

# Réunion flash

Point hebdomadaire

Duzés Florian




# Sommaire

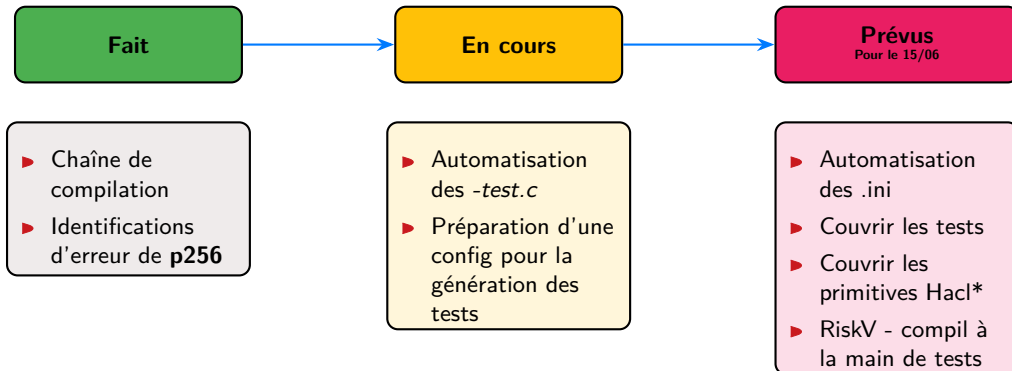
1. État des lieux
2. Automatique, pas facile
3. Risc-V me voilà
4. Conclusion

# 01

## État des lieux

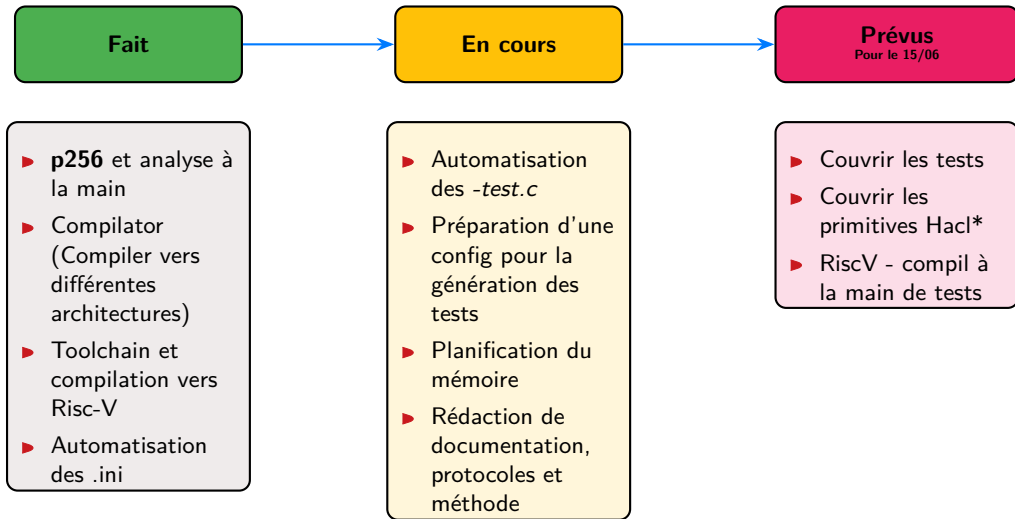


## Point actuel






## Réalisation



# 02

## Automatique, pas facile





## Automatique et facile

```
uint32_t
Hacl_AEAD_Chacha20Poly1305_decrypt
uint8_t *output, uint8_t *input,
uint32_t input_len, uint8_t *data,
uint32_t data_len, uint8_t *key,
uint8_t *nonce, uint8_t *tag
```

Code – Hacl\_AEAD\_Chacha20Poly1305\_decrypt

Hacl\_AEAD\_Chacha20Poly1305\_decrypt.c

```
1 {
2   "input": "BUF_SIZE",
3   "input_len": "BUF_SIZE",
4   "output": "BUF_SIZE",
5   "output_len": "BUF_SIZE",
6   "key": "KEY_SIZE",
7   "nonce": "NONCE_SIZE",
8   "tag": "TAG_SIZE",
9   "data": "AAD_SIZE",
10  "data_len": "AAD_SIZE",
11  "BUF_SIZE": 16384,
12  "KEY_SIZE": 32,
13  "NONCE_SIZE": 12,
14  "AAD_SIZE": 12,
15  "TAG_SIZE": 16
16 }
```

Code – matching.json



## Coquille dans l'engrenage

```
1 {  
2     "*output": 41,  
3     "*tag": 12,  
4     "*input": 35,  
5     "input_len": 33,  
6     "*data": 14,  
7     "data_len": 14,  
8     "*key": 44,  
9     "*nonce": 15,  
10    "*expanded_key": 2,  
11    "*cipher": 6,  
12    "*plain": 16,  
13    "*a": 112,  
14    "*b": 107,  
15    "*res": 97,  
16    "*n": 70,  
17    "bBits": 28,  
18    "*k": 79,  
19    "len": 64,  
20 }
```

Code – counting.json





## Besoin de plus d'information

```
1 /**
2  Write `a + b mod 2^256` in `res`.
3
4  This functions returns the carry.
5
6  The arguments a, b and res are meant
   to be 256-bit bignums, i.e.
   uint64_t[4]
7 */
8 uint64_t Hacl_Bignum256_add(uint64_t *a
   , uint64_t *b, uint64_t *res);
```

