UNIVERSITEE DES SCIENCES ET DE LA TECHNOLOGIE

HOUARI BOUMEDTENNE

FACULTE DE MATHEMATIQUES

Résumé du Mémoire de Magister en Mathématique
Option : Algèbre et Théorie des Nombres
Présenté par [1]
BEKHOUCHE Saïda
Sujet

Fondements mathématiques et fonctionnement du standard de chiffrement avancé *Rijndael (AES)*

Résumé

Le ***Rijndael***, un algorithme de chiffrement par blocs itérés mis au point par deux cryptographes belges *J. Daemen* et *V. Rijmen*, fut choisi en Octobre 2000 par le NIST *(National Institute of Standards and Technology- USA)* en tant que standard de chiffrement avancé ***AES*** (*Advanced Encryption Standard*) , en remplacement du ***DES*** (*Data Encryption Standard*) dont la sécurité a été remise en cause par l'attaque dite «*brute force attack*» opérant sur l'espace de ses 2^{56} clés possibles.
En Juin 2003, le gouvernement américain a annoncé: «l'architecture et la longueur de toutes les tailles de clés de l'algorithme ***AES*** sont suffisantes pour protéger des documents classifiés jusqu'au niveau «*SECRET*».
Le niveau «*TOP SECRET*» nécessite des clés de 192 /256 bits »

[1] Directeur de thèse :HACHICHI Mohamed – Salah (Maître de Conférences – A)

INTRODUCTION

Le 15 mai 1973 le *NBS* des Etats-Unis (*National Bureau of Standards*, aujourd'hui appelé *National Institute of Standards and Technology- NIST*) a lancé un appel dans le *Federal Register* pour la création d'un algorithme de chiffrement répondant aux critères suivants :

- posséder un haut niveau de sécurité lié à une clé de petite taille servant au chiffrement et au déchiffrement
- être compréhensible

- ne pas dépendre de la confidentialité de l'algorithme
- être adaptable et économique
- être efficace et exportable

L'AES (Advanced Encryption Standard) est le nouveau standard de cryptage symétrique (a clé Secrète), destiné à remplacer le DES (Data Encryption Standard) .

Le 2 Octobre 2000, le ***NIST*** choisi le ***Rijndael*** comme support pour l'***AES***, un algorithme mis au point par deux belges, *J. Daemen* et *V. Rijmen*.

L'***AES*** procède par blocs de 128 bits avec des clés de longueur 128, 192 ou 256 bits. Chaque bloc (*state*) subit une séquence de cinq transformations répétées 10 (***AES 128***), 12 (***AES 192***) ou 14 fois (***AES 256***)

Ce mémoire est organisé en quatre chapitre . Dans le premier chapitre est consacré à une étude bibliographique. Dans le deuxième chapitre, on va présenté dans lequel nous développons les mathématiques utilisées de l'AES. Dans le troisième chapitre, nous présenterons le relatif à la description et au fonctionnement de l'AES. La dernière chapitre, on va présenté une application clôturera notre étude.

Dans le Chapitre 1, on a étudié* l'*étude Bibliographique

Le but traditionnel de la cryptographie est d'élaborer des méthodes permettant de transmettre des données de manière confidentielle selon les principes énoncés par *A. Kerckhoffs* [Ker] :

Une information cryptée ne doit en aucun cas pouvoir être déchiffrée sans la connaissance de sa clé,
Les interlocuteurs ne doivent pas subir de dégâts au cas où le système de cryptage serait dévoilé,
La clé doit être simple et modifiable à souhait,
Les cryptogrammes doivent être transportables,
Le système doit être simple d'utilisation,
Le cryptosystème doit être au préalable examiné par des experts.

La cryptographie moderne s'intéresse plus généralement aux problèmes de sécurité des communications, *partant du principe que tout secret est un point de cassure possible*. Le but est d'offrir un certain nombre de services de sécurité comme :

- **La confidentialité,**
- **L'intégrité ,**
- **L'authentification des données transmises,**
- **L'authentification d'un tiers,**
- **Le contrôle d'accès (autorisation ou non d'accès à des objets)**

On utilise à cet effet des techniques basées sur des algorithmes cryptographiques. Il existe deux grandes familles d'algorithmes de chiffrement : les algorithmes à clef secrète, ou algorithmes symétriques, et les algorithmes à clef publique, ou algorithmes asymétriques.

