# TP introduction à la cryptographie

## AES: Comment ça marche?

### Introduction

Comme souvent, une bonne introduction sur un sujet peut être trouvée sur Wikipedia. Voici ce qu'on trouve sur l'AES :

The Advanced Encryption Standard (AES), also known by its original name Rijndael (Dutch pronunciation: ['rsinda:|]), is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

AES is a subset of the Rijndael <u>block cipher</u> developed by two <u>Belgian</u> cryptographers, <u>Vincent Rijmen</u> and <u>Joan Daemen</u>, who submitted a proposal to NIST during the <u>AES selection process</u>. Rijndael is a family of ciphers with different key and block sizes.

For AES, NIST selected three members of the Rijndael family, each with a **block size of 128 bits**, but three different key lengths: **128**, **192** and **256** bits.

AES has been adopted by the <u>U.S. government</u> and is now used worldwide. It supersedes the <u>Data Encryption Standard</u> (DES), which was published in 1977. The algorithm described by AES is a <u>symmetric-key algorithm</u>, meaning the same key is used for both encrypting and decrypting the data.

In the United States, AES was announced by the NIST as U.S. <u>FIPS</u> PUB 197 (FIPS 197) on November 26, 2001. This announcement followed a five-year standardization process in which fifteen competing designs were presented and evaluated, before the Rijndael cipher was selected as the most suitable (see <u>Advanced Encryption Standard process</u> for more details).

AES became effective as a federal government standard on May 26, 2002, after approval by the <u>Secretary of Commerce</u>. AES is included in the ISO/IEC 18033-3 standard. AES is available in many different encryption packages, and is the first (and only) publicly accessible <u>cipher</u> approved by the <u>National Security Agency</u> (NSA) for <u>top secret</u> information when used in an NSA approved cryptographic module (see <u>Security of AES</u>, below).

En résumé, AES est un algorithme cryptographique:

- Symétrique : le chiffrement et le déchiffrement utilisent la même clé « secrète »
- De type bloc : il traite des données de taille 128 bits (128 bits en entrée / 128 bits en sortie)
- Utilisant des clés de 128, 192 ou 256 bits (en fonction du niveau de sécurité souhaité)

L'objectif de ce TP est de réaliser une implémentation de l'algorithme de chiffrement AES en langage C. Pour cela, il va vous falloir maîtriser :

- le langage C (tableaux, structures, boucles, conditions...)
- l'anglais (pour lire la spécification...)
- les représentations polynomiales (???)
- les opérations sur les corps de Galois (aïe!)
- vos connexions synaptiques (aïe aussi!) ©
- votre esprit de synthèse

## Etape 1 : Découverte du standard

La vie étant bien faite, le standard AES est totalement défini dans un document de spécification.

La première chose à faire est certainement d'en prendre connaissance. Comme il est présent dans le dossier du TP, faîtes-vous plaisir ③ (20 minutes max).

## Etape 2 : Quelques rappels de C

Nous allons manipuler des variables qui peuvent faire 8 ou 32 bits.

Pour être tous d'accord, on va utiliser les types suivants :

typedef unsigned char u8; typedef unsigned short u16; typedef unsigned int u32;

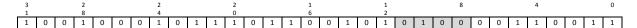
Si on tente de représenter une variable toto\_u8 de 8 bits valant 0xA3 (déclaration : u8 toto\_u8 = 0xA3)

 Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	0	0	0	1	1

Pour manipuler cette donnée, on va utiliser différents opérateurs :

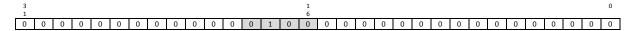
- Z = Y >> X : résultat Z = décalage de Y de X bits à droite
- Z = Y << X : résultat Z = décalage de Y de X bits à gauche
- Z = Y & X : résultat Z = et logique entre Y et X
- Z = Y | X : résultat Z = ou logique entre Y et X
- Z = Y ^ X : résultat Z = xor entre Y et X

De même, on peut représenter une variable tata\_u32 de 32 bits valant 0x98765432:



On va utiliser les mêmes opérateurs que précédemment :

```
tutu_u32 = ((tata_u32 >> 8) & 0xF) << 16
```



On va également manipuler quelques (beaucoup) de tableaux à 1 ou 2 dimensions :

- u8 mon\_tableau\_dimension\_1[nb\_colonnes]
- u8 mon\_tableau\_dimension\_2[nb\_lignes][nb\_colonnes]

La conversion décrite Figure 3 est réalisée en appliquant l'algorithme suivant :

### Par exemple, si input vaut :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0xAB	0xCD	0xEF	0x01	0x23	0x45	0x67	0x89	0xBA	0xDC	0xFE	0x10	0x32	0x54	0x76	0x98

### state vaudra:

	0	1	2	3
0	0xAB	0x23	0xBA	0x32
1	0xCD	0x45	0xDC	0x54
2	0xEF	0x67	0xFE	0x76
3	0x01	0x89	0x10	0x98

### Etape 3: Développement/intégration

Le temps imparti ne permet (malheureusement !?!) pas de comprendre tous les détails. Que peut-on retenir pour nous approcher de notre objectif ?

