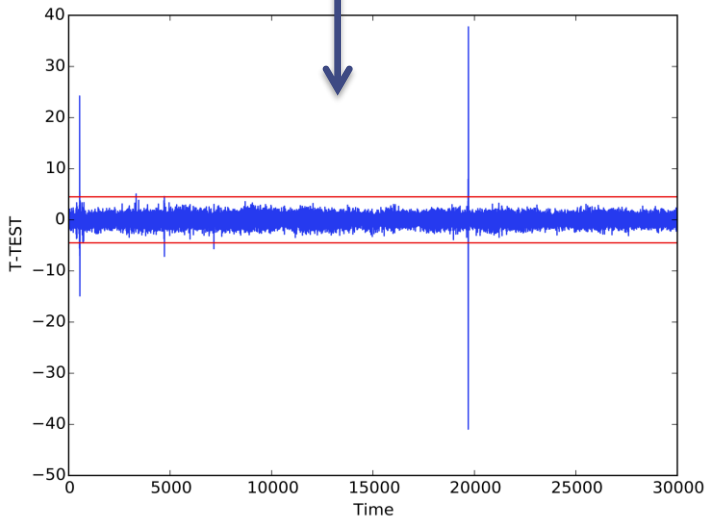
PhD. Nicolas Belleville, 2019

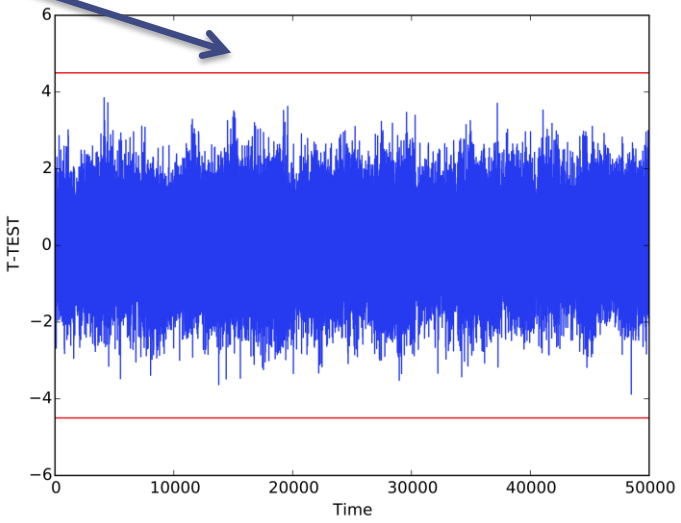
Security evaluation

- **Formally verified** in the value-based leakage model [TCAD, 2020]
  - Collaboration LIP6 [Ben El Ouahma & al., 2017]
- Leakage observed from EM measurements
- Leakage hypothesis: transition-based leakage, **empirically removed by the compiler's back-end**
  - Memory accesses
  - ALU operations

		Distribution of instruction result				
Function	**#inst**	**CST**	**RUD**	**ISD**	**SDD**	**UKD**
AddRoundKey	181	53	128	0	0	0
InterpolatedSboxAccess	707	211	496	0	0	0
SubBytes	11332	3396	7936	0	0	0
ShiftRows	54	2	52	0	0	0
MixColumn	346	30	273	43	0	0
KeySchedule	2971	858	2113	0	0	0

No leakage detected: 0 cases in SSD (statistically dependent from secrets) or UKD (unknown)





[Ben El Ouahma & al., 2019] Side-channel robustness analysis of masked assembly codes using a symbolic approach. 10.1007/s13389-019-00205-7
[TCAD, 2020] Maskara: Compilation of a Masking Countermeasure with Optimised Polynomial Interpolation. 10.1109/TCAD.2020.3012237

Microarchitectural information leakage

IDROMEL, ANR 2021-2024

PhD. Lorenzo Casalino, 2024

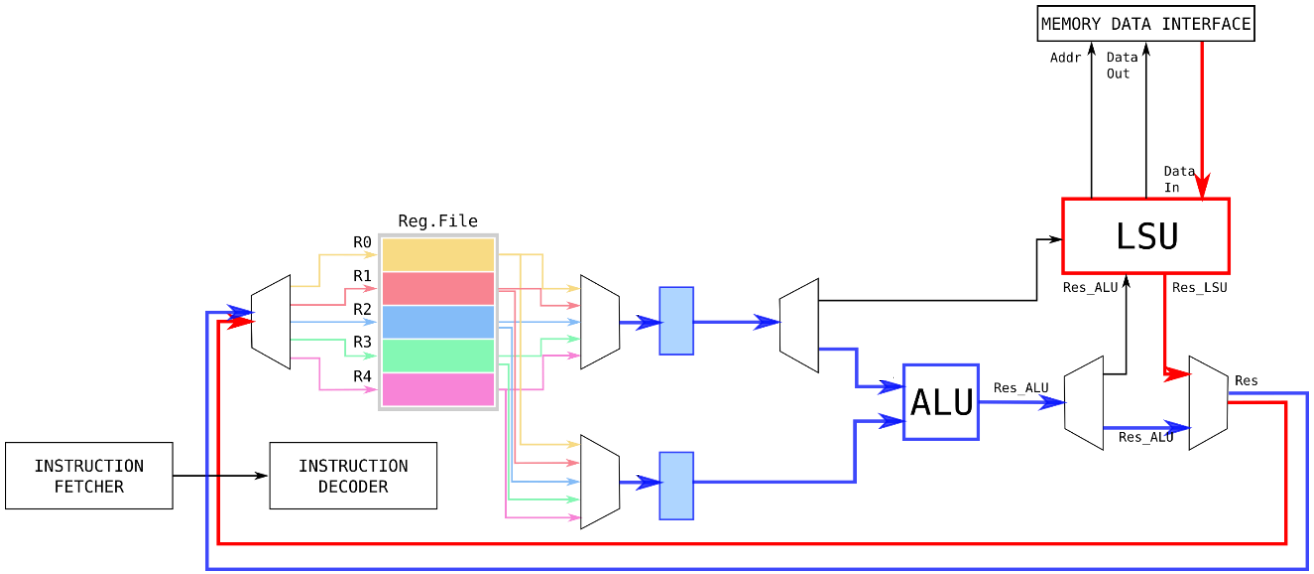
Practical resilience analysis of masked software implementations

