L3

29/04/2023

RAPPORT PROJET CRYPTOGRAPHIE

PRÉSENTÉ PAR

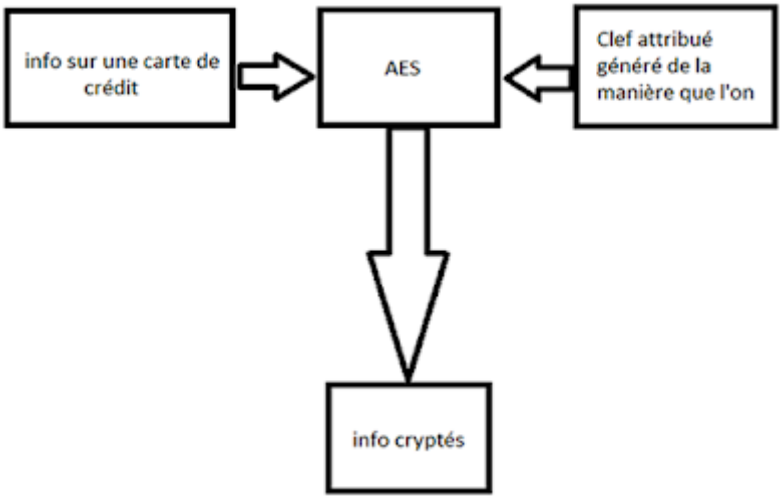
KASHI Mohand-Hedi
Weng Marc
Kbealy Yanis

Sommaire

Introduction

Qu’est-ce que l’AES ?

L'Advanced Encryption Standard (AES) est un algorithme de cryptage symétrique utilisé pour protéger les données sensibles. Il est considéré comme l'un des algorithmes de chiffrement les plus sécurisés actuellement disponibles. L'AES a été adopté en 2001 par le gouvernement américain en tant que standard de chiffrement pour les communications gouvernementales sensibles. Depuis lors, il est devenu un standard mondial pour la sécurité des données et est largement utilisé dans de nombreuses applications, telles que les systèmes de paiement en ligne, les services bancaires et financiers, les réseaux privés virtuels (VPN), les systèmes de stockage cloud, etc. L'algorithme AES utilise une clé de chiffrement de longueur variable (128, 192 ou 256 bits) pour chiffrer et déchiffrer les données. Il utilise une technique de substitution-permutation pour mélanger les données et ajouter de l'entropie, rendant ainsi la tâche de casser le chiffrement extrêmement difficile.



Objectif du Projet :

L’objectif de notre projet est d’implémenter une méthode de cryptage AES à 4 tours en Python puis d’effectuer une attaque de type « Square Attack » qui seront tout 2 détaillé dans leurs parties respectives

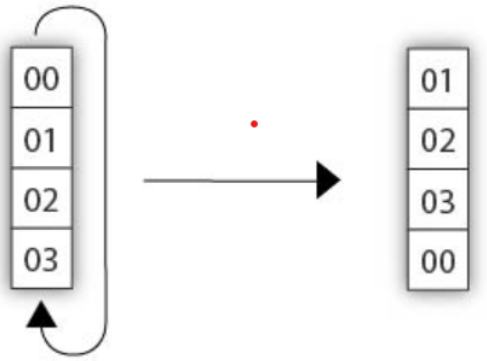
AES à 4 TOURS

1) Implémentation

Pour implémenter la Fonction **The Key Expansion** on va créer plusieurs sous fonctions afin de faciliter l'implémentation de cette dernière.