1.1- Chiffrement symétrique ou à clef secrète

Dans la cryptographie symétrique (*ou conventionnelle*), les *clefs de chiffrement et de déchiffrement sont identiques* : c'est la clef secrète, qui doit être connue des tiers communicants et d'eux seuls.

Les algorithmes symétriques sont de deux types :

- les algorithmes de *chiffrement en continu*, qui agissent sur le texte en clair un bit à la fois; - les algorithmes de *chiffrement par blocs*, qui opèrent sur le texte en clair par groupes de bits appelés blocs.

1.2- Modes de chiffrement par blocs

Les algorithmes de chiffrement par blocs peuvent être utilisés suivant différents modes, dont les deux principaux sont le mode ***ECB*** (*Electronic CodeBook*) et le mode ***CBC*** (*Cipher Block Chaining*).

1-2-1- Le mode *ECB* (*Electronic CodeBook*)

1-2-2- Le mode *CBC* (*Cipher Block Chaining*)

1.3- Chiffrement par blocs avec itération

Un algorithme de chiffrement par blocs avec itération est un algorithme qui chiffre les blocs par un processus comportant plusieurs rondes. *Dans chaque ronde, la même transformation est appliquée au*

bloc, en utilisant une sous-clef dérivée de la clef de chiffrement. En général, un nombre de rondes élevé garantit une meilleure sécurité, au détriment bien évidemment des performances. .

1.4- Exemples d'algorithmes symétriques

La méthode la plus employée pour concevoir un procédé de chiffrement est de chercher à réaliser une transformation suffisamment complexe et irrégulière pour que sa crypanalyse soit d'une complexité aussi élevée que possible. Cette méthode empirique, toutefois, ne fournit aucune garantie quant à la robustesse de l'algorithme résultant.

Des exemples d'algorithmes symétriques sont le ***DES*** (*Data Encryption Standard*), le ***DES triple à deux ou trois clefs***, l'***IDEA*** (*International Data Encryption Algorithm*), le ***RC5*** (*Rivest's Code 5*), etc… .

1.4.1- Masque jetable (One-Time-Pad)

1.4.2- DES (Data Encryption Standard)

1.5- Chiffrement asymétrique ou à clef publique

Avec les algorithmes asymétriques, *les clefs de chiffrement et de déchiffrement sont distinctes et ne peuvent se déduire l'une de l'autre*. On peut donc rendre l'une des deux clés, publique tandis que l'autre reste privée (*secrète*). C'est pourquoi on parle de chiffrement à clef publique. Si la clef publique sert au chiffrement, tout le monde peut chiffrer un message, que seul le propriétaire de la *clef privée* pourra déchiffrer. On assure ainsi la *confidentialité*. Certains algorithmes permettent d'utiliser la clef privée pour chiffrer. Dans ce cas, n'importe qui pourra déchiffrer, mais seul le possesseur de la clef privée peut chiffrer. Cela permet donc la signature de messages.

1.5.1- Algorithmes à empilement.

1.5.2- RSA

1.6- Générateurs aléatoires et pseudo-aléatoires

La cryptographie a souvent recours à des nombres aléatoires. Ainsi, lorsqu'une personne génère une clef secrète (*ou privée*), elle doit faire intervenir le hasard de façon à empêcher un adversaire de deviner la clef. De même, certains protocoles cryptographiques nécessitent, pour éviter la re-jouabilité par exemple, l'utilisation d'aléas imprévisibles par les attaquants.

Malheureusement, il est impossible de produire des suites aléatoires à l'aide uniquement d'un ordinateur : le générateur sera toujours périodique, donc prévisible. On a donc recours à des générateurs dits pseudo-aléatoires et cryptographiquement sûrs. Un tel générateur doit présenter les caractéristiques suivantes :

1. La période de la suite doit être suffisamment grande pour que les sous-suites finies utilisées avec l'algorithme ou le protocole cryptographique ne soient pas périodiques.
2. Ces sous-suites doivent, sur le plan statistique, être aléatoires.
3. Le générateur doit être imprévisible, au sens où il doit être impossible de prédire le prochain aléa à partir des aléas précédents.

1.7- Fonctions de hachage à sens unique, signature numérique et scellement

Aussi appelée fonction de condensation, une *fonction de hachage* est une fonction qui convertit une chaîne de longueur quelconque en une chaîne de taille inférieure et généralement fixe; la chaîne résultante est appelée *empreinte* (*digest* en anglais) ou condensé de la chaîne initiale.

