CHAPITRE 1

Références

TABLE 1.1 – Liste des options de compilations et leurs effets (non exhaustive), `https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`

Option de compilation	Effet
-O0	Compile le plus vite possible
-O1 / -O	Compile en optimisant la taille et le temps d'exécution
-O2	Comme -O1 mais en plus fort, temps de compilation plus élevé mais exécution plus rapide
-O3	Comme -O2, avec encore plus d'options, optimisation du binaire
-Os	Comme -O2 avec des options en plus, réduction de la taille du binaire au détriment du temps d'exécution
-Ofast	optimisations de la vitesse de compilation
-Oz	optimisation agressive sur la taille du binaire

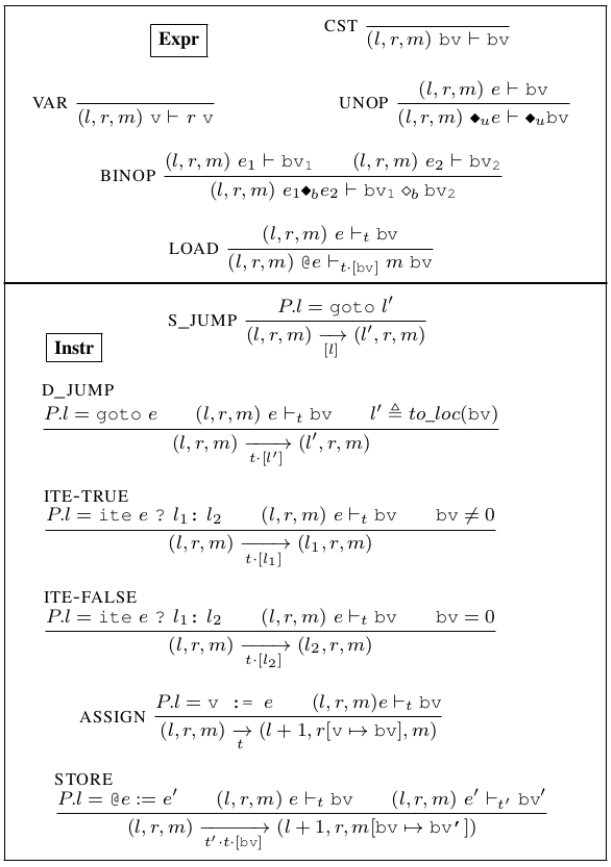

FIGURE 1.1 – Ensemble d'instructions défini formellement par [DBR19]

L'ensemble de ces règles a permis de décrire formellement le comportement de l'extension `checkct`, qui permet de vérifier les propriétés temps constants d'un programme.