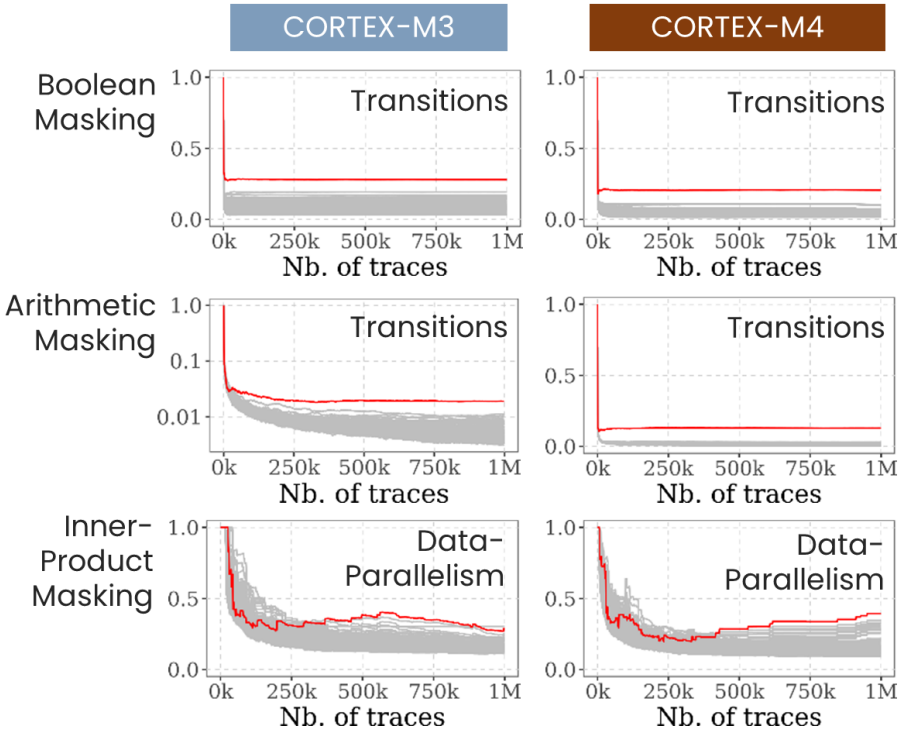
➔ Also vulnerable to micro-architectural leakage [Access, 2023]

Compilation of masking secured against micro-architectural leakage

➔ Instruction scheduling: compiler's backend extended with a micro-architectural model of the data path

➔ Register allocation: policy with leakage interference constraints





[Access, 2023] A Tale of Resilience: On the Practical Security of Masked Software Implementations. 10.1109/ACCESS.2023.3298436

Hardware-Software Co-Design

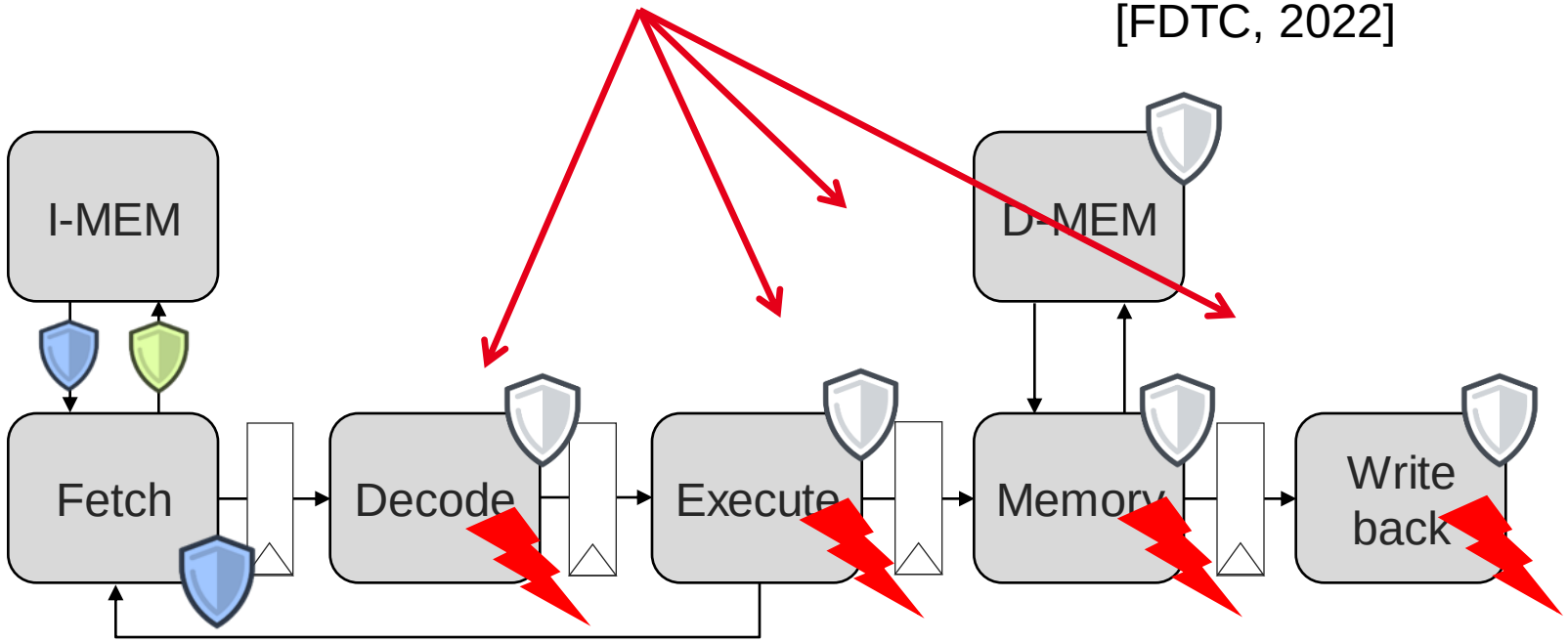
Problem: Faults targeting control signals

COFFI, ANR 2019-2023

PhD. Thomas Chamelot, 2022

- Data integrity
- Code authenticity / integrity
- Control-flow integrity

many possible fault effects! [Laurent & al., 2021] [FDTC, 2022]



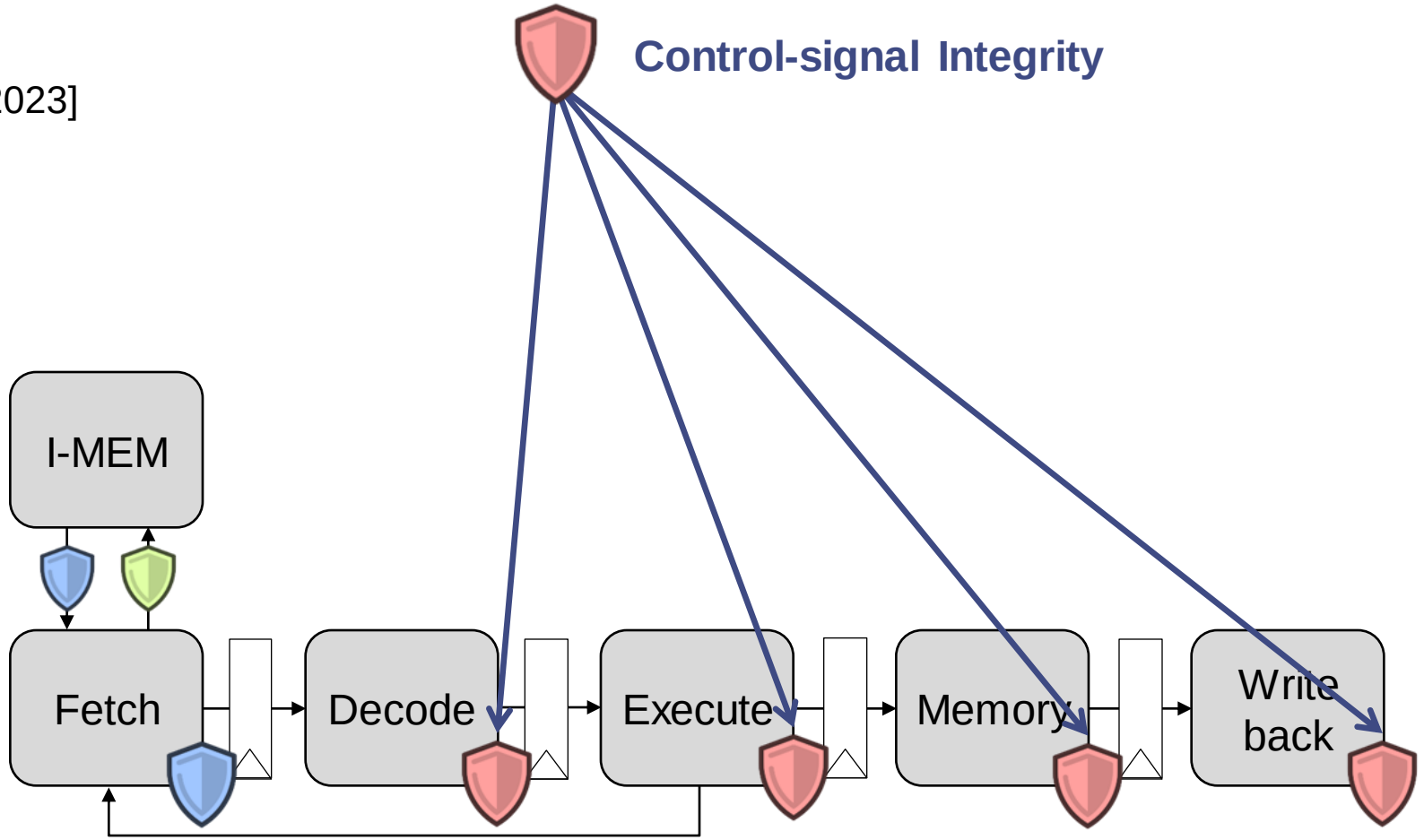
[Laurent & al., 2021] Bridging the Gap between RTL and Software Fault Injection. 10.1145/3446214
[FDTC, 2022] Exploration of fault effects on formal RISC-V microarchitecture models. 10.1109/FDTC57191.2022.00017