Des exemples de fonctions ainsi conçues sont ***MD5, SHA, RIPE-MD.***

1.7.1- MD5 (Message Digest 5)

1.7.2- SHA (Secure Hash Algorithm)

1.7.3- RIPE-MD

1.8- Intégrité et authentification de l'origine des données

Parmi les problèmes auxquels s'attaque la cryptographie, on trouve l'authentification de l'origine des données et l'intégrité : lorsque l'on communique avec une autre personne au travers d'un canal peu sûr, on cherche à ce que *le destinataire puisse s'assurer que le message émane bien de l'auteur auquel il est attribué et qu'il n'a pas été altéré pendant le transfert.*

1.9- Signature numérique

La norme [ISO 7498-2] définit la signature numérique comme des "*données ajoutées à une unité de données, ou transformation cryptographique d'une unité de données, permettant à un destinataire de prouver la source et l'intégrité de l'unité de données et protégeant contre la contrefaçon (par le*

destinataire, par exemple)". La mention "*protégeant contre la contrefaçon*" implique que seul l'expéditeur doit être capable de générer la signature.

1.9.1- DSS

1.9.2- RSA

1.9.3- Scellement

1.9.4- HMAC

1.10- Protocoles cryptographiques

Tout comme les protocoles de communication, les protocoles cryptographiques sont une série d'étapes prédéfinies, basées sur un langage commun, qui permettent à deux ou plusieurs participants d'accomplir une tâche. Dans le cas des protocoles cryptographiques, les tâches en question sont bien sûr liées à la cryptographie: ce peut être une authentification, un échange de clef,... Une particularité des protocoles cryptographiques est que les tiers en présence ne se font généralement pas confiance et que le protocole a donc pour but d'empêcher l'espionnage et la tricherie.

1.10.1- Protocoles à apport nul de connaissance

1.10.2- Le concept de tiers de confiance

1.11- Echange de clef

Les deux méthodes les plus courantes d'établissement de clef sont le *transport* de clef et la *génération* de clef. Un exemple de transport de clef est l'utilisation d'un algorithme à clef publique pour chiffrer une clef de session générée aléatoirement. Un exemple de génération de clef est le protocole *Diffie–Hellman* [Dif**]**, qui permet de générer un secret partagé à partir d'informations publiques.

Le seul protocole décrit ici est le principe de génération de clef de *Diffie–Hellman* et les façons de l'étendre pour contrer les attaques de l'intercepteur. *Diffie–Hellman* est à la base de nombreux protocoles d'échange de clef.

1.11.1- Diffie–Hellman

1.11.2- Échange de clef et authentification mutuelle

1.11.3- Propriétés des protocoles d'échange de clefs

1.11.4- Hiérarchie des clefs

On peut définir plusieurs types de clefs suivant leurs rôles :

- Clefs de chiffrement de clefs
- Clefs maîtresses
- Clefs de session ou de chiffrement de données

1.12- Cryptanalyse

Si le but de la cryptographie est d'élaborer des méthodes de protection, le but de la cryptanalyse est au contraire de casser ces protections. Une tentative de cryptanalyse d'un système est appelée une attaque, et elle peut conduire à différents résultats :

- *Cassage complet* : le cryptanalyste retrouve la clef de déchiffrement.
- *Obtention globale* : le cryptanalyste trouve un algorithme équivalent à l'algorithme de déchiffrement, mais qui ne nécessite pas la connaissance de la clef de déchiffrement.
- *Obtention locale* : le cryptanalyste retrouve le texte en clair correspondant à un message chiffré.
- *Obtention d'information* : le cryptanalyste obtient quelques indications sur le texte en clair ou la clef (certains bits de la clef, un renseignement sur la forme du texte en clair,...)

1.12.1- Attaques des fonctions de chiffrement

1.12.1.1- Classement des attaques en fonction des données dont dispose le cryptanalyste

On distingue plusieurs types d'attaques suivant les informations que peut obtenir le cryptanalyste. Ce sont :

- L'attaque à texte chiffré seulement
- L'attaque à texte en clair connu
- L'attaque à texte en clair choisi
- L'attaque à texte chiffré choisi