CHAPITRE 2

Érysichthon, structure, exemples et résultats

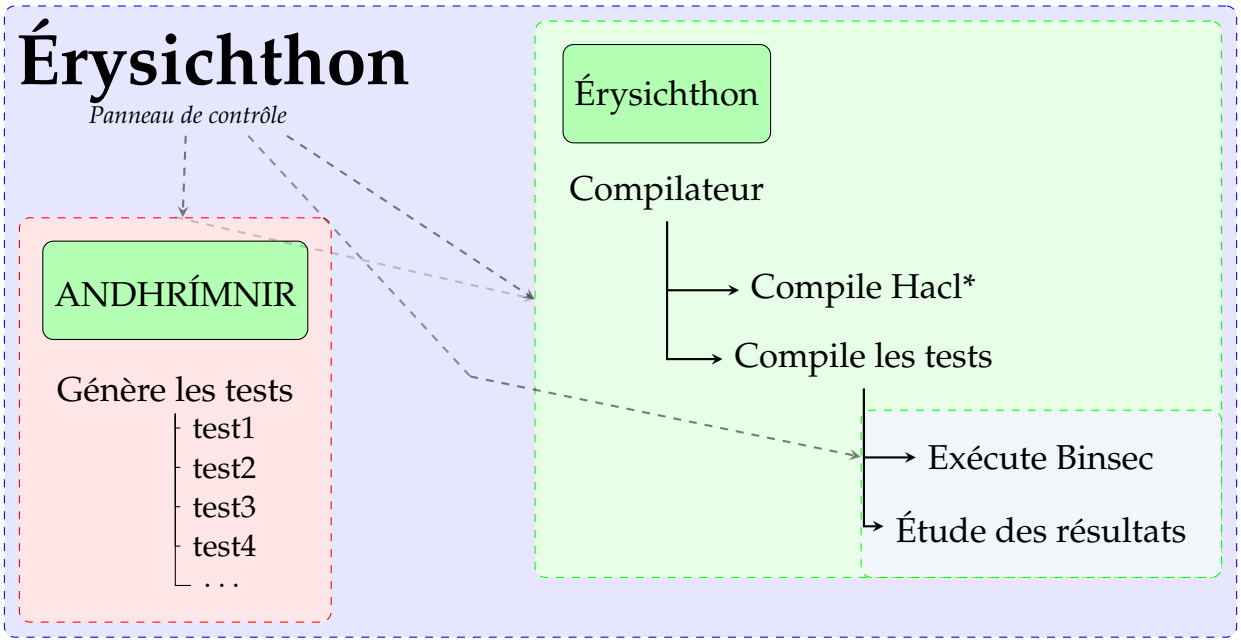

FIGURE 2.1 – Structure d'Érysichthon, schéma du point de vue de l'usager
Les flèches grises indiquent tous les éléments actionnables individuellement.

```c
/**
Encrypt a message `input` with key `key`.

The arguments `key`, `nonce`, `data`, and `data_len` are same in encryption/decryption.
Note: Encryption and decryption can be executed in-place, i.e.,
`input` and `output` can point to the same memory.

@param output Pointer to `input_len` bytes of memory where the ciphertext is written to.
@param tag Pointer to 16 bytes of memory where the mac is written to.
@param input Pointer to `input_len` bytes of memory where the message is read from.
@param input_len Length of the message.
@param data Pointer to `data_len` bytes of memory where the associated data is read from.
@param data_len Length of the associated data.
@param key Pointer to 32 bytes of memory where the AEAD key is read from.
@param nonce Pointer to 12 bytes of memory where the AEAD nonce is read from.
*/
void
Hacl_AEAD_Chacha20Poly1305_Simd256_encrypt(
  uint8_t *output,
  uint8_t *tag,
  uint8_t *input,
  uint32_t input_len,
  uint8_t *data,
  uint32_t data_len,
  uint8_t *key,
  uint8_t *nonce
);
```

Code 2.1 – Déclaration de la fonction **encrypt** dans le fichier d'en-tête Hacl_AEAD_-Chacha20Poly1305_Simd256.h

```json
{
"Meta_data":{
    "build" : "13-06-2025",
    "version" : "0.2.0"
}

,"Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt": {
    "*output":"BUF_SIZE"
   ,"*input":"BUF_SIZE"
   ,"input_len":"BUF_SIZE"
   ,"*data":"AAD_SIZE"
   ,"data_len":"AAD_SIZE"
   ,"*key":"KEY_SIZE"
   ,"*nonce":"NONCE_SIZE"
   ,"*tag":"TAG_SIZE"
   ,"BUF_SIZE":16384
   ,"TAG_SIZE":16
   ,"AAD_SIZE":12
   ,"KEY_SIZE":32
   ,"NONCE_SIZE":12
 }
}
```

Code 2.2 – Extrait du fichier Hacl_AEAD_Chacha20Poly1305_Simd256.json

```c
//
// Made by
// ANDHRÍMNIR - 0.5.4
// 12-08-2025
//

#include <stdlib.h>
#include "Hacl_AEAD_Chacha20Poly1305_Simd128.h"

#define BUF_SIZE 16384
#define TAG_SIZE 16
#define AAD_SIZE 12
#define NONCE_SIZE 12
#define KEY_SIZE 32
uint8_t output[BUF_SIZE];
uint8_t tag[TAG_SIZE];
uint8_t input[BUF_SIZE];
uint32_t input_len_encrypt = BUF_SIZE;
uint8_t data[AAD_SIZE];
uint32_t data_len_encrypt = AAD_SIZE;
uint8_t key[KEY_SIZE];
uint8_t nonce[NONCE_SIZE];


int main (int argc, char *argv[]){
Hacl_AEAD_Chacha20Poly1305_Simd128_encrypt(output, tag, input, input_len_encrypt,
 data, data_len_encrypt, key, nonce);
   exit(0);
}
```

