TD Advanced Encryption Standard

Rappels sur la théorie des ensembles

Définition 1 ***Groupe***
Un ensemble non vide G muni d'une loi de composition interne + est un groupe si :

- $\forall (x, y) \in G^2$, $X + Y \in G$ *(ensemble fermé)*
- $\forall x \in G$, $\exists 0 \in G$ *tq* $0 + x = x + 0 = x$ *(element neutre)*
- $\forall x \in G$, $\exists x^{-1} \in G$ *tq* $x + x^{-1} + 0$ *(inverse)*
- $\forall (a, b, c) \in G^3$, $a + (b + c) = (a + b) + c$ *(associatif)*

On le note (G, +).
Remarque : Un groupe est dit ***abélien*** *si la loi + est commutative* ($\forall (x, y) \in G,\ x + y = y + x$).

Définition 2 ***Anneau***
Un ensemble non vide A muni de deux lois de composition interne + et × est un anneau si :

- *(A, +) est un groupe abélien de neutre 0*
- *la loi × est associative*
- $\forall (a, b, c) \in A$,

  $$(a + b) \times c = a \times c + a \times c$$
  $$c \times (a + b) = c \times a + c \times b$$

  (distributivité)

On le note (A, +, ×).
Remarque 1 : Un anneau est dit ***commutatif*** *si la loi × est commutative.*
Remarque 2 : Un anneau est dit ***unitaire*** *si la loi × possède un élément neutre* $1 \in A$ *(appelé unité de l'anneau).*

Définition 3 $\mathbb{Z}/p\mathbb{Z}$
$\mathbb{Z}/p\mathbb{Z}$ *est un anneau défini par la relation de congruence sur les entiers. En particulier, les lois de relation internes sont les suivantes :*

Loi '+' : $a + b \mod p$

Loi '×' : $a \times b \mod p$

C'est un anneau qui possède un nombre fini d'éléments (il en possède p).

Définition 4 ***Corps***
Un corps K est un anneau unitaire tel que (K, ×) est un groupe. Autrement dit, il faut que tous les éléments d'un corps aient un inverse excepté l'élément nul.*
Remarque : Il est parfois mentionné que l'anneau doit être commutatif.

Corps finis

Un corps fini est un corps commutatif possédant un nombre fini d'éléments (cardinal fini). Son cardinal est toujours une puissance d'un nombre premier (appelé caractéristique), et il existe un corps pour tout nombre premier p élevé à une puissance n. Pour la suite, on s'intéressera plus particulièrement aux corps finis de caractéristique de la forme 2^n.

Remarque : Pour p premier, l'anneau $\mathbb{Z}/p\mathbb{Z}$ est un corps (la réciproque est vraie également).

Afin de créer de nouveaux corps finis, on utilise la structure d'anneau Euclidien $\mathbb{Z}_p[X]$. Pour construire un corps fini, de manière similaire aux corps $\mathbb{Z}/p\mathbb{Z}$, on choisi un polynôme irréductible $f(X)$ (qui fait office de nombre premier) et on construit le corps $\mathbb{Z}_p[X]/f(X)$.

Pour n le degré du polynôme $f(X)$, on note en général le corps fini $\mathbb{Z}_p[X]/f(X)$ avec la notation $\mathrm{GF}(p^n)$.

Corps finis et AES

Les opérations d'AES se font dans le corps fini $\mathbb{Z}_2[X]/m(x)$, où $m(x)$ vaut :

$$m(x) = x^8 + x^4 + x^3 + x + 1 \tag{1}$$

On note en général ce polynôme 11B, ce qui correspond à l'écriture en hexadécimal de la séquence de bits donné par la concaténation de la valeur de ses coefficients.

Question 1

Répondez aux questions suivantes :

1. Donnez le nom du corps fini associé en suivant la notation $\mathrm{GF}(p^n)$. Cette notation est-elle suffisante pour décrire entièrement le corps fini ?
2. Pourquoi avoir choisi un polynôme irréductible avec ce degré particulier ?
3. Comment les opérations d'addition sont réalisées ? Comment cette opération peut s'exécuter efficacement sur une machine ?
4. Comment les opérations de multiplication sont réalisées ? Comment cette opération peut s'exécuter efficacement sur une machine ?

Réponse

1. $\mathrm{GF}(2^8)$. Cette notation n'est pas suffisante car elle ne donne pas d'information sur le polynôme irréductible choisi.
2. Ce polynôme permet de faire des opérations sur des octets, opérations qui sont courantes sur les processeurs.
3. L'addition revient à faire des additions de polynôme dans $\mathbb{Z}_2[X]$, ce qui revient à faire une opération de ou-exclusif sur la chaîne de bit.
4. Le multiplication revient à faire des multiplications de polynôme dans $\mathbb{Z}_2[X]/m(X)$. Il n'y a pas de façon triviale de faire cette opération et demande d'exécuter un algorithme de réduction modulaire (polynomiale). Cette opération peut éventuellement se simplifier si on multiplie par une constante connue.

L'opération SubByte

SubByte fait partie des 4 opérations de base effectuée durant 1 tour d'AES. C'est une opération de substitution de symbole qui pour des raisons de sécurité et d'efficacité a été conçu pour respecter les propriétés suivantes :

1. inversibilité ;
2. minimisation de la plus grande corrélation entre une combinaison des entrée et une combinaison des sorties ;
3. minimisation de la plus grande valeur dans la table EXOR ;
4. complexité de représentation dans GF(2^8) ;
5. description simple.

Les auteurs d'AES sont partis de la transformation inverse, définie de la manière suivante :

$$F(x) = \begin{cases} x^{-1} & , si \quad x \neq 0 \\ 0 & , si \quad x = 0 \end{cases} \tag{2}$$

Question 2

Justifiez pourquoi cette substitution est bien applicable dans le cas d'AES.

