

# Réunion flash

Point hebdomadaire

Duzés Florian




# Sommaire

1. État des lieux
2. Retour sur les fuites
3. Projections futures
4. Antoine geimer
5. Conclusion

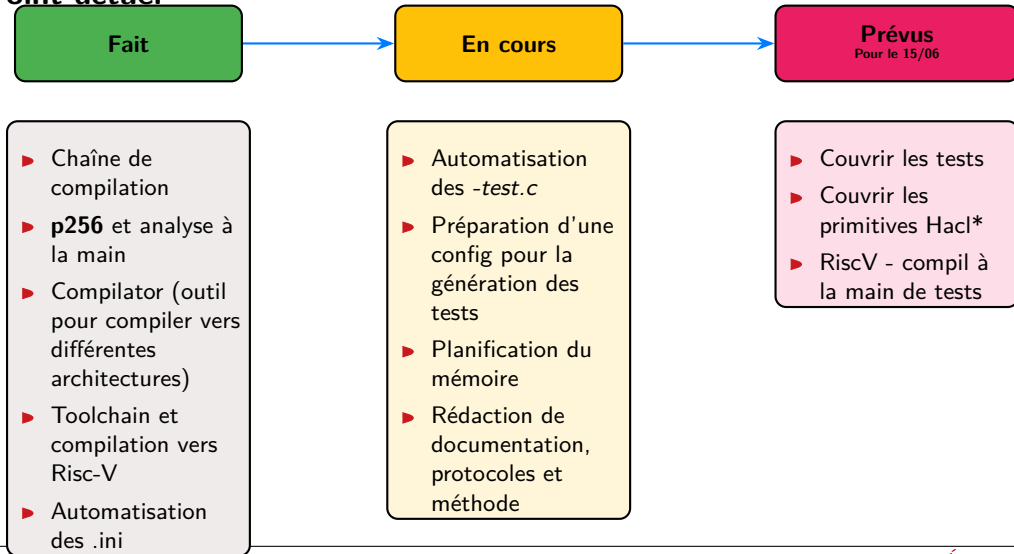
# 01

## État des lieux





## Point actuel





## Nouveau organigramme

### Risc-V

- ☐ toolchain 32 bits
- ☐ Clang+LLVM toolchain 64/32
- ☐ Binsec sur les tests Hacl\*
  - chacha-encrypt
  - \*\*\*-decrypt
  - poly32
  - blake2b-256
- ☐ Intégrer Érysichton

### Mémoire

- ☐ Plan de l'intro
- ☐ Plan général
- ☐ template CSI
- ☐ Latex prêt

### Automatisation

- ☐ générer *-test.c*
- ☐ *-test.json*
- ☐ Compilation croisé
- ☐ Option de compilation

### Poursuite en thèse ?

- ☐ CV à jour
- ☐ Identifier sujets
- ☐ Domaine de prédilection
- ☐ mél de contact



## Travail fait

### Risc-V

- ☒ ~~toolchain 32-bits~~
- ☒ ~~Clang+LLVM toolchain 64/32~~
- ☐ Binsec sur les tests HACL\*
  - chacha-encrypt
  - \*\*\*-decrypt
  - poly32
  - blake2b-256
- ☒ ~~Intégrer Érysichton~~

### Mémoire

- ☒ ~~Plan de l'intro~~
- ☐ Plan général
- ☒ ~~template CSI~~
- ☒ ~~Latex prêt~~

### Automatisation


- ☐ générer `-test.c`
- ☐ `-test.json`
- ☐ Compilation croisé
- ☐ Option de compilation
- ☐ COMPILER HACL\* en répétition

### Poursuite en thèse ?

- ☒ ~~CV à jour~~
- ☒ ~~Identifier sujets~~
- ☒ ~~Domaine de prédilection~~
- ☒ ~~mél de contact~~

# 02

## Retour sur les fuites





# Introduction

## Première utilisation de la toolchain RiscV - Première analyse en direct

```
$ binsec -sse -sse-depth 1000000 -sse-script study.ini -checkct p256-bug.exe -fml-solver-timeout 10
[checkct:result] Instruction 0x000104e0 has memory access leak (3.218s)
[checkct:result] Instruction 0x000104ec has memory access leak (6.443s)
[checkct:result] Instruction 0x00010504 has memory access leak (9.664s)
[checkct:result] Instruction 0x00010512 has memory access leak (12.909s)
[checkct:result] Instruction 0x0001052a has memory access leak (16.126s)
[checkct:result] Instruction 0x00010538 has memory access leak (19.357s)
[checkct:result] Instruction 0x00010550 has memory access leak (22.504s)
[checkct:result] Instruction 0x0001055e has memory access leak (25.196s)
[checkct:result] Instruction 0x00010578 has memory access leak (26.887s)
...
[checkct:result] Program status is : insecure (75.302)
```