1.12.1.2- Attaques sur les algorithmes symétriques

- *Attaques au niveau des clefs*
- *Cryptanalyse différentielle*

- *Cryptanalyse linéaire*

1.12.2.3- Attaques sur les algorithmes asymétriques

- *Attaques au niveau des clefs*
- *Attaque à texte en clair deviné et problème de la faible entropie*
- *Attaque à texte chiffré choisi*
- *Attaque temporelle*
- *Attaques des générateurs pseudo-aléatoires*
- *Attaques des fonctions de hachage à sens unique*
- *Attaque des anniversaires*

1.12.2.4- Attaques des protocoles cryptographiques

On distingue deux types d'attaques sur les protocoles : les *attaques passives* et les *attaques actives*. Dans le premier cas, l'attaquant ne peut qu'espionner les données échangées par les tiers communicants, alors que dans le second cas il peut modifier ou supprimer des messages, en ajouter des nouveaux ou des anciens qu'il rejoue.

- *L'attaque par mascarade*
- *La « replay attack »*
- *Attaque par réflexion*
- *Attaque dite de l'intercepteur*

Dans le Chapitre 2, on a étudié les fondements mathématiques de l'AES

Dans cette partie, on va donner quelques théorèmes et propositions de la théorie des corps fini ont servi l'algorithme AES pour envoyer un message qui nous ont servi à appliquer AES.

2.1.1 - Principales propriétés

Comme **Z** est un anneau euclidien, a fortiori principal, tout idéal ***I*** s'écrit de manière unique sous la forme ***I*** = ***n***. ***Z*** où ***n*** est un entier appelé le générateur positif de ***I***. Par définition, c'est le plus petit entier positif non nul appartenant à ***I***. Les seules structures quotients obtenues à partir de ***Z*** sont donc les anneaux commutatifs unitaires **Z**/***n***.**Z**. Ces anneaux donnent les premiers exemples de corps finis. On a en effet le résultat suivant :

Théorème 2.1.1.1

L'anneau **Z/nZ** *est un corps si, et seulement si,* ***n*** *est un nombre premier.*

Théorème 2.1.1.2

Le cardinal d'un corps fini est une puissance d'un nombre premier qui en est sa caractéristique. Plus exactement, tout corps fini est une extension finie d'un corps premier $\boldsymbol{F_p}$.

$X^{p^n} - X$

Les polynômes irréductibles de ***K*** *de degré* ***n*** *sont les diviseurs premiers de degré* ***n*** *du polynôme suivant et sont donc tous cyclotomiques :*

Proposition 2.1.1.4

Le groupe multiplicatif d'un corps fini : $\boldsymbol{F_q = Z/qZ^*}$ ($\boldsymbol{q=p^n}$, ***p*** premier) *est cyclique d'ordre* ***(q-1)***

Proposition 2.1.1.5

K[***X***]/***P***[***X***] est un corps si et seulement si ***P*** est irréductible dans ***K***[***X***]

2.2 Les Mathématiques de l'AES

p^m

2.2.1- Rappels: construction d'un corps fini à éléments

On considère l'anneau $Z_2[x]$ des polynômes à une indéterminée à coefficients dans $Z_2 = Z/2Z$.
On rappelle que $Z_2 \sim \{0, 1\}$ est muni des deux opérations suivantes :

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

$$\frac{Z_2[x]}{m(x)\ Z_2[x]} \quad x^4 + x^3 + x + 1$$

Le polynôme est irréductible dans $Z_2[x]$, car $m(0)=1$ et $m(1)=1$, et donc l'ensemble des classes résiduelles modulo $m(x)$ des éléments de $Z_2[x]$, que l'on note : , est un corps: c'est le *corps de Galois* $GF(2^8)$.

≡

On rappelle que si $A(x)$ est un polynôme de $Z_2[x]$, il existe deux polynômes $B(x)$ $Z_2[x]$ et $R(x)$ $Z_2[x]$, *uniques*, tels que : $A(x) = B(x)\ m(x) + R(x)$, avec $d°R < d°m$, ce qui équivaut à dire que : $A(x)$ $R(x)$ **mod.** $m(x)$ *:* $R(x)$ est la classe résiduelle modulo $m(x)$ de $A(x)$

$$\frac{Z_2[x]}{m(x)\ Z_2[x]} \cong$$

Comme $m(x)$ est de degré 8, $GF(2^8)$ est donc constitué des polynômes de $Z_2[x]$ de degré au plus égal à 7, et à coefficients dans $Z/2Z=F_2$:

$$GF(2^8) \cong \left\{ \sum_{i \geq 0} a_i X^i \ \bmod m(X),\ a_i \in \{0,1\} \right\} \cong \left\{ \sum_{i=0}^{i=7} a_i X^i,\ a_i \in \{0,1\} \right\}$$

$$a_7 X^7 + \ldots + a_1 X + a_0$$

En pratique, on représentera le polynôme $A(X) =$ par l'octet

$$a_7 a_6 \ldots a_1 a_0$$

A={} : on dira qu'on utilise une notation polynomiale des éléments de $GF(2^8)$.