Problem: Faults targeting control signals

COFFI, ANR 2019-2023

PhD. Thomas Chamelot, 2022

- Data integrity
- Code authenticity / integrity
- Control-flow integrity
- **Control-signal integrity** [TCAD, 2023]

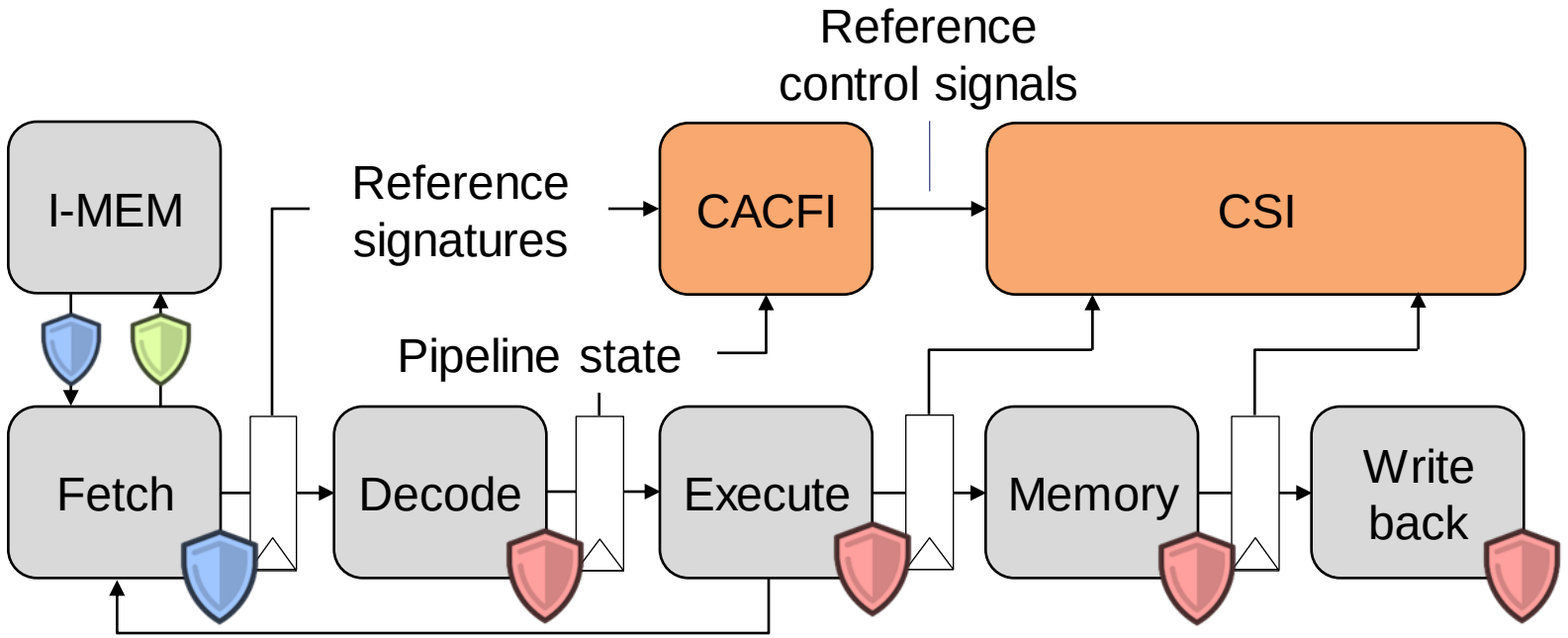


MAFIA: Protection of the microarchitecture against fault injection attacks

COFFI, ANR 2019-2023

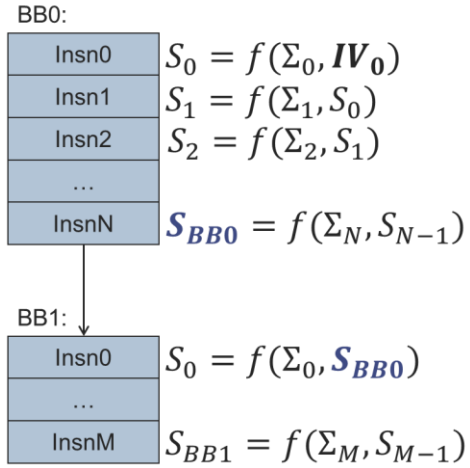
PhD. Thomas Chamelot, 2022

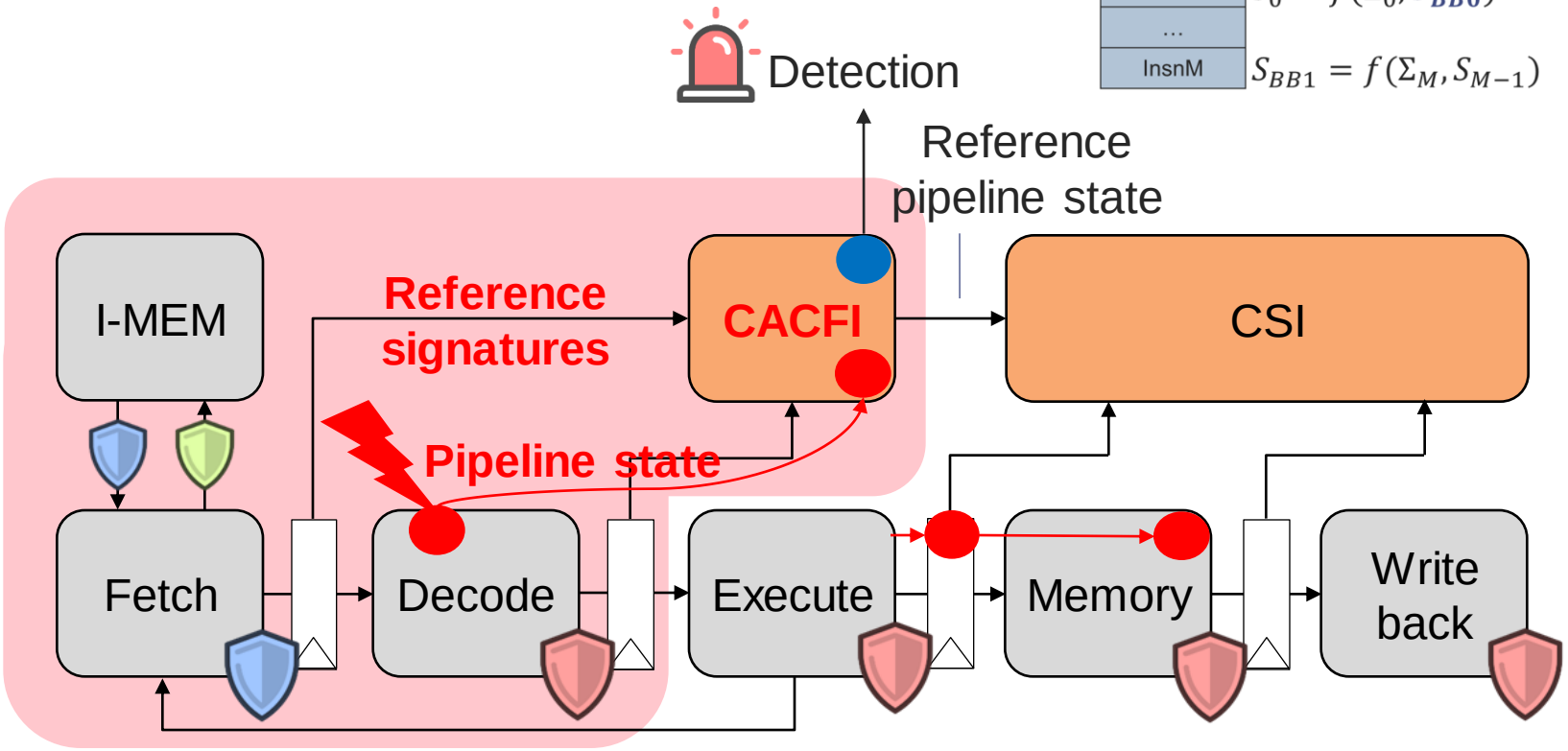
- Data integrity (not supported)
- Code authenticity / integrity