Code – Exécution de la dernière fois





# Compilation

Revoyons la commande de compilation (1)

```
mkdir -p $HOME/apprentissage/compilation/builds/riscv64-unknown-linux-gnu-gcc
$HOME/cross_compil/riscv64-gnu-toolchain-x86_64-riscv64-unknown-linux-gnu/bin/riscv64-unknown-linux-gnu-gcc \
-DHACL_CAN_COMPILE_VEC128 -I$HOME/recoules-hacl-star/hacl-star/dist/karamel/include \
-I$HOME/recoules-hacl-star/hacl-star/dist/gcc-compatible \
-I$HOME/recoules-hacl-star/hacl-star/dist/gcc-compatible/internal \
-I$HOME/recoules-hacl-star/hacl-star/dist/karamel/krmlib/dist/minimal \
-I$HOME/recoules-hacl-star/hacl-star/secure_api/merkle_tree \
-Werror -Wall -Wno-deprecated-declarations -Wno-unused \
-c source/p256-bug.c -o \
$HOME/apprentissage/compilation/builds/riscv64-unknown-linux-gnu-gcc/p256-bug.o
$HOME/cross_compil/riscv64-gnu-toolchain-x86_64-riscv64-unknown-linux-gnu/bin/riscv64-unknown-linux-gnu-gcc \
-DHACL_CAN_COMPILE_VEC128 -I$HOME/recoules-hacl-star/hacl-star/dist/karamel/include \
-I$HOME/recoules-hacl-star/hacl-star/dist/gcc-compatible \
-I$HOME/recoules-hacl-star/hacl-star/dist/gcc-compatible/internal \
-I$HOME/recoules-hacl-star/hacl-star/dist/karamel/krmlib/dist/minimal \
-I$HOME/recoules-hacl-star/hacl-star/secure_api/merkle_tree \
-Werror -Wall -Wno-deprecated-declarations -Wno-unused \
$HOME/apprentissage/compilation/builds/riscv64-unknown-linux-gnu-gcc/p256-bug.o \
-o $HOME/apprentissage/compilation/builds/riscv64-unknown-linux-gnu-gcc/p256-bug.exe
```

Code – Commande de compilation



## Compilation simplifier

Revoyons la commande de compilation (2)

```
riscv64-unknown-linux-gnu-gcc \  
-I$HACL_STAR/dist \  
-Werror -Wall -Wno-deprecated-declarations -Wno-unused \  
-c p256-bug.c -o p256-bug.o  
  
riscv64-unknown-linux-gnu-gcc \  
-I$HACL_STAR/dist \  
-Werror -Wall -Wno-deprecated-declarations -Wno-unused \  
p256-bug.o -o p256-bug.exe
```

Code – Commande de compilation



## Ajout de force au compilateur

```
riscv64-unknown-linux-gnu-gcc \  
-I$HACL_STAR/dist \  
-Werror -Wall -Wno-deprecated-declarations -Wno-unused -Ox \  
-c p256-bug.c -o p256-bug.o  
  
riscv64-unknown-linux-gnu-gcc \  
-I$HACL_STAR/dist \  
-Werror -Wall -Wno-deprecated-declarations -Wno-unused -Ox \  
p256-bug.o -o p256-bug.exe
```

Code – Commande de compilation avec force

```
[checkct:result] Program status is : secure (0.041)  
[checkct:info] 1 visited path covering 32 instructions  
[checkct:info] 0 / 0 control flow checks pass  
[checkct:info] 14 / 14 memory access checks pass
```

Code – Résultat significatif : options {2,3,s,z}



## Identification de la fuite

```
$ for d in 104e0 104ec 10504 10512 1052a 10538 10550 1055e 10578; do
  cat disas | grep "$d"
done
104e0: 6398          ld  a4,0(a5)
104ec: 6394          ld  a3,0(a5)
10504: 6398          ld  a4,0(a5)
10512: 6394          ld  a3,0(a5)
1052a: 6398          ld  a4,0(a5)
10538: 6394          ld  a3,0(a5)
10550: 6398          ld  a4,0(a5)
1055e: 6394          ld  a3,0(a5)
10578: e398          sd  a4,0(a5)
```

Code – Instruction correspondantes aux fuites

Instructions de chargement et de mémorisation.