En fait, l'***AES*** fonctionne essentiellement avec le système ***Hexadécimal***: ***0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.***

2.2.1.1-Les opérations de $GF(2^8)$

$Z/2Z$

L'addition de deux éléments de F_2 = correspond à l'opération *XOR (*. Ainsi, l'addition de deux éléments ***A(X)*** et ***B(X)*** de $GF(2^8)$ correspond à l'addition de deux polynômes à coefficients dans

$$A(x) = a_7x^7 + a_6x^6 + \ldots + a_1x + a_0$$
$$B(x) = b_7X^7 + b_6X^6 + \ldots + b_1X + b_0$$
$$\Rightarrow \quad A(x) \oplus B(x) = (a_7 \oplus b_7)x^7 + (a_6 \oplus b_6)x^6 + \ldots + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

Multiplication dans $GF(2^8)$

$\otimes$

Soient ***A(x)*** et ***B(x)*** 2 polynômes de $GF(2^8)$: on note l'opération de multiplication du *corps de* ***Galois***

$\in$

On va expliciter le produit $A(x)\ B(x)\ \ GF(2^8)$; deux cas sont à considérer :

$\otimes$ - **Si** $deg\ (A)+deg(B)$7 : alors $A(x)\ B(x) = A(x)$ x $B(x)$, x étant la multiplication dans $Z_2[x]$

$\otimes$ - **Si** $deg(A)+deg(B)\ \ 8$: alors $A(x)\ B(x) = R(x)$, où $R(x)$ est la classe résiduelle du produit $A(x)$ x $B(x)$ (dans $Z_2\ [x]$) modulo $m(x)$

Remarque : En notation hexadécimale, le polynôme ***m(X)*** s'écrit **0*x*11B**.

Calcul de l'inverse d'un élément de $GF(2^8)$

$$X(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \Leftrightarrow (a_7a_6a_5a_4a_3a_2a_1a_0)_2, a_i = 0 \text{ ou } 1$$

$\otimes$

Il s'agit de calculer $X^{-1}(x)$ au sens de la loi du groupe multiplicatif du corps $GF(2^8)$.

$$X(x) \times U(x) + m(X) \times V(x) = 1$$

On utilise l'***algorithme étendu d'Euclide***, en calculant ***U(x)*** et ***V(x)*** 2 polynômes de $GF(2^8)$ tels que:

(E)

Les polynômes ***U(x)*** et ***V(x)*** existent toujours, car ***m(x)*** est premier à tout polynôme de $GF(2^8)$

$\otimes\ 1 \bmod m(x)$

Alors: $X(x)U(x)$) ce qui veut dire que: $X(x)U(x) = 1$, *i.e.* $X^{-1}(x) = U(x)$ dans $GF(2^8)$.

Algorithme pour la résolution de l'équation (*E*)

*Si **A** et **B** sont 2 polynômes premiers entre eux i.e. PGCD(**A**,**B**)=1, et si $q_1, q_2 \ldots q_t$ est la suite des quotients obtenus en employant l'algorithme d'**Euclide** :*

$A = B\,q_k + r_k$, $0 \leq \deg r_k < \deg B$, *on construit la matrice:*

		q_1	q_2	q_3	……	q_{t-1}	q_t
0	1	P_1	P_2	P_3	……	P_{t-1}	A
1	0	Q_1	Q_2	Q_3	……	Q_{t-1}	B

Avec : $P_1 = q_1$, $Q_1 = 1$; $P_2 = q_2 P_1 + 1$, $Q_2 = q_2 Q_1$; *et pour* $k \geq 3$:

$P_k = q_k P_{k-1} + P_{k-2}$ *et* $Q_k = q_k Q_{k-1} + Q_{k-2}$; *alors :*

$A\,Q_{t-1} - B\,P_{t-1} = (-1)^t$, *d'où la solution de l'équation (E) :*

$$\begin{cases} U = (-1)^t Q_{t-1} \\ V = (-1)^{t+1} P_{t-1} \end{cases}$$

□

2-2-1-2- Méthodes pour multiplier deux éléments de $GF(2^8)$

En pratique, pour réaliser cette multiplication, on dispose de deux méthodes.