Etape 1 : Rotword

La fonction **Rotword** doit prendre en entrée 4 octet et doit retourner une rotation de ces 4 octets de la même manière que le schéma ci-dessous



Etape 2 : SubWord

La fonction **SubWord** prend une entrée de 4 octets comme la fonction précédente. Elle est essentiellement une Sbox, chaque octet est vérifié par rapport à une table de recherche et remplacé par sa valeur associée.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1.	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2.	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3.	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4.	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5.	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6.	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8

Etape 3 : Rcon

La dernière fonction d'assistance **Rcon** prend un entier en entrée et renvoie un tableau de 4 octets avec les 3 octets les moins significatifs mis à 0. Afin de faciliter l'implémentation, nous avons décidé de choisir la méthode de la boîte. En effet, la fonction renvoie la valeur du tableau par rapport à l'entrée, qui a été donnée.

```
var rcon = [256]byte{
0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4
0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x9
0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1
0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6
0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xf
0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x6
0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x8
0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xc
0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4
0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x1
0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x9
0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc
0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x0
0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xb
0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xc
0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xc
}
```

Etape 4 : Key Expansion

Les sous-fonctions nécessaires à la création des clés sont finies. Nous avons maintenant besoin d'une fonction produisant ses clés. Afin d'effectuer les opérations pour générer les clés, nous avons besoin de séparer la clé maître en 4 parties, cette séparation sera aussi utiliser pour produire les autres clés. Les 4 colonnes correspondent donc aux 4 parties de la clé.

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

Clé : 2b7e151628aed2a6abf7158809cf4f3c

La clé précédente est utilisée pour générer la clé suivante.

Afin de créer la 1ère colonne, nous devons effectuer différentes opérations avec la clé précédente:

- Une rotation à gauche de la dernière colonne de la clé
- Une substitution de cette colonne
- Un XOR du résultat de la substitution avec la première colonne de la clé
- un XOR entre le résultat de l'étape précédente et le résultat de la fonction **Rcon**, où l'argument passé est le numéro du tour en cours

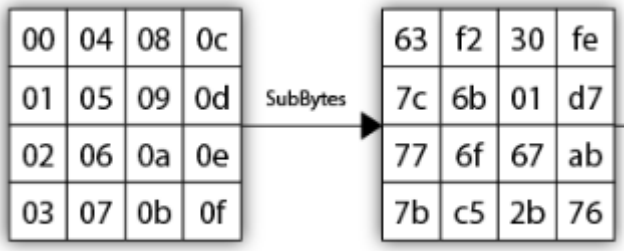
Pour créer les 3 colonnes restantes de la clé, il faut effectuer un XOR en décalage entre la clé précédente C1[x] et la clé C2[x], qui est en train d'être généré, où x est le numéro de la colonne :

- C2[2] = C1[2] XOR C2[1]
- C2[3] = C1[3] XOR C2[2]
- C2[4] = C1[4] XOR C2[3]

L'expansion se termine quand 10 clés ont été généré, nous nous retrouvons donc avec 11 clés au total.

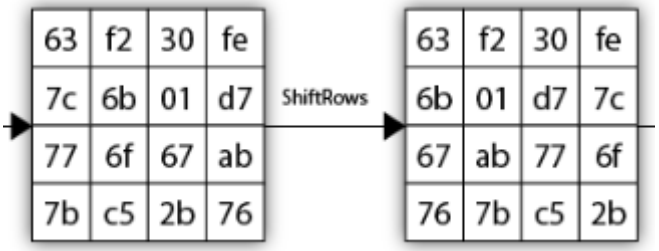
Etape 5 : SubBytes

Cette fonction ressemble à **SubWord**. En effet la fonction va reprendre une **Sbox** et va convertir l'entier entrée en argument en un nouveau entier en fonction d'une table de correspondance



Etape 6 : ShiftRow

ShiftRows regarde ses lignes et les fait pivoter. La première ligne n'est pas touchée, la seconde est tournée d'une position à gauche, la troisième de deux positions et la quatrième de trois positions.



Etape 7 : MixColumn

La fonction **MixColumn** effectue une multiplication matricielle avec une matrice pré-établit par le standard de l'AES, mais en remplaçant l'addition par des XOR.

Mais dans le cadre du code, nous avons préféré utiliser plusieurs listes contenant déjà le résultat des multiplications pour simplifier l'implémentation, nous avons donc juste à effectuer les XOR.