# Identification de la fuite

```
000000000000104b0 1      F .text 00000000000000f8      cmovznz4
104b0 - 105a6
```

## Code – Fonction ciblé

```
000000000000105a8 <main>:
105a8: 1141          addi  sp,sp,-16
105aa: e406          sd   ra,8(sp)
105ac: e022          sd   s0,0(sp)
105ae: 0800          addi  s0,sp,16
105b0: 67c9          lui   a5,0x12
105b2: 0107b703      ld    a4,16(a5) # 12010 <cin>
105b6: 85818693      addi  a3,gp,-1960 # 12060 <r>
105ba: 83818613      addi  a2,gp,-1992 # 12040 <y>
105be: 67c9          lui   a5,0x12
105c0: 02078593      addi  a1,a5,32 # 12020 <x>
105c4: 853a          mv    a0,a4
105c6: eebff0ef      jal   104b0 <cmovznz4>
```

## Code – Appels précédents

```
#define SIZE 4 uint64_t cin; uint64_t x[SIZE]; uint64_t y[SIZE]; uint64_t r[SIZE]; static void cmovznz4(uint64_t cin, uint64_t *x, uint64_t *y, uint64_t *r)
```



## Identification de la fuite (2)

```
104a6: 853e          mv  a0,a5
104a8: 70e2          ld  ra,56(sp)
104aa: 7442          ld  s0,48(sp)
104ac: 6121          addi sp,sp,64
104ae: 8082          ret
```

Code – Extrait de la fonction de masquage


```
104ce: f91ff0ef      jal 1045e <FStar_UInt64_eq_mask>
104d2: 87aa          mv  a5,a0
104d4: fff7c793      not a5,a5
104d8: fef43423      sd  a5,-24(s0)
104dc: fa843783      ld  a5,-88(s0)
104e0: 6398          ld  a4,0(a5)
```

Code – Extrait avant la première fuite

Hypothèse : *Résultats de **FStar\_UInt64\_eq\_mask** qui provoque la fuite*

# 03

## Projections futures





## Point sur le travail

### Sentiment de dispersion

- ▶ Plein de choses à faire
- ▶ Plein de connaissance à avaler
- ▶ Priorisation de tâches (point faible)
- ▶ Temps contraint





## Point sur le travail

### Sentiment de dispersion

- ▶ Plein de choses à faire
- ▶ Plein de connaissance à avaler
- ▶ Priorisation de tâches (point faible)
- ▶ Temps contraint
  - Immobilisation - ralentissement



## Point sur le travail

### Sentiment de dispersion

- ▶ Plein de choses à faire
- ▶ Plein de connaissance à avaler
- ▶ Priorisation de tâches (point faible)
- ▶ Temps contraint
  - Immobilisation - ralentissement

### Retour au bases

- ▶ Méthodologie de travail
- ▶ Rétro planning pour le rendu
- ▶ Fixer une fin au développement