- Control-flow integrity
- **Control-signal integrity**



Code Authenticity and Control-Flow Integrity (CACFI)

- Code authenticity / integrity — signature function f
- Control-flow integrity — signature chaining
- Control-signal integrity — signature computed from **pipeline state** values

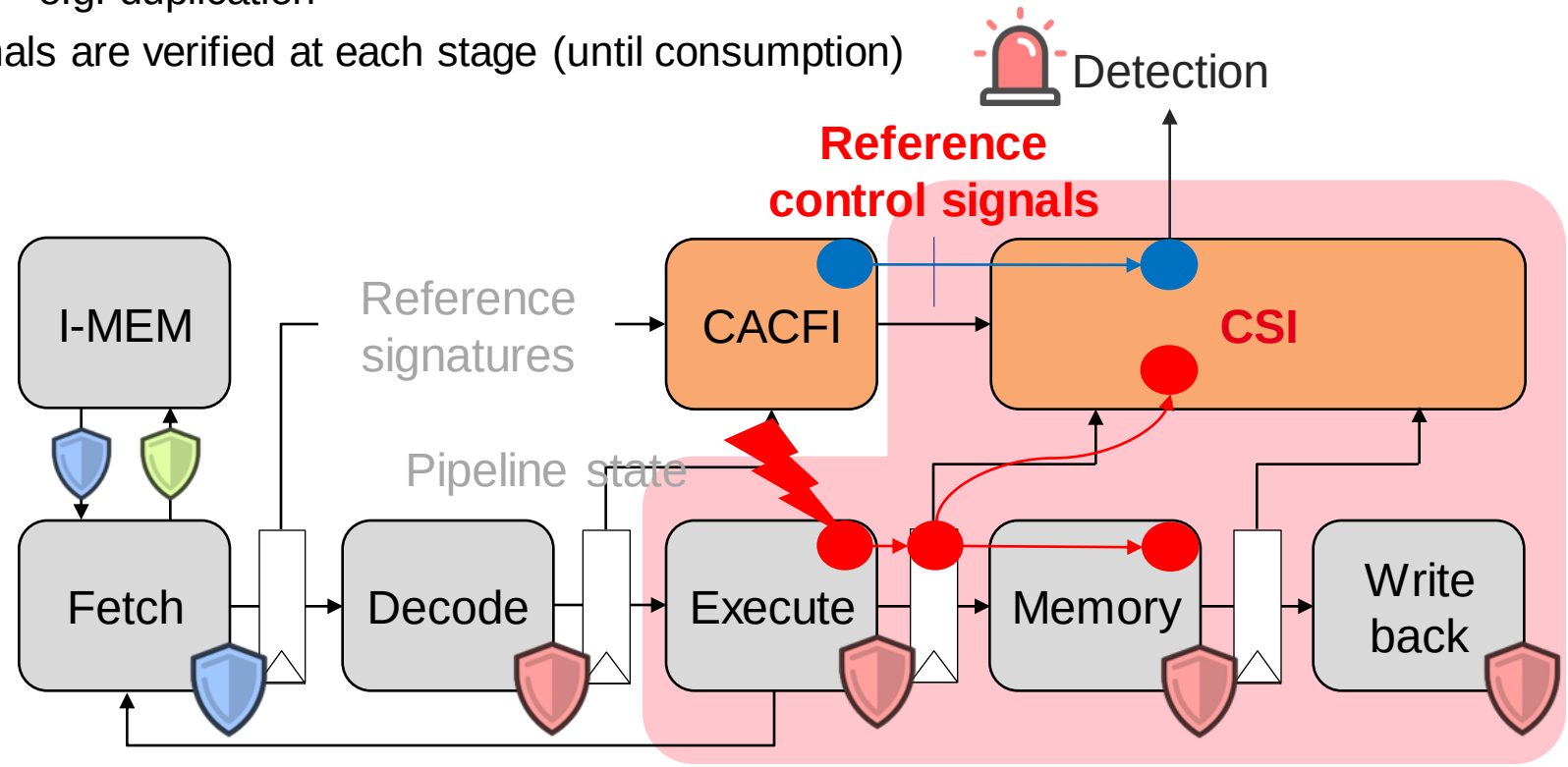




Control-Signal Integrity (CSI)

Control-signal integrity

- Integrity ensured by redundancy
  e.g. duplication
- Signals are verified at each stage (until consumption)