2-2-1-2-1- Méthode "lente" : multiplication par x^i .

2-2-1-2-2-Méthode "rapide" à partir d'une représentation exponentielle.

2-2-2-Les polynômes à coefficients dans $GF(2^8)$

Tout comme un octet peut être représenté par un polynôme de degré 7, un mot de 32 bits (4 octets) peut l'être par un polynôme de degré 3 à coefficients dans $GF(2^8)$, chaque coefficient représentant un octet du mot.

L'addition de deux mots est alors égale à l'addition des polynômes les représentants, soit un ou-exclusif entre les coefficients de même degré.

$$a(X) + b(X) = (a_3 \oplus b_3)X^3 + (a_2 \oplus b_2)X^2 + (a_1 \oplus b_1)X + (a_0 \oplus b_0)$$

La multiplication de deux mots donne un polynôme de degré 3+3 dont on peut calculer les coefficients par la définition précédente et la définition générale du produit de polynômes.

$$a(X) \times b(X) = c(X) = c_6X^6 + c_5X^5 + c_4X^4 + c_3X^3 + c_2X^2 + c_1X + c_0$$

$$
\begin{aligned}
c_0 &= a_0 \otimes b_0 \\
c_1 &= a_1 \otimes b_0 \oplus a_0 \otimes b_1 \\
c_2 &= a_2 \otimes b_0 \oplus a_1 \otimes b_1 \oplus a_0 \otimes b_2 \\
c_3 &= a_3 \otimes b_0 \oplus a_2 \otimes b_1 \oplus a_1 \otimes b_2 \oplus a_0 \otimes b_3 \\
c_4 &= a_3 \otimes b_1 \oplus a_2 \otimes b_2 \oplus a_1 \otimes b_3 \\
c_5 &= a_3 \otimes b_2 \oplus a_2 \otimes b_3 \\
c_6 &= a_3 \otimes b_3
\end{aligned}
$$

$$A = \frac{GF(2^8)[X]}{(X^4+1)GF(2^8)[X]}$$

Afin de rester dans l'espace des mots de 32 bits, on considère les polynômes **c(X)** $modulo$ un polynôme de degré ***4*** qui vaut pour l'**AES**: $M(X) = X^4+1$, formant un groupe que l'on pourrait noter :

$$d(X) = d_3X^3 + d_2X^2 + d_1X + d_0 = c(X) \bmod (X^4+1) = a(X) \otimes b(X)$$

Notons alors : , le produit modulaire dans $\mathcal{A}$ des polynômes $a(X)$ et de $b(X)$ à coefficients dans $GF(2^8)$.

On sait par ailleurs que : $X^j \bmod (X^4+1) = X^{j \bmod 4}$, et donc :

$$c(X) \bmod (X^4+1) = [c_6X^6 + c_5X^6 + c_4X^4 + c_3X^3 + c_2X^2 + c_1X + c_0] \bmod (X^4+1)$$

ou :

$$c(X) \bmod (X^4+1) = \sum_{j=0}^{j=6} c_j X^j \bmod (X^4+1) = \sum_{j=0}^{j=6} c_j X^{j \bmod 4}$$

$d(X) =$

$$c_6X^2 + c_5X + c_4 + c_3X^3 + c_2X^2 + c_1X + c_0$$

$d(X) =$

$$c_3X^3 + (c_2 + c_6)X^2 + (c_1 + c_5)X + (c_0 + c_4)$$

$d(X) =$

$$d_3X^3 + d_2X^2 + d_1X + d_0$$

= , et par simple identification et compte tenu des formules précédentes donnant les c_j , on obtient finalement :

$$
\begin{aligned}
d_0 &= a_0 \otimes b_0 \oplus a_3 \otimes b_1 \oplus a_2 \otimes b_2 \oplus a_1 \otimes b_3 \\
d_1 &= a_1 \otimes b_0 \oplus a_0 \otimes b_1 \oplus a_3 \otimes b_2 \oplus a_2 \otimes b_3 \\
d_2 &= a_2 \otimes b_0 \oplus a_1 \otimes b_1 \oplus a_0 \otimes b_2 \oplus a_3 \otimes b_3 \\
d_3 &= a_3 \otimes b_0 \oplus a_2 \otimes b_1 \oplus a_1 \otimes b_2 \oplus a_0 \otimes b_3
\end{aligned}
$$