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Chaque colonne est alors multipliée par cette matrice ci-dessus afin d'obtenir une nouvelle colonne, qui remplacera celui-ci.

Exemple :

Prenons comme état :

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

Nous allons tout d'abord effectuer la multiplication de la 1ère colonne de l'état avec la 1ère ligne de la matrice :

$$2 * 2b \oplus 3 * 7e \oplus 1 * 15 \oplus 1 * 16$$

Nous devons changer toutes ses valeurs en leur représentation binaire, nous allons aussi associer une puissance X^n, où n est la position du bit, donc avec la 1ère opération $2 * 2b$:

$$0000\ 0010 * 0010\ 1011 \rightarrow X^1 * (1 + X^1 + X^3 + X^5) \rightarrow X^1 + X^2 + X^4 + X^6 \rightarrow 0101\ 0110$$

Nous obtenons donc 56 comme 1ère valeur de la colonne 1, nous continuons jusqu'à remplir l'entièreté des colonnes pour obtenir un nouvel état.

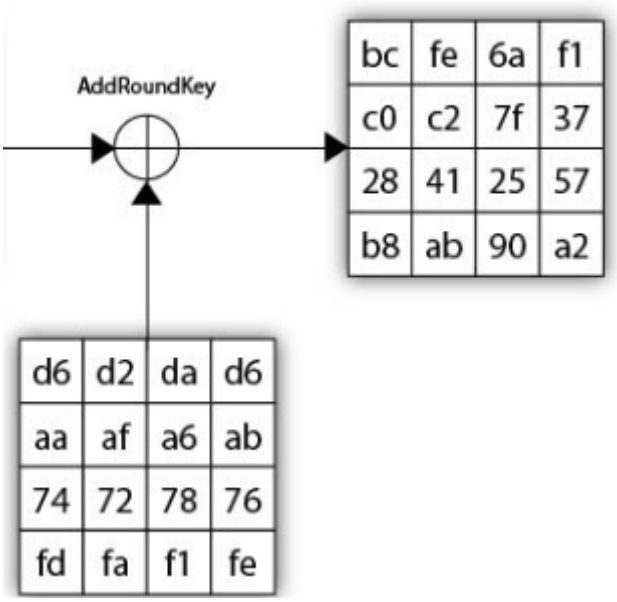
Dans le cas où il y'aurait un X^n avec un $n \geq 8$, nous utiliserons l'égalité :

$$X^8 + X^4 + X^3 + X + 1 = 0$$

Afin d'avoir un X^n avec un $n \leq 7$.

Etape 8 : AddRoundKey

La fonction **AddRoundKey** est juste comme on pouvait l'attendre au vue de la fonction précédente un Xor entre cette fois ci la matrice de valeur en sortie de la fonction MixColumns et la valeur de votre round key.



Etape 9 : Encrypt

La fonction **Encrypt** revient à pouvoir effectuer le chiffrement AES à 4 tours. En effet la fonction Encrypt récupère la matrice de valeur de la fonction **Mixcolumn** du tour précédent afin de pouvoir l'utiliser comme valeur d'entrée pour le tour précédent excepté si nous sommes au dernier tour où il n'y a pas de **Mixcolumn**.

Square Attack

Qu’est ce que la Square Attack ?

La "Square Attack" est une technique d'attaque cryptographique utilisée pour évaluer la sécurité des systèmes de chiffrement basés sur la cryptographie asymétrique. Elle tire son nom de l'exploitation de la vulnérabilité connue sous le nom de "problème de la racine carrée" dans certains schémas cryptographiques.Dans de nombreux schémas de chiffrement asymétrique, tels que le chiffrement basé sur la factorisation ou le chiffrement basé sur les courbes elliptiques, une opération mathématique appelée "extraction de la racine carrée" est utilisée pour effectuer des calculs cryptographiques. La Square Attack exploite le fait que, dans certains cas, cette opération de racine carrée peut être inversée pour retrouver la clé privée ou d'autres informations sensibles.