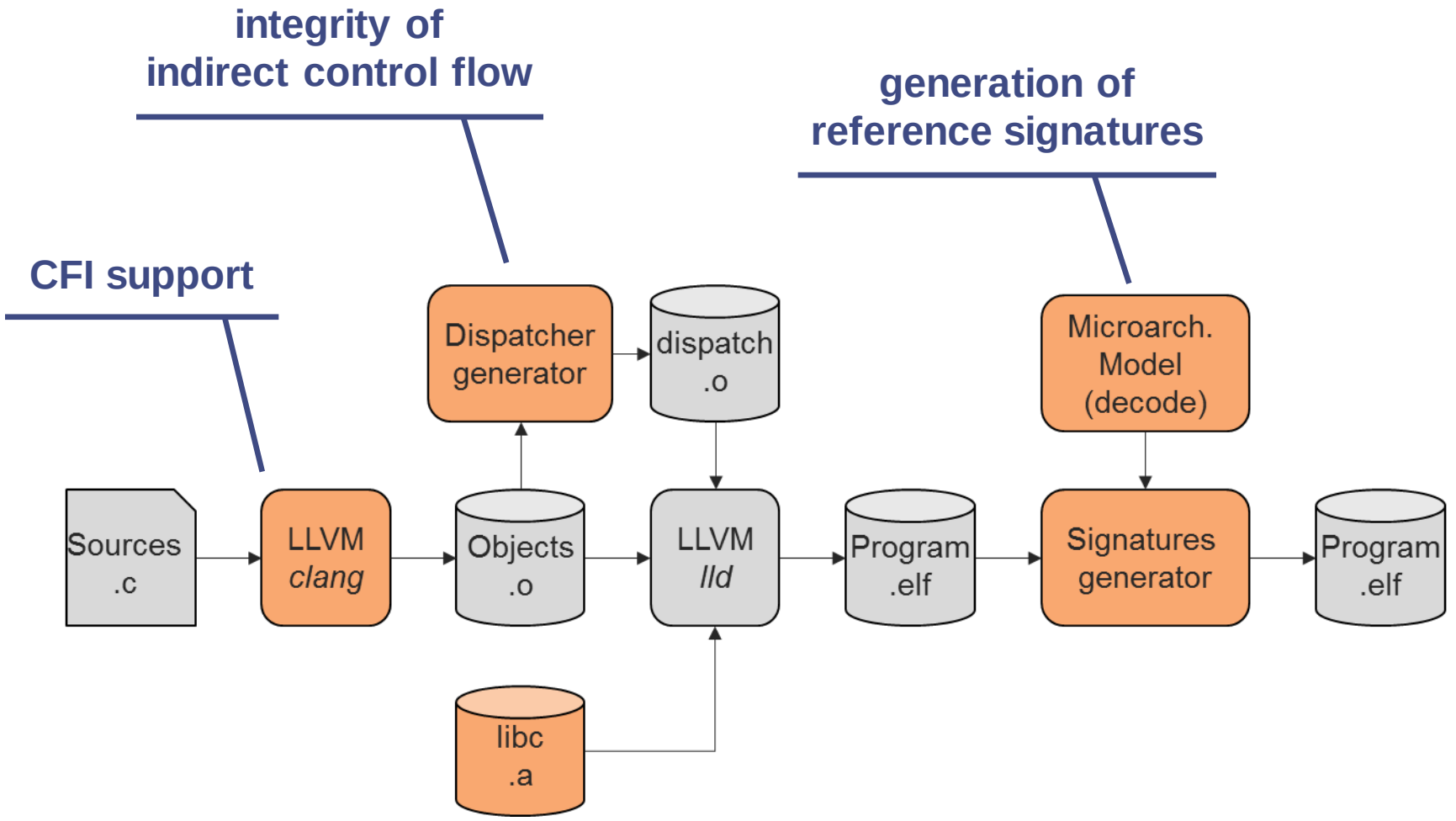


Code authenticity / integrity
Control-flow integrity
Control-signal integrity

MAFIA: Software support

COFFI, ANR 2019-2023

PhD. Thomas Chamelot, 2022



Perspectives

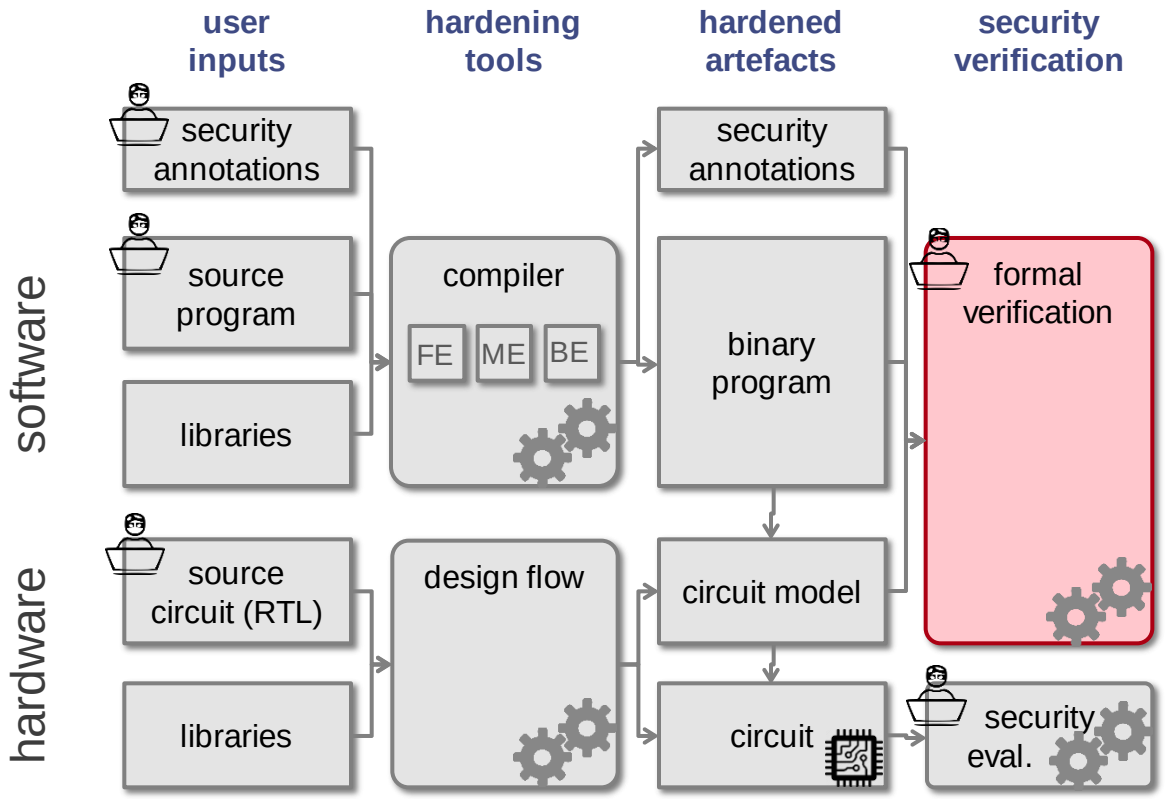
Conclusion

The compiler: the “place to be” for the application of countermeasure?