# Aperçu pour la thèse

*Petit hors-sujet de partage et de retour de ma démarche*

# Aperçu pour la thèse

<p>07 50 64 15 22</p> <p>florian.duzes@inria.fr</p> <p>www.inria.fr/~florian-duzes/</p> <p>github.com/FlorianDuzes</p>	<p>Duzés Florian</p> <p>UC</p> <p>université BORDEAUX</p> <p>Inria</p>
<p><b>Appétences</b> — Je suis intéressé par de nombreux sujets que j'ai découverts au long de mon cursus : Implémentation de la recherche, Implémentation sûr et sécurisé, Chiffrement homomorphe, Attaque cryptographique, Sécurité des logiciels embarqués, Sécurité des canaux auxiliaires</p>	
<p><b>Compétences</b></p>	
<p><b>Algorithmique</b> : Quantum Computing, algèbre, complexité</p> <p><b>Cryptologie</b> : Mélléus, LWE, Kyber, Dilithium, AES, RSA, DSS</p> <p><b>Systèmes</b> : Windows, Linux</p> <p><b>Architectures</b> : INT2, ARM, RISC-V, shellcodes</p>	<p><b>Langage</b> : Python, SageMath, C, Rust, LaTeX, OCaml, F#</p> <p><b>Langues</b> : Anglais et Espagnol - C1</p> <p><b>Personnelles</b> : Travail d'équipe, organisé, curieux, prise d'initiative sous supervision, autonome</p>
<p><b>Expériences professionnelles</b></p>	
<p><b>Chercheur en Sécurité</b></p> <p>INRIA Paris - Prosecco / CEA</p> <p>– Recherche sur la sécurité des attaques par canaux auxiliaires, Travaux avec BINSec sur Haci*. Automatisation de l'analyse avec différentes versions de compilateurs et vers différentes architectures.</p>	<p>2025</p> <p>5 mois</p>
<p><b>Prévidences étudiantes</b></p> <p>Associations Alao - Université Champollion</p> <p>CPVU Université Champollion</p> <p>Conseil Etudiant Université Champollion</p>	<p>2022 - 2023</p> <p>12 mois</p> <p>12 mois</p> <p>12 mois</p>
<p><b>Animation jeunesse - Directeur de centre</b></p> <p>Eclaireurs et éclaireuses de France (EJEF, associations laïque) - bénévoles</p> <p>Wolanga - direction de centre de vacances en 2024</p>	<p>2017 - 2024</p> <p>7 ans</p> <p>3 ans</p>
<p><b>Expériences laborales</b></p>	<p>2015 - 2024</p>
<p><b>Formations</b></p>	
<p><b>Master Cryptologie et sécurité informatique</b></p> <p>Option : Cryptographie Post-Quantique, Algorithmique arithmétique, Cryptanalyse, Cartes à puce, Sécurité des systèmes</p> <p>Université de Bordeaux</p>	<p>2025</p>
<p><b>Licence Informatique, mention Intelligence Artificielle</b></p> <p>Institut National Universitaire Jean-François Champollion</p>	<p>2022</p>
<p><b>Diplômes et certifications</b></p> <p>– Certificat de langue anglaise, Cambridge Center - IRY</p> <p>– Divers PACT, BAKA, RSSE</p> <p>– Bachelierato</p>	
<p><b>Projets informatiques</b></p>	
<p><b>Travaux de recherche, projets et stage</b></p> <p>– Attaque par canal auxiliaire sur ECCDSA et réduction de réseau, implémentation SageMath - <i>Recherche</i></p> <p>– Introduction complète aux Générateurs de nombres pseudo-aléatoires - <i>stage</i></p> <p>– Etude du cryptosystème de Puaillier en application sur une application de messagerie, implémentation en C - <i>Recherche</i></p> <p>– Logiciel d'étude des comportements humains pour dériver une IA, programmation Python - <i>Recherche</i></p> <p>– Développement d'un compilateur C en OCaml - <i>Application (personnelle)</i></p>	<p>2019 - 2025</p>
<p><b>Global Game Jam - 2020/2021/2022/2023/2024</b></p> <p>– Créativité, travail à distance et organisation avec les fuseaux horaires d'une équipe de huit personnes</p> <p>– Développement de jeu vidéo durant cet événement mondial, sur 48h.</p>	<p>2020 - 2024</p>
<p><b>Nuit de l'Info - Passage Python</b></p> <p>– Programmation de site web sous forme de concours national avec des défis proposés par des entreprises</p> <p>– Travail d'équipe, brainstorming, développement web et gestion de projets en ~14h.</p> <p>– Récompensé pour les solutions de l'équipe</p>	<p>2019 - 2021</p>
<p><b>Loisirs</b></p>	
<p>Gastronomie    Cinéma    Sports: Judo, Taekwondo, VTT    Lectures: Hypérion, Belgarade, Les chevaliers d'Émeraude</p>	

## Mél

Je suis Florian Duzés. Je suis en train de terminer mon master en Cryptologie et Sécurité Informatique, à Bordeaux, avec un stage aux côtés de M. Aymeric Fromherz à l'INRIA Paris sur la sécurité de Hacl\* face aux attaques par canaux auxiliaires par mesures de temps.


Je voudrais poursuivre avec une thèse en lien avec mon sujet, mais je suis plus généralement intéressé par la programmation sécurisée, la sécurité embarqué et la sécurité des systèmes face aux canaux auxiliaires.