Ces opérations peuvent être mises sous forme matricielle.

$$
\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}
$$

Comme X^4+1 n'est pas un polynôme irréductible, $\mathcal{A}$ n'est pas un corps et ses éléments ne sont pas forcément inversibles. L'***AES*** utilise alors pour ses calculs sur les mots le polynôme ***a(X)*** inversible dans $\mathcal{A}$.

$$
\begin{aligned}
a(X) &= (0\text{x}03)X^3 + (0\text{x}01)X^2 + (0\text{x}02)X + (0\text{x}02) \\
a^{-1}(X) &= (0\text{x}0\text{b})X^3 + (0\text{x}0\text{d})X^2 + (0\text{x}09)X + (0\text{x}0\text{e})
\end{aligned}
$$

Multiplication par x

$$b(x) = b_3x^4 + b_2x^3 + b_1x^2 + b_0x$$

Soit ,

$$x \otimes b(x)$$

Alors : est obtenu en réduisant ***xb(x)*** modulo (x^4+1), ce qui donne :

$$b_2x^3 + b_1x^2 + b_0x + b_3$$

c(x)==

que l'on peut écrire sous forme matricielle :

$$
\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \begin{bmatrix} 00 & 00 & 01 \\ 00 & 00 & 00 \\ 01 & 00 & 00 \\ 00 & 01 & 00 \end{bmatrix}
$$

=

Cette procédure simplifie notablement l'opération de multiplication par *x*.

Dans le Chapitre 3, on a étudié la description et fonctionnement de l'algorithme AES

3.1- Entrées et Sorties

Les entrées et les sorties de l'***AES*** consistent en des séquences de 128 bits. La clé secrète de chiffrement est une suite de 128, 192 ou 256 bits. Un bloc peut être représenté comme une matrice d'octets 4x**Nb** (*State*). Les octets lus en entrée y sont copiés colonne après colonne (chaque colonne représente un mot lu), selon la figure n° :

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$\begin{Bmatrix} a_{0,3} \\ a_{1,3} \\ a_{2,3} \\ a_{3,3} \end{Bmatrix}$

$w_0 =$; $w_1 =$; $w_2 =$; $w_3 =$

Figure n° : Structure d'un bloc d'octets

3.2- Notation, structure de données

Les conventions de notation issues de la spécification du *NIST* sont les suivantes :

AddRoundKey()	:	transformation (pour le chiffrement et le déchiffrement) qui ajoute une clé de tour (*Round key*) au bloc (*State*) courant.
MixColumns()	:	transformation qui permute les colonnes d'un bloc lors du chiffrement.
RotWord()	:	fonction utilisée par la routine d'expansion de la clé qui applique à un mot de 4 octets une permutation circulaire.
ShiftRows()	:	transformation qui applique des permutations circulaires aux trois dernières lignes du bloc lors du chiffrement.
SubBytes()	:	transformation qui opère une substitution non linéaire de chaque octet du bloc en utilisant une table (***S-box***) lors du chiffrement.
SubWord()	:	fonction utilisée par la routine d'expansion de la clé qui prend un mot en entrée et applique à ses 4 octets une substitution non linéaire (***S-box***)
InvMixColumns()	:	transformation du déchiffrement qui est l'inverse de la transformation **MixColumns()**.
InvShiftRows()	:	transformation du déchiffrement qui est l'inverse de la transformation **ShiftRows()**.
InvSubBytes()	:	transformation du déchiffrement qui est l'inverse de la transformation **SubBytes()**.
K	:	la clé de chiffrement.
Nb	:	Nombre de colonnes (mots de 32 bits) du bloc. Pour l'***AES*** : **Nb=4**.
Nk	:	nombre de mots de 32 bits dans la clé de chiffrement. Pour ***AES:*** **Nk= 4,6 ou 8**.
Nr	:	Nombre de tours, fonction de **Nb** et **Nk**. Pour l'***AES*** **Nr=10,12 ou 14**.

Rcon[] : table constante.

3.3- Description de l'algorithme de chiffrement *A.E.S*

L'algorithme ***AES*** n'est pas exactement celui de *Rijndael* dans la mesure où ce dernier supporte des tailles de blocs plus nombreux qu'***AES***. L'*Advanced Encryption Standard* **fixe la taille de blocs à 128 bits**: **Nb=4** reflète le nombre de mots de 32 bits dans un tel bloc (c'est aussi le nombre de colonnes nécessaire pour une représentation matricielle).