- automated
- optimized
- configurable and flexible

Can also leverage support for complex countermeasures

HW/SW security formal verification of fault robustness



Status

- Faults incurs extra analysis complexity:
  - State space explosion
  - Multiple faults: combinatorial explosion
- Must consider HW+SW [Laurent & al., 2021]
- Proof of concept of formal HW+SW [FDTC, 2022]

Challenge

- Growing complexity of real case studies
  - Large HW designs, large programs (SW)

First step

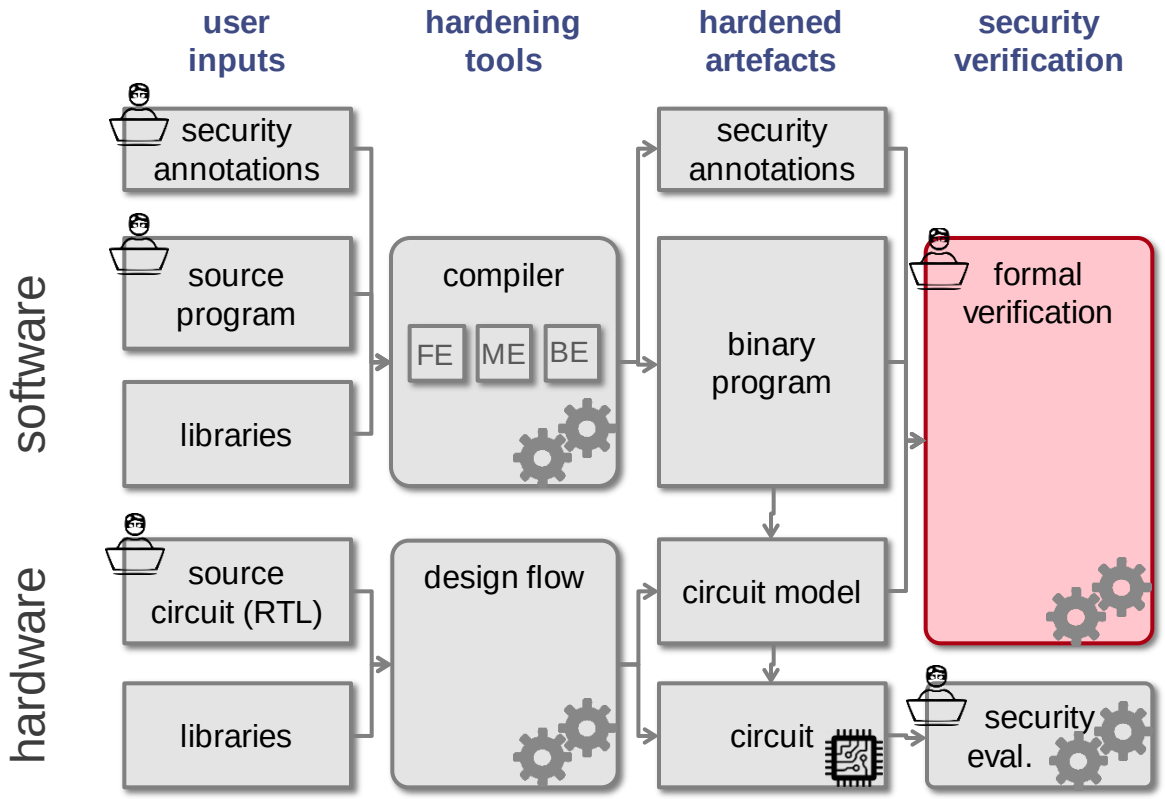
- Opportunity: rely on specific design properties
- k-partitioning for concurrent error detection [TCHES, 2024]
  - 1-fault analysis on OpenTitan + bootloader
  - 3-fault analysis on crypto IPs

PhD Simon Tollec (2021-2024)

PhD 2024-2027. collab. CEA-List / ANSSI

[Laurent & al., 2021] Bridging the Gap between RTL and Software Fault Injection. 10.1145/3446214
[FDTC, 2022] Exploration of fault effects on formal RISC-V microarchitecture models. 10.1109/FDTC57191.2022.00017
[TCHES, 2024] Fault-Resistant Partitioning of Secure CPUs for System Co-Verification against Faults. https://eprint.iacr.org/2024/247

HW/SW security formal verification of fault robustness



Status

- Faults incurs extra analysis complexity:
  - State space explosion
  - Multiple faults: combinatorial explosion
- Must consider HW+SW [Laurent & al., 2021]
- Proof of concept of formal HW+SW [FDTC, 2022]

Challenge

- Growing complexity of real case studies
  - Large HW designs, large programs (SW)

First step

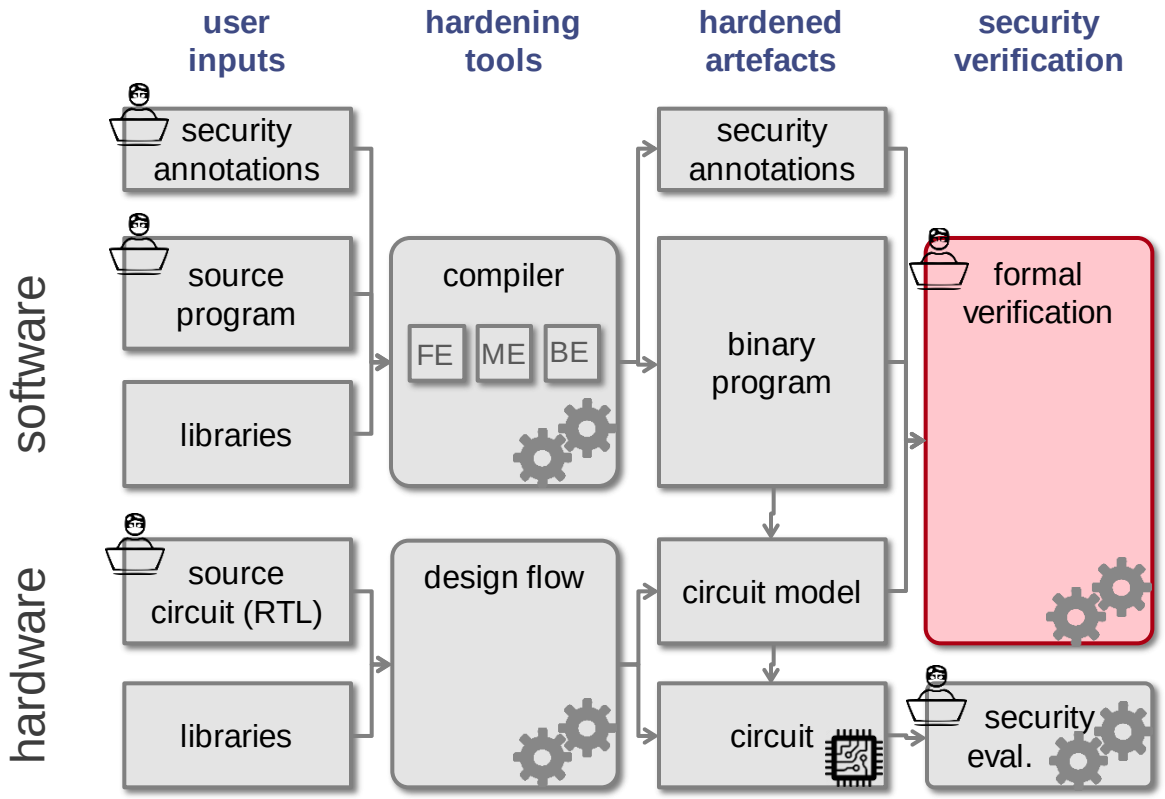
- Opportunity: rely on specific design properties
- k-partitioning for concurrent error detection [TCHES, 2024]
  - 1-fault analysis on OpenTitan + bootloader
  - 3-fault analysis on crypto IPs

