Réunion flash

Point hebdomadaire

Duzés Florian

Sommaire

01
Informations supplémentaires

Point d'informations

- Semaine du 30 juin
- Semaine du 14 juillet

02
État des lieux

Point actuel

Fait

- Étendu test de p256 avec LLVM / 32 / 64
- -test.json / generic json
- Hacl* -> RiscV

En cours

- Compiler Hacl* vers RiscV
- Test Binsec sur RiscV
- -test.json

Prévus

- Chaîne de bout en bout
- Couverture des architectures différentes
- Couverture des compilateurs

Réalisation

Fait → **En cours** → **Prévus**

Fait

- Mise à jour de Binsec

En cours

- Remplir les config
- Gérer les dépendances
- Activer chaîne de compilation
- Compilation en x86_64

Prévus

- Chaîne de bout en bout
- Couverture des architectures différentes
- Couverture des compilateurs

03

Écrire les config et les *Typedef*

Au commencement...

```c
/* MIT License*/

#ifndef __Hacl_Chacha20_H
#define __Hacl_Chacha20_H

#if defined(__cplusplus)
extern "C" {
#endif

#include <string.h>
#include "krml/internal/types.h"
#include "krml/lowstar_endianness.h"
#include "krml/internal/target.h"

void
Hacl_Chacha20_chacha20_encrypt(
  uint32_t len,
  uint8_t *out,
  uint8_t *text,
  uint8_t *key,
  uint8_t *n,
```

```c
  uint32_t ctr
);

void
Hacl_Chacha20_chacha20_decrypt(
  uint32_t len,
  uint8_t *out,
  uint8_t *cipher,
  uint8_t *key,
  uint8_t *n,
  uint32_t ctr
);

#if defined(__cplusplus)
}
#endif

#define __Hacl_Chacha20_H_DEFINED
#endif
```

Code – Hacl_AEAD_Chacha20Poly1305.h

… le néant

```
scheme_C = r'#if defined\(__cplusplus\)\s*\n(?:extern "C" \{\s*\n|}\s*\n)#endif\s*\n'
scheme_define = r'#.*?(?<!\\)\n'
scheme_comment = r'/\*.*?\*/'
scheme_debug = r'//'
scheme_type = r'typedef\s+(?:[^{}]|{(?:[^{}]|{[^{}]*})*})*?;'
scheme_krml = r'KRML_DEPRECATED'


#subduction
try:
    content = re.sub(scheme_C, '', content)
    content = re.sub(scheme_define, '', content, flags=re.DOTALL)
    content = re.sub(scheme_comment, '', content, flags=re.DOTALL)
    content = re.sub(scheme_type, '', content, flags=re.DOTALL)
    content = re.sub(rf'^{scheme_debug}.*;?', '', content, flags=re.MULTILINE)
    content = re.sub(rf'^{scheme_krml}.*;?', '', content, flags=re.MULTILINE)
```

Code – get_data.py

La simplicité - *28/05*

Fonction extraite d'un .h

```
{
  "Chacha20Poly1305_encrypt":"encrypt"
    ,"Chacha20Poly1305_decrypt":"
        encrypt"
    ,"32_add":"32_add"
    ,"32_sub":"32_add"
    ,"32_add_mod":"32_add"
    ,"32_sub_mod":"32_add"
    ,"32_mul":"32_mul"
    ,"32_sqr":"32_mul"
    ,"32_mod":"32_mod"
}
```

Code – twin.json

```
{
  "input_8_encrypt":"BUF_SIZE"
  ,"input_len_32_encrypt":"
      BUF_SIZE"
  ,"output_8_encrypt":"BUF_SIZE
      "
  ,"key_8_encrypt":"KEY_SIZE"

  "BUF_SIZE":16384,
  "KEY_SIZE":32,
}
```

Code – matching.json

function-tested.c

Les problèmes

```c
typedef struct
    Hacl_Hash_Blake2b_blake2_params_s
{
  uint8_t digest_length;
  uint8_t key_length;
  uint8_t fanout;
  uint8_t depth;
  uint32_t leaf_length;
  uint64_t node_offset;
  uint8_t node_depth;
  uint8_t inner_length;
  uint8_t *salt;
  uint8_t *personal;
}
Hacl_Hash_Blake2b_blake2_params;

typedef struct
    Hacl_Hash_Blake2b_index_s
{
  uint8_t key_length;
```

```c
  uint8_t digest_length;
  bool last_node;
}
Hacl_Hash_Blake2b_index;

Hacl_Hash_Blake2b_state_t
*H_H_Blake2b_malloc_with_params_and_key
    (
  Hacl_Hash_Blake2b_blake2_params *p,
  bool last_node,
  uint8_t *k
);
```

Code – Hacl_Hash_Blake2b.h

Solution

Remodéliser la génération des tests

- Identifier les *typedef*
- Ajouter dans les fichiers config
- Concevoir les tests en "pointant" vers ces définitions
- *Construire un compilateur C -> C*