Mon cœur balance aussi vers la programmation cryptographique, l'implémentation de recherche en cryptologie, le chiffrement homomorphe et la cryptologie post-quantique.

Je viens vous demander si vous connaissez une personne qui puisse être intéressée ou si vous connaissez quelqu'un qui puisse m'aiguiller pour trouver une thèse en lien avec ces sujets.

# 04

## Antoine geimer





# Courte présentation

## Antoine Geimer

- ▶ Thèsard à l'université de Lille
- ▶ Sujet : **Détection automatique de vulnérabilités dues aux canaux auxiliaires microarchitecturaux.**
- ▶ Intervention à l'INRIA le 16/06
- ▶ Dépôt Github de son article

## A Systematic Evaluation of Automated Tools for Side-Channel Vulnerabilities Detection in Cryptographic Libraries

Antoine Geimer  
Univ. Lille, CNRS, Inria  
Univ. Rennes, CNRS, IRISA  
Lille, France

Mathéo Vergnolle  
Université Paris-Saclay, CEA, List  
Gif-sur-Yvettes, France

Frédéric Recoules  
Université Paris-Saclay, CEA, List  
Gif-sur-Yvettes, France

Lesly-Ann Daniel  
KU Leuven, imec-DistriNet  
Leuven, Belgium

Sébastien Bardin  
Université Paris-Saclay, CEA, List  
Gif-sur-Yvettes, France

Clémentine Maurice  
Univ. Lille, CNRS, Inria  
Lille, France

### Abstract


To protect cryptographic implementations from side-channel vulnerabilities, developers must adopt constant-time programming practices. As these can be error-prone, many side-channel detection tools have been proposed. Despite this, such vulnerabilities are still manually found in cryptographic libraries. While a recent paper by Jancar et al. shows that developers rarely perform side-channel detection, it is unclear if existing detection tools could have found these vulnerabilities in the first place.

To answer this question we surveyed the literature to build a classification of 34 side-channel detection frameworks. The classification we offer compares multiple criteria, including the methods used, the scalability of the analysis or the threat model considered. We then built a unified common benchmark of representative cryptographic operations on a selection of 5 promising detection tools. This benchmark allows us to better compare the capabilities of each tool, and the scalability of their analysis. Additionally, we offer a classification of recently published side-channel vulnerabilities. We then test each of the selected tools on benchmarks reproducing a subset of these vulnerabilities as well as the context in which they appear. We find that existing tools can struggle to find vulnerabilities for a variety of reasons, mainly the lack of support for SIMD instructions, implicit flows, and internal secret generation.

### 1 Introduction

Implementing cryptographic algorithms is an arduous task. Beyond functional correctness, the developers must also ensure that their code does not leak potentially secret information through side channels. Since Paul Kocher's seminal work [82], the research community has combed through software and hardware to find vectors allowing for side-channel attacks, from execution time to electromagnetic emissions. The unifying principle behind this class of attacks is that they do not exploit the algorithm specification but rather physical characteristics of its execution. Among the aforementioned attack vectors, the processor microarchitecture is of particular interest, as it is a shared resource between multiple programs. By observing the target execution through microarchitectural components (e.g., cache [88, 139], branch predictor [6, 57], DRAM [98], CPU ports [9]), an attacker can deduce secret information beyond what is normally possible with classical cryptanalysis. Side-channel primitives using these components allow an attacker to reconstruct secret-dependent control flow and table look-ups. This problem is exacerbated in multi-core processors and VM environments, where execution can be shared concurrently by multiple actors, and in trusted execution environments, where the privileged control of untrusted operating systems can be leveraged to perform controlled-channel attacks [137].

# 05 Conclusion





# Conclusion

## Un stage super intéressant

- ▶ Riche
- ▶ Ouverture au monde





# Conclusion

## Un stage super intéressant

- ▶ Riche
- ▶ Ouverture au monde

## Modèle de travail

- ▶ À reconcevoir
- ▶ À fixer les prochaines missions

# Conclusion

## Un stage super intéressant

- ▶ Riche
- ▶ Ouverture au monde

## Modèle de travail

- ▶ À reconcevoir
- ▶ À fixer les prochaines missions

## Travaux à déterminer

- ▶ *Je note tous sur mon carnet*

*Merci.*