PhD Simon Tollec (2021-2024)

PhD 2024-2027. collab. CEA-List / ANSSI

[Laurent & al., 2021] Bridging the Gap between RTL and Software Fault Injection. 10.1145/3446214
[FDTC, 2022] Exploration of fault effects on formal RISC-V microarchitecture models. 10.1109/FDTC57191.2022.00017
[TCHES, 2024] Fault-Resistant Partitioning of Secure CPUs for System Co-Verification against Faults. https://eprint.iacr.org/2024/247

Characterization of vulnerabilities



Status

- Formal methods for proving security properties
- UNSAT → examples
  - Corner case? Unreachable in practice
  - Easily exploitable?

Challenges

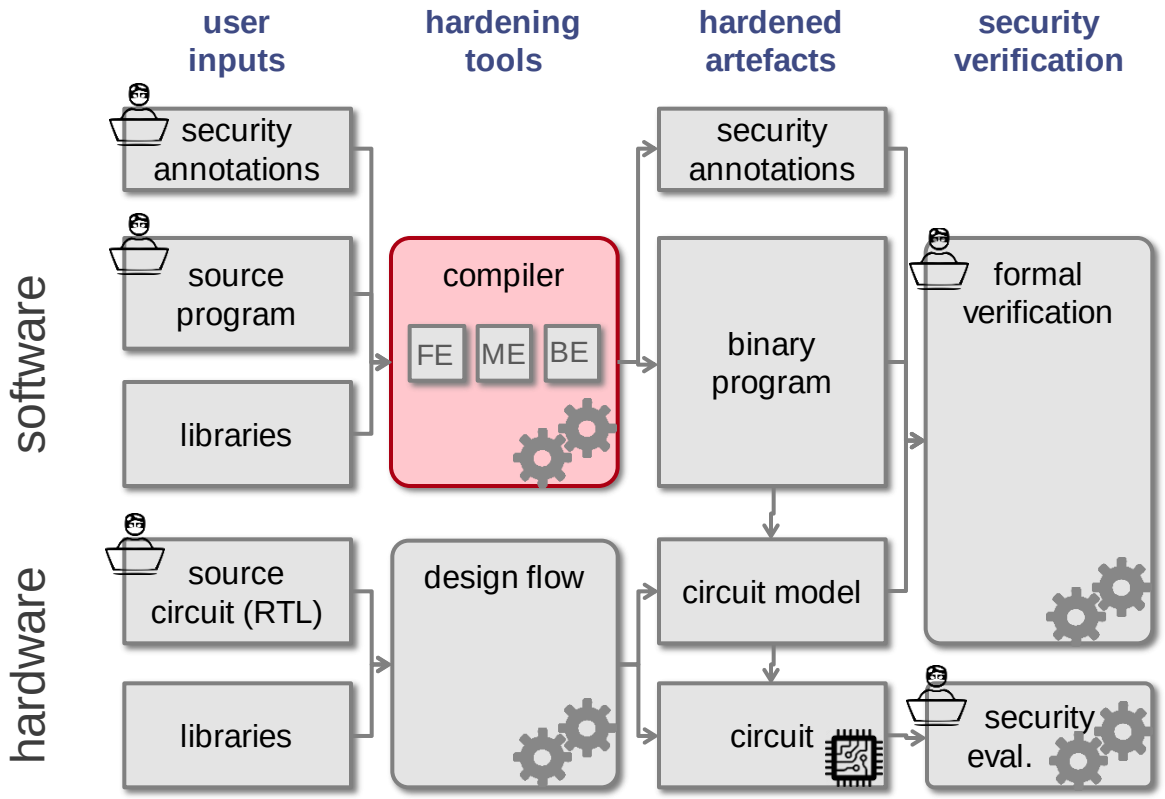
- « severity » of a vulnerability wrt. threat model?
- Characterization of countermeasures that partially support a threat model

First step

- Generation of constraints ensuring the satisfiability of a target property [POPL, 2024]
- Application to fault injection robustness analysis

[POPL, 2024] Inference of Robust Reachability Constraints. 10.1145/3632933

Combination of countermeasures



Status

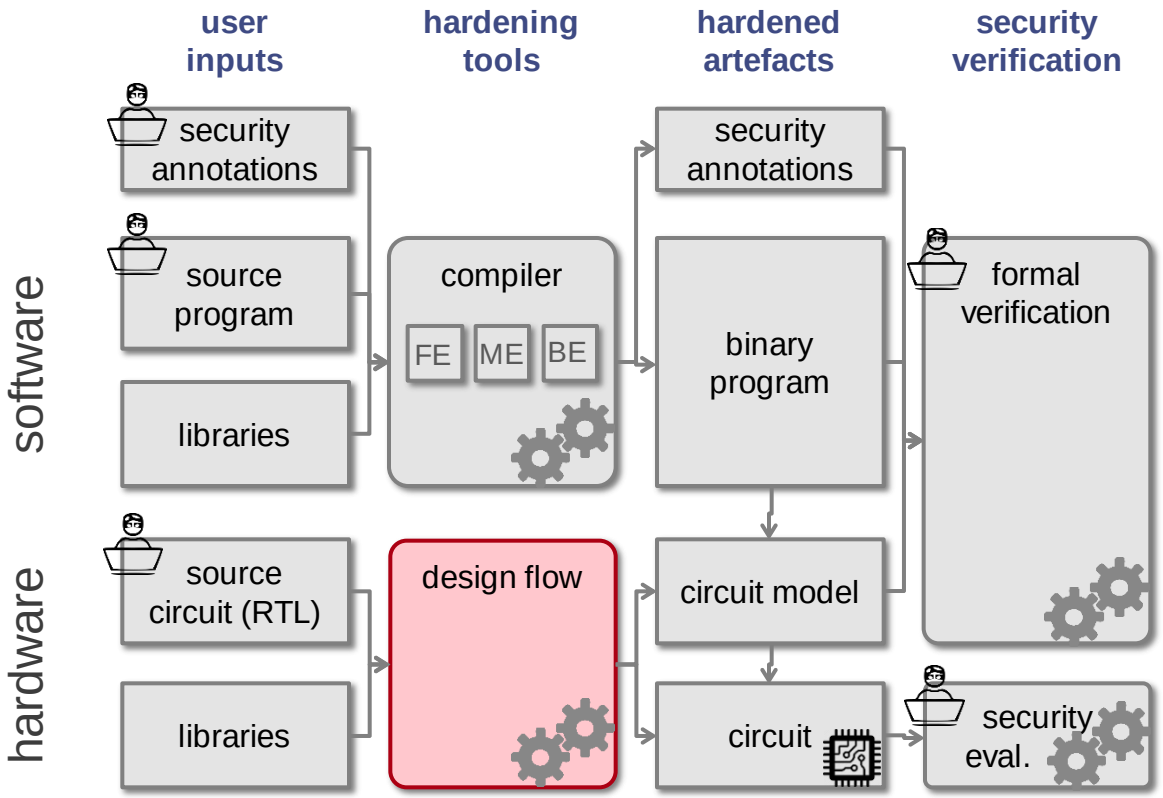
- Many existing countermeasures [Springer, 2018], but **designed in silos**
- Trend: « one countermeasure to rule them all » **SCA + FIA**
  - Data protection only, in crypto