Aperçu

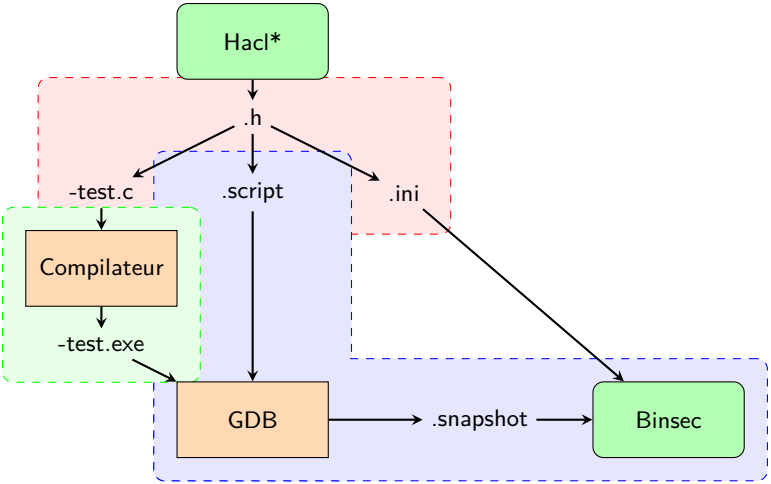
```
{"Meta_data":{
    "build" : "19-06-2025",
    "version" : "0.2.3"
}
,"Hacl_Bignum_MontArithmetic_bn_mont_ctx_u32": {
    "len":"BUFFER_SIZE"
  ,"*n":"BUFFER_SIZE"
  ,"mu":"BUFFER_SIZE"
  ,"*r2":"BUFFER_SIZE"
  ,"BUFFER_SIZE":8
}
,"Hacl_Bignum_MontArithmetic_bn_mont_ctx_u64": {
    "len":"BUFFER_SIZE"
  ,"*n":"BUFFER_SIZE"
  ,"mu":"BUFFER_SIZE"
  ,"*r2":"BUFFER_SIZE"
  ,"BUFFER_SIZE":8
}}
```

Code – Hacl_Bignum.json

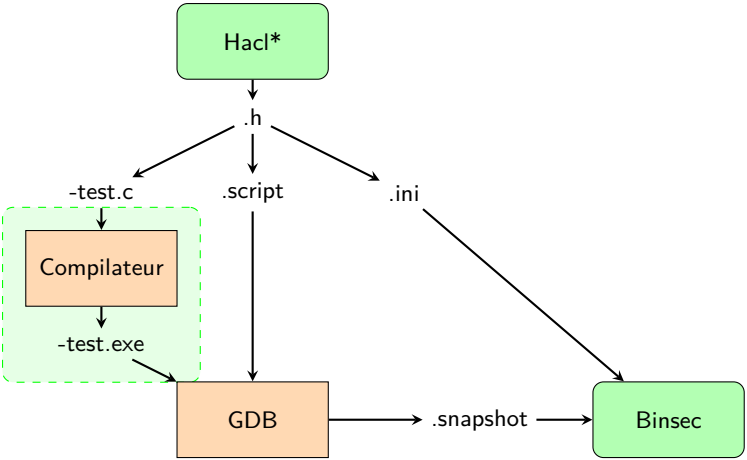
04

Compilation avec Érysichton

Structure


Hacl*
.h
-test.c
.script
.ini
Compilateur
-test.exe
GDB
.snapshot
Binsec

Structure


Hacl*
.h
-test.c
.script
.ini
Compilateur
-test.exe
GDB
.snapshot
Binsec

Modèle de compilation

Réplication de la compilation de Hacl*

```
Érysichton
├── Makefile
├── build.sh
├── x86_64
│   ├── Makefile.config
│   └── builds/
├── ARMV7
├── ARMV8
├── riscv-32
└── riscv-64
```

```
(##############################
        x86_64 section        #
##############################

6_64:
@./build.sh x86_64
@make --no-print-directory -f .Makefile
    .generated all
```

Code – Makefile

Modèle de compilation

Réplication de la compilation de Hacl*

```
Érysichton
├── Makefile
├── build.sh
├── x86_64
│   ├── Makefile.config
│   └── builds/
├── ARMV7
├── ARMV8
├── riscv-32
└── riscv-64
```

```
(DATA=/home/florian/Documents/recoules-
    hacl-star/hacl-star
ARCHI=x86_64

LDFLAGS="-Xlinker -z -Xlinker
    noexecstack -Xlinker --unresolved-
    symbols=report-all"

HELP=gcc
FORCE=Os
```

Code – Makefile.config

Modèle de compilation

Réplication de la compilation de Hacl*

```
Érysichton
├── Makefile
├── build.sh
├── x86_64
│   ├── Makefile.config
│   └── builds/
├── ARMV7
├── ARMV8
├── riscv-32
└── riscv-64
```

```
(# Makefile automatically generated
ARCHI := --target=x86_64
FORCE := -Os \ CC := gcc
CFLAGS := ... \ LDFLAGS := ...
BUILD_DIR := ... \ COMPIL_DIR := ...
SRC_DIR := make_tests/source/
SRC_FILES := Hacl_AEAD_Chacha20Poly1305_decrypt
    .c Hacl_AEAD_Chacha20Poly1305_encrypt.c

SOURCES := $(wildcard $(SRC_DIR)/*.c)
OBJECTS := $(patsubst $(SRC_DIR)/%.c,$(
    BUILD_DIR)/%,$(SOURCES))
EXECUTABLES := $(OBJECTS)

all:
  echo "$(SOURCES)"

%.exe: %.o
  $(CC) $(CFLAGS) $(LDFLAGS) $^ ../dist/gcc-
      compatible/libevercrypt.a -static -o $@
```

Code – build.sh

Reconceptualiser les intéractions

Communication entre Modules

- Fonctionnement pertinent
- Précision manuelle pour la compilation
- Activation de Binsec

05
Conclusion

Conclusion

Objectif

Finir le module x86_64.

- ☐ Remplir les configurations
- ☐ Générer les tests
- ☐ Compiler les tests
- ☐ Analyser les tests

Merci.