Réponse

Les opérations d'AES se font sur GF(2^8). Comme celui-ci est un corps fini, tout élément a un unique inverse.

Question 3

Cette substitution est-elle suffisante pour respecter les propriétés énoncées plus haut ?

Réponse

Non, sa représentation est triviale dans GF(2^8).

Afin de renforcer la sécurité de la fonction de substitution, la transformation affine suivante est appliquée après l'opération de transformation inverse :

$$b(x) = (x^6 + x^5 + x + 1) + a(x)(x^7 + x^6 + x^5 + x^4 + 1) \mod x^8 + 1 \tag{3}$$

Question 4

Cette transformation (affine) est-elle inversible ?

Réponse

$x^8 + 1$ n'étant pas un polynome irreductible, il faut vérifier si $x^7 + x^6 + x^5 + x^4 + 1$ est premier avec $x^8 + 1$. Les 8 racines de $x^8 + 1$ etant 1, et 1 n'est pas racine de $x^7 + x^6 + x^5 + x^4 + 1$, il est bien inversible.

Question 5

Etudiez comment se comporte la réduction modulaire polynomiale par $x^8 + 1$. En déduire la représentation matricielle de l'opération de transformation affine sous la forme :

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} . \\ . \\ . \\ . \\ . \\ . \\ . \\ . \end{bmatrix}$$

où x_i (resp. y_i) correspond au i^{eme} coefficient du polynôme d'entrée (resp. de sortie) de la transformation affine.

Réponse

La représentation matricielle de la fonction affine est la suivante :

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (4)$$

Question 6

En quoi ce choix a-t-il renforcé la sécurité ?

Réponse

1. Cela supprime l'existance de point fixe (i.e. SubByte(a) = a. 2. La représentation de l'opération est triviale dans $\mathbb{Z}_2[X]/(x^8+1)$, mais ce n'est pas le cas si combinée avec la transformation inverse.

Question 7

Donnez les valeurs de **SubByte** pour les 3 premières valeurs.

Réponse

$$
\begin{aligned}
\textbf{SubByte}(0) &= x^6 + x^5 + x + 1 \\
&= 99 \text{ (en decimal)} \\
\textbf{SubByte}(1) &= (x^4 + x^3 + x^2 + x + 1) + (x^6 + x^5 + x + 1) \\
&= x^6 + x^5 + x^4 + x^3 + x^2 \\
&= 124 \text{ (en decimal)} \\
\textbf{SubByte}(2) &= 119 \text{ (en decimal)}
\end{aligned}
$$

Question 8

Comment l'implémentation de **SubByte** peut être optimisée ? Comment rapidement calculer $\textbf{SubByte}^{-1}$?

Réponse

Il suffit de pré-calculer toutes les valeurs et les stocker dans un tableau. Pour calculer $\textbf{SubByte}^{-1}$, il suffit d'inverser la table.

L'opération MixColumn

MixColumn est l'avant dernière opération effectuée durant un tour d'AES, la dernière étant l'addition avec la clé de tour. La conception de **MixColumn** a suivi les considérations suivantes :

1. inversibilité ;
2. linéarité dans GF(2^8) ;
3. propriété de diffusion suffisante ;
4. rapidité de calcul sur processeur 8 bits ;
5. symétrie ;
6. description simple.

Dans le cas du **MixColumn**, les colonnes sont considérées comme des polynômes à coefficients dans GF(2^8) réduits par le polynôme $x^4 + 1$. Pour a(x) le polynôme représentant une colonne, l'opération **MixColumn** est la suivante :

$$b(x) = c(x) \times a(x), \quad c(x) = 03 \times x^3 + 01 \times x^2 + 01 \times x + 02 \tag{5}$$

Question 9

Donnez la représentation matricielle de cette opération.

Réponse

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (6)$$

Question 10

Simplifiez les opérations 01 × a(x), 02 × a(x) et 03 × a(x).
Pour information : L'opération 02 × a(x) possède sa propre instruction. Elle s'appelle xtime.

Réponse

- $01 \times a(x)$: C'est une opération de copie de la valeur ;
- $02 \times a(x)$: $(a_6, a_5, a_4, a_3, a_2, a_1, a_0, 0) \oplus (0, 0, 0, a_7, a_7, 0, a_7, a_7)$
- $03 \times a(x)$: $a(x) \oplus 02 \times a(x)$

L'opération ShiftRows

Nous n'allons pas particulièrement étudier l'opération **ShiftRows**, mais vous pouvez noter que cette opération a été introduite pour empécher 2 attaques connues au moment de la soumission, à savoir les attaques basées sur la cryptanalyse différentielle tronquée ainsi qu'une attaque qui avait été trouvée sur une proposition précédente des auteurs d'AES (chiffrement carré).

Le nombre de tours

Question 11

En étudiant les différentes opérations successives, donnez le nombre de tour minimal pour obtenir une "diffusion complète", c'est-à-dire que chaque bit dépendent de tous les autres bits de l'état interne. Vous pourrez regarder la diffusion du premier bit de l'état interne.

Réponse

Il faut au minimum 2 tours. Si on suit par exemple la diffusion du premier bit de l'état interne :

1. **SubByte** permet de faire que le premier bit se diffuse sur le premier octet de l'état interne ;
2. **ShiftRows** n'a pas d'impact ;
2. **MixColumn** permet de faire diffuser le premier octet sur toute la colonne.

Pour le deuxième tour, **ShiftRows** permet de diffuser la première colonne sur toutes les autres, et **MixColumn** termine le travail de diffusion sur chaque colonne.