L'attaque consiste à collecter un grand nombre de paires de texte clair-chiffré (connus sous le nom de paires de "plaintext-ciphertext") et à analyser les relations entre ces paires. En appliquant des techniques d'analyse statistique et en exploitant les propriétés mathématiques des opérations de racine carrée, l'attaquant peut tenter de retrouver des informations sur la clé privée.

La Square Attack est considérée comme une attaque à chiffre choisi, ce qui signifie que l'attaquant peut choisir les textes clairs utilisés lors de l'attaque. Cette caractéristique la rend potentiellement plus puissante que certaines autres attaques qui reposent sur des textes clairs aléatoires.

Comment implémenter l’attaque ?

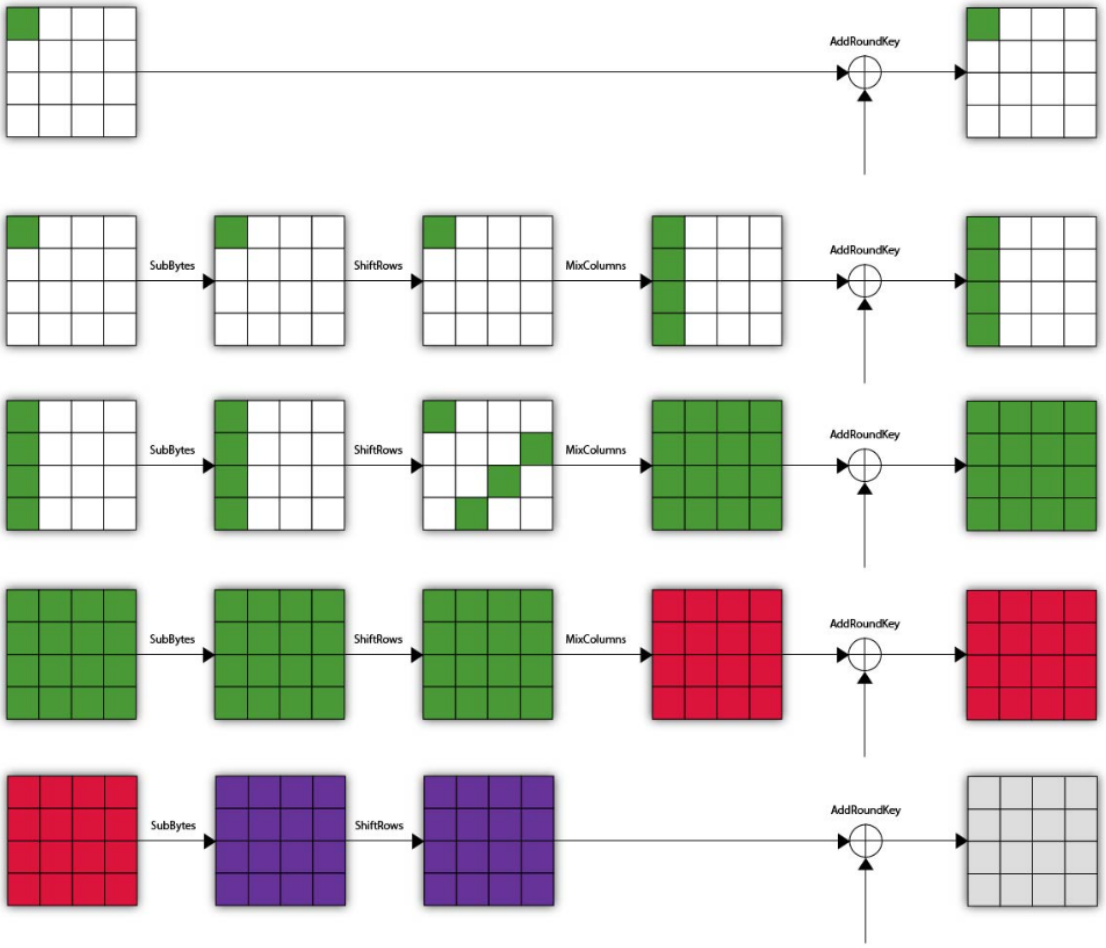
Rappel : L’AES à 4 tours n’a pas de transformation avec **MixColumns** au dernier tour. Nous allons donc dans un premier temps de trouver la clé du deuxième tour