Détails de la diversification de la clef dans l'*AES*

Cette étape, noté *KeyExpansion*, permet de diversifier la clé de chiffrement ***K*** (de 4 ***Nk*** octets) dans une clé étendue ***W*** de 4***Nb*** (***Nr*** +1) octets. On disposera ainsi de ***Nr*** +1 clés de rondes (chacune de 4***Nb*** octets).

Les matrices ***K*** et ***W*** peuvent être vues comme une succession de colonnes, chacune constituée de 4 octets. Dans la suite, on notera c[i] (resp. k[i]) la $(i+1)^{ème}$ colonne de ***W*** (resp. de ***K***).

L'algorithme utilisé pour la diversification de clé diffère légèrement selon que $N_k \leq 6$ ou $N_k > 6$.

Dans tous les cas, les N_k premières colonnes de ***K*** sont recopiées sans modifications aux N_k premières colonnes de ***W***. Les colonnes suivantes sont définies récursivement à partir des colonnes précédentes. *KeyExpansion* y utilise notamment les éléments suivants :

- *SubWord* qui est une fonction prenant en entrée un mot de 4 octets et applique la boîte-S ***S-box*** sur chacun des octets.
- La fonction *RotWord* qui prend en entrée un mot de 4 octets $a = [a_0, a_1, a_2, a_3]$ et effectue une permutation circulaire de façon à renvoyer le mot $[a_1, a_2, a_3, a_0]$.
- Le tableau de constantes de rondes *Rcon[i],* indépendant de N_k, qui est défini récursivement par :

$$\mathrm{Rcon}[i] = [x^{i-1}, 00, 00, 00], \forall i \geq 1$$

On pourra utiliser la fonction ***x^i-time*** pour calculer la valeur de *Rcon[i]*.

Lars Knudsen **et *John Mathiassen* [Knu] ont montré en 2004 que le « *key schedule* » joue un rôle primordial dans la résistance aux cryptanalyses de type linéaire ou différentielle. Les générations de sous-clés, lorsqu'elles sont bien conçues, permettent d'arriver rapidement à l'absence de biais statistiques potentiellement exploitables par la cryptanalyse.**

La clé étendue

La clé étendue est un vecteur ***W*** composé de mots de 4 octets ($W[i] = (a, b, c, d)$), de longueur $N_b (N_r + 1)$:

- Les N_k premiers mots ($W[1], \ldots W[N_k]$) contiennent la clé.
- Les mots suivants sont calculés en faisant un *XOR* du mot précédent ($W[i-1]$) et du mot situé N_k positions avant ($W[i - N_k]$).
- Pour les mots situés sur une position qui est un multiple de N_k, une transformation est appliquée à $W[i-1]$ avant le *XOR*. Il s'agit d'une permutation cyclique d'un cran vers la gauche $(a, b, c, d) \rightarrow (b, c, d, a)$, suivie d'une application de la ***S-Box*** séparément sur chaque octet du mot et d'un *XOR* avec un certain vecteur dépendant du tour.
- Si la clé est de 192 bits, et que $(i-4)$ est un multiple de 4, on applique encore la ***S-Box*** séparément sur chaque octet du mot $W[i-1]$ déjà modifié, avant de faire le *XOR* final.

Au fur et à mesure des tours, on prend les sous-clés nécessaires les unes à la suite des autres (pour le tour i, on prend les mots de $W[i \,.\, N_b]$ à $W[(i+1) \,.\, N_b]$)

Déchiffrement dans AES

La routine de chiffrement peut être inversée et réordonnée pour produire un algorithme de déchiffrement utilisant les transformations *InvSubBytes*, *InvShiftRows*, *InvMixColumns*, et *AddRoundKey*.

Dans cette version du déchiffrement, la séquence des transformations diffère de celle du chiffrement, le traitement de la clé restant inchangé.

Attaques

L'***AES*** n'a pour l'instant pas été cassé et la recherche exhaustive (« *force brute* ») demeure la seule solution. ***Rijndael*** a été conçu de telle manière à rendre des méthodes classiques comme la cryptanalyse linéaire ou différentielle impraticables.

Attaques sur des versions simplifiées

Attaques sur la version complète

L'attaque XSL

Cette méthode a causé une controverse quant à sa réelle efficacité : les auteurs prétendaient que *XSL* pouvait potentiellement casser ***