Le §5 est décidemment le plus intéressant!

- Il existe une fonction de chiffrement (§5.1) et une fonction de chiffrement inverse (§5.2).
   C'est une bonne nouvelle : on pourra déchiffrer les messages qu'on a chiffrés...
  - Discussion philosophique : Est-ce vraiment nécessaire ?
- Les versions AES128, AES192, AES256 se différencient par un certain nombre de paramètres (Nb, Nk, Nr).
- Il existe une notion de « key scheduling » permettant d'obtenir une clé de tour.

Les paramètres (Nk, Nb, Nr) sont définis en figure 4. Nous allons travailler sur le cas AES avec une clé de 128 bits (Nr = 10 tours).

#define Nk 4 #define Nb 4 #define Nr 10

La tâche est colossale pour atteindre notre objectif : nous avons environ 200-300 lignes de codes à écrire... Nous allons donc nous organiser pour répartir la charge de travail :

- Il y aura des développeurs de fonctions élémentaires
- Il y aura des intégrateurs de ces fonctions élémentaires pour créer la fonctionnalité globale

Pour que ça fonctionne, il va falloir :

- être forts...
- communiquer

#### Communiquer, ça signifie:

- définir qui fait quoi (je vais aider en distribuant les rôles et responsabilités...)
- définir les interfaces (je vais aider en les tentant de les imposer mais, conseil, validez entre développeurs et intégrateurs vos prototypes de fonctions/formats de données)
- définir un processus de livraison :
  - o il faut que les développeurs préviennent les intégrateurs de leurs livraisons
  - o il faut que les développeurs livrent des fonctions élémentaires validées
- respecter le planning (parce qu'on ne va pas y passer la nuit...)

### Les intégrateurs vont devoir créer les fonctions suivantes:

void Cipher(u8 in[4\*Nb], u8 out[4\*Nb], u32 sub\_key[Nb\*(Nr+1)])
void InvCipher(u8 in[4\*Nb], u8 out[4\*Nb], u32 sub\_key[Nb\*(Nr+1)])

Pour le chiffrement, l'algorithme est décrit en §5.1 et vous pouvez le vérifier avec les données fournies en §Appendice B ou §Appendice C.1.

Pour le déchiffrement, l'algorithme est décrit en §5.3 et vous pouvez le vérifier avec les données fournies en §Appendice C.1

#### Les développeurs vont devoir créer les fonctions suivantes :

```
void SubBytes(u8 in_state[4][4], u8 out_state[4][4])
void ShiftRows(u8 in_state[4][4], u8 out_state[4][4])
void InvSubBytes(u8 in_state[4][4], u8 out_state[4][4])
void InvShiftRows(u8 in_state[4][4], u8 out_state[4][4])
void AddRoundKey(u8 in_state[4][4], u32 sub_key[4], u8 out_state[4][4])
u8 GF256_mult(u8 a, u8 b)
void MixColumns(u8 in_state[4][4], u8 out_state[4][4])
u32 RotWord(u32 in_word)
u32 SubWord(u32 in_word)
void KeyExpansion(u8 key[4*Nk], u32 w[Nb*(Nr+1)])
```

#### **Quelques tips**

- Le fichier tp\_aes\_begin.c vous fournit en guise de cadeau de bienvenue un certain nombre de fonctions vous permettant conversion et affichage en tout genre.
  - Observez les bien et n'hésitez pas à les instancier pour être certains de leur fonctionnement !
- La fonction **MixColumns** est définie en (5.6). La multiplication est une multiplication dans le corps de Galois GF(2<sup>8</sup>) = GF(256) (§4.2).
  - Assurer vous (manuellement) que  $\{57\} \bullet \{83\} = \{c1\}$
- Assurez-vous, en considérant que X et Y quelconques, que :
  - X = InvSubBytes(SubBytes(X))
  - X = InvShiftRows(ShiftRows(X))
  - X = InvMixColumns(MixColumns(X))
  - X = AddRoundKeys(AddRoundKeys(X,Y),Y)
- La description du paramètre Rcon est une nouvelle fois mystérieuse. Comme on approche de Noël, sa définition est donnée dans le fichier de départ.
- Vous pouvez vérifier le fonctionnement de votre code en vous référant au **§Appendice**.
- La fonction main qui vous est gracieusement offerte doit vous permettre de vérifier le fonctionnement de vos implémentations!

### Etape 4: Validation

Bravo! Maintenant qu'on dispose de nos belles fonctions, on peut faire quelques essais.

Par exemple, on pourrait chiffrer les messages suivants :

#### Et alors?

Que pourrait-on faire?