Code 2.3 – Code généré du fichier test Hacl_AEAD_Chacha20Poly1305_Simd256_encrypt.c

```
starting from core

secret global output, input, data, key, nonce, tag
replace opcode 0f 01 d6 by
zf := true
end
replace opcode 0f 05 by
  if rax = 231 then
    print ascii "exit_group"
    print dec rdi
    halt
  end
  print ascii "syscall"
  print dec rax
  assert false
end
halt at <exit>
```

Code 2.4 – Instruction Binsec générée automatiquement, Hacl_AEAD_Chacha20Poly1305_Simd256_encrypt.ini

Optimisation	Secure	Unknown	Insecure
-O0	359	168	21
-O1	372	154	22
-O2	378	148	22
-O3	382	139	27
-Os	372	154	22
-Oz	373	153	22

TABLE 2.1 – Résultats d'Érysichthon en x86_64

Erreur / Option	-O0	-O1	-O2	-O3	-Os	-Oz
KO	0	0	2	2	0	0
syscall	0	0	0	0	0	0
max depth	92	59	45	23	58	59
killed	6	12	12	7	12	12
error	19	46	46	46	46	46
timeout	14	0	6	24	0	0
symbole	37	37	37	37	37	37

TABLE 2.2 – Tableau détaillant les erreurs interrompant l'analyse Binsec

Fonctions	Options concernées					
Hacl_EC_Ed25519_point_eq	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_ecdh	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_ecdsa_verify_hashed_msg	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_ecdsa_verify_sha256	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_is_public_key_valid	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_public_key_compressed_from_raw	O0					
Hacl_K256_ECDSA_public_key_uncompressed_to_raw	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_secp256k1_ecdsa_is_signature_normalized	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_secp256k1_ecdsa_signature_normalize	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_secp256k1_ecdsa_verify_hashed_msg	O0	O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_secp256k1_ecdsa_verify_sha256	O0	O1	O2	O3	Os	Oz
Hacl_NaCl_crypto_box_open_detached_afternm	O0	O1	O2	O3	Os	Oz
Hacl_NaCl_crypto_secretbox_open_detached	O0	O1	O2	O3	Os	Oz
Hacl_P256_compressed_to_raw	O0					
Hacl_P256_dh_responder	O0	O1	O2	O3	Os	Oz
Hacl_P256_ecdsa_verif_p256_sha2	O0	O1	O2	O3	Os	Oz
Hacl_P256_ecdsa_verif_p256_sha384	O0	O1	O2	O3	Os	Oz
Hacl_P256_ecdsa_verif_p256_sha512	O0	O1	O2	O3	Os	Oz
Hacl_P256_ecdsa_verif_without_hash	O0	O1	O2	O3	Os	Oz
Hacl_P256_uncompressed_to_raw	O0	O1	O2	O3	Os	Oz
Hacl_P256_validate_public_key	O0	O1	O2	O3	Os	Oz
Hacl_FFDHE_ffdhe_shared_secret_precomp		O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_secp256k1_ecdsa_sign_hashed_msg		O1	O2	O3	Os	Oz
Hacl_K256_ECDSA_secp256k1_ecdsa_sign_sha256		O1	O2	O3	Os	Oz
Hacl_NaCl_crypto_box_beforenm				O3		
Hacl_NaCl_crypto_box_detached				O3		
Hacl_NaCl_crypto_box_easy				O3		
Hacl_NaCl_crypto_box_open_detached				O3		
Hacl_NaCl_crypto_box_open_easy				O3		

TABLE 2.3 – Détails des fonctions non sécurisées en fonction des optimisations entrées, éxecution d'Érysichthon en x86_64

Résumé

texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte
texte texte texte texte texte texte texte texte texte texte texte texte