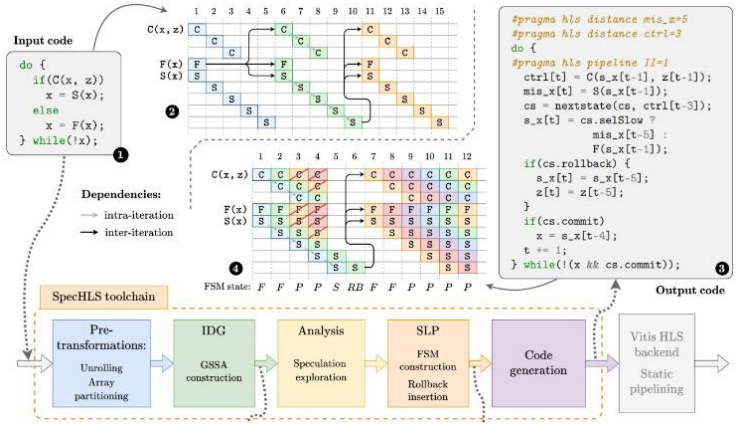
Challenges

- Rely on known countermeasures / security designs
- Compose countermeasures at compilation time
- Articulation with hardware properties

Combination of hiding and masking against SCA (FlexSecurity, Carnot List)

Application of hardware countermeasures





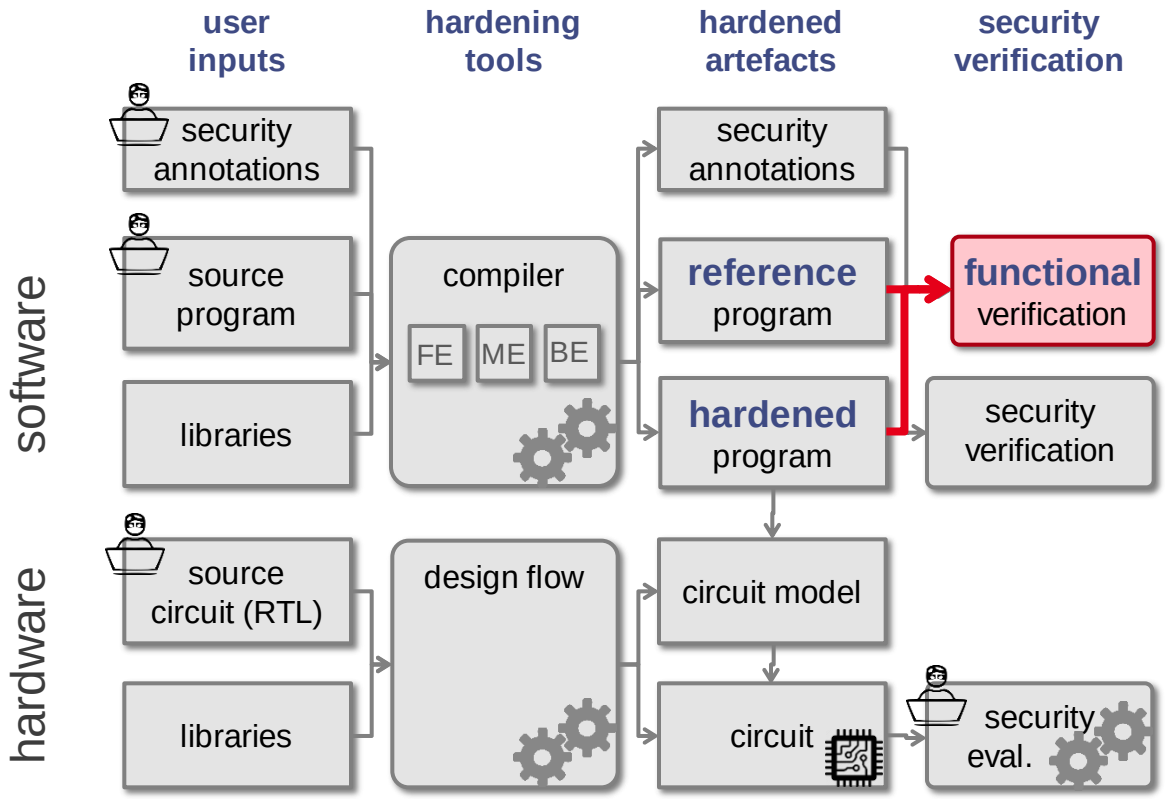
[Gorius & al., 2022]

Challenges

- Demonstrate the automated synthesis of typical countermeasures (redundancy, etc.)
- Microarchitectural-level countermeasures (MAFIA) require to « open the box » of the synthesis flow

ANR LOTR (lead IRISA)

Proof of functional correctness



Status

- ✓ Automated security application
- ✓ Formal security verification
- ? Functional correctness

Challenges

- Extend relational analysis [Daniel & al., 2020] to program analysis
- Prove that the **reference** and the **hardened programs** are **functionally equivalent**

[PhD, 2024-2027] collab. CEA-List DILS/DSCIN

Supervision & funding support

Supervision and co-supervision

- 6 PhD students
- 7 research engineers, post-docs
- 1 development engineer

Collaborative projects (most representative)

- COGITO (ANR 2013-2017) – coordinator
- PROSECCO (ANR 2016-2019)
- CLAPs (IRT Pulse, 2017-2020) – coordinator
- Serene-IoT (PENTA, 2016-2019) – WP lead
- COFFI (ANR 2019-2023)
- IDROMEL (ANR 2021-2025)
- LOTR (ANR 2023-2027)

Scientific collaborations (most representative)

CEA

- CEA Leti/List – DACLE / DSCIN
- CEA Leti – DSYS
- CEA Leti – Cesti
- CEA List – DILS

France

- LSAS Gardanne – joint team CEA Leti + EMSE
- XLIM
- INRIA Rennes
- LIP6 – Sorbonne Univ.
- VERIMAG – Univ. Grenoble Alpes
- ARM France
- Thalès
- ANSSI

International

- TU Graz
- Univ. Milano − POLIMI

Application outillée de contre-mesures contre les attaques matérielles

Soutenance d'Habilitation à Diriger des Recherches

Damien Couroussé

28 août 2024

Composition du jury

Rapporteurs	**Guy Gogniat**	professeur à l'Université Bretagne Sud
	Guillaume Hiet	professeur à CentraleSupélec Rennes
	Ingrid Verbauwhede	professeure à KU Leuven
Examinateurs	**Aurélien Francillon**	professeur à EURECOM
	David Hély	professeur à LCIS, Université Grenoble Alpes
	Karine Heydemann	experte sécurité à Thalès DIS, chercheuse associée au LIP6
	Marie-Laure Potet	professeure à Université Grenoble Alpes