Code – Hacl\_Bignum256.h

```
1 /**
2  Write `a * b` in `res`.
3
4  The arguments a and b are meant to be
   256-bit bignums, i.e. uint64_t
   [4].
5  The outparam res is meant to be a
   512-bit bignum, i.e. uint64_t[8].
6 */
7 void Hacl_Bignum256_mul(uint64_t *a,
   uint64_t *b, uint64_t *res);
```

Code – Hacl\_Bignum256.h



## Affectation de signature

Fonction extraite d'un .h

```
1 {  
2   "input_8_encrypt": "BUF_SIZE"  
3   , "input_len_32_encrypt": "  
4     BUF_SIZE"  
5   , "output_8_encrypt": "BUF_SIZE"  
6     "  
7   , "key_8_encrypt": "KEY_SIZE"  
8  
9   "BUF_SIZE": 16384,  
10  "KEY_SIZE": 32,  
11 }
```

Code – matching.json

```
1 {  
2   "Chacha20Poly1305_encrypt": "encrypt"  
3   , "Chacha20Poly1305_decrypt": "  
4     encrypt"  
5   , "32_add": "32_add"  
6   , "32_sub": "32_add"  
7   , "32_add_mod": "32_add"  
8   , "32_sub_mod": "32_add"  
9   , "32_mul": "32_mul"  
10  , "32_sqr": "32_mul"  
11  , "32_mod": "32_mod"  
12 }
```

Code – twin.json

function-tested.c



## Création du tampon

### Besoin d'identification

Format des données :

- ▶ Nom de la fonction
  - *Hacl\_family\_[op]*
- ▶ Type de données
  - *uint[X]\_f*



## Création du tampon

### Besoin d'identification

Format des données :

- ▶ Nom de la fonction
  - *Hacl\_family\_[op]*
- ▶ Type de données
  - *uint[X]\_f*

### Tampon unique

parametre\_X\_op

```
1  "a_32_32_add": "BUF_SIZE_8"  
2  , "b_32_32_add": "BUF_SIZE_8"  
3  , "res_32_32_add": "BUF_SIZE_8"  
4  , "n_32_32_add": "BUF_SIZE_8"  
5  , "BUF_SIZE_8": 8  
6  , "a_32_32_mul": "BUF_SIZE_8"  
7  , "b_32_32_mul": "BUF_SIZE_8"  
8  , "res_32_32_mul": "BUF_SIZE_16"  
9  , "BUF_SIZE_16": 16
```


Code – matching.json

```
1  "32_add": "32_add"  
2  , "32_sub": "32_add"  
3  , "32_add_mod": "32_add"  
4  , "32_sub_mod": "32_add"  
5  , "32_mul": "32_mul"  
6  , "32_sqr": "32_mul"
```

Code – twin.json

# 03

## Risc-V me voilà





## À partir de rien

### Conception du compilateur croisé

Depuis la source officielle :

► <https://github.com/riscv-collab/riscv-gnu-toolchain>



## À partir de rien

### Conception du compilateur croisé

Depuis la source officielle :

- ▶ <https://github.com/riscv-collab/riscv-gnu-toolchain>
- ▶ Après 3h et des poussières, ajout du compilateur : **riscv64-unknown-linux-gnu-gcc**



# Analyse de Binsec

## Le fameux p256

Modifications du script `.ini` :

1. Section `".data"` -> `".sdata"`
2. Ajout de l'adresse de fin

```
[sse:warning] Enumeration of jump targets @ 0x000104ae hit the limit 3 and may be incomplete  
[sse:warning] Cut path 3 (non executable) @ 0x00000000  
[sse:warning] Cut path 2 (non executable) @ 0xffffffffffffffff  
[sse:warning] Cut path 1 (non executable) @ 0xfffffffffffffffe
```

[Code – study.ini](#)

*Analyse corrigé en direct*






# Plan pour Risc-V

## Objectifs fixés

- ▶ Étendre l'analyse Binsec sur les tests natifs de Hacl\*
- ▶ Intégrer la chaîne de compilation à Érysicthon
- ▶ Ajouter une compilation vers l'architecture 32 bits
- ▶ Ajouter la compilation via Clang et LLVM

# 04

## Conclusion





# Conclusion

## Automatisation

- ▶ Continuer la génération des fichiers `-test.c`
- ▶ Activer la chaîne de compilation



# Conclusion

## Automatisation

- ▶ Continuer la génération des fichiers `-test.c`
- ▶ Activer la chaîne de compilation

## Toolchain Risc-V

- ▶ Compilation d'autres toolchain : 32 bits
- ▶ Compiler plus de `-test.c`
- ▶ Effectuer plus d'analyse Binsec



# Conclusion

## Automatisation

- ▶ Continuer la génération des fichiers `-test.c`
- ▶ Activer la chaîne de compilation

## Toolchain Risc-V

- ▶ Compilation d'autres toolchain : 32 bits
- ▶ Compiler plus de `-test.c`
- ▶ Effectuer plus d'analyse Binsec

## Rédaction du mémoire

- ▶ Poser les idées
- ▶ Concevoir le puit
- ▶ Plan du plan le 10/06

*Merci.*