L’attaque se déroule en 2 étapes majeurs:

On doit créer une fonction nommée **ReverseState** qui prend une estimation de clé d'un octet, la position de cette estimation de clé et le Λ-set chiffré renvoyé par la setup()fonction. Il devrait alors inverser l'octet à cette position sur chaque élément du Λ-set , jusqu'au début du dernier tour. Il devrait alors renvoyer cet ensemble d'octets inversés.

On implémente ensuite une fonction nommée **CheckKeyGuess** qui prend l'estimation de clé d'un octet et l'ensemble des valeurs d'octet renvoyées par la **ReverseState**. La fonction doit essayer de XOR tous les octets donnés et vérifier si le résultat est égal à 0. Si c'est le cas elle doit stocker dans une liste

Nous devons maintenant répéter cette étape plusieurs fois à l’aide de **Search_SybKey4.** La fonction génère 16 listes, par rapport au delta set, contenant les bytes actifs avec une position différente pour chacune d'entre elles. Tout d'abord, nous devons effectuer 16 opérations correspondant aux 16 parties de la clé. Pour chaque partie de la clé, on effectue un retour vers la version de la fin du tour 3 de l'AES avec toutes les valeurs possibles en utilisant la fonction **ReverseState**. Toutes les valeurs possibles, qu'on appellera byte hypothétique, sont des bytes recouvrent les valeurs entre 0x00 et 0xFF, chaque liste contenant les 256 bytes est passé par la fonction **CheckKeyGuess** pour vérifier si le XOR de tous les éléments est égal à 0. Si c'est le cas, alors le byte hypothétique est ajouté à une liste. Cette liste L2 est contenu dans une autre liste L1, ou la L1[L2] correspond à la position choisie.On se retrouve alors avec une liste, contenant 16 listes, qui possèdent chacunes des entiers, qui ont réussi à passer l'égalité.



Dernière étape :

Après avoir connu La clé du dernier tour de notre instance, nous devons terminer le travail et inverser des clés pour obtenir **la clé principale** . La fonction **InvertKeyExpansion** prend un index (l'index de la clé ronde) et la clé ronde, et renvoie la clé AES d'origine.

Conclusion

En conclusion, l'implémentation de la méthode de cryptage AES (Advanced Encryption Standard) a apporté des connaissances essentielles dans le domaine de la sécurité des données. AES est devenu l'algorithme de cryptage symétrique le plus largement utilisé et accepté dans le monde entier.

Grâce à son adoption généralisée, nous avons appris que AES offre un niveau élevé de sécurité et de confidentialité pour les communications numériques et les systèmes de stockage de données. Sa conception robuste et ses propriétés mathématiques bien étudiées en font un choix fiable pour protéger les informations sensibles.

L'implémentation d'AES a également révélé l'importance de la gestion appropriée des clés de chiffrement. Les clés jouent un rôle crucial dans la sécurité du cryptage AES, et une mauvaise gestion des clés peut compromettre toute la sécurité du système. Par conséquent, nous avons appris qu'il est essentiel de mettre en place des protocoles solides pour la génération, le stockage et la distribution des clés.

Source

- https://www.davidwong.fr/blockbreakers/aes.html (Site web)

- Aes3602 (Fichier pdf)

- differential_cryptanalysis3603 (Fichier pdf)

- https://www.youtube.com/watch?v=5PHMbGr8eOA (Vidéo)

- https://www.youtube.com/watch?v=pSCoquEJsIo (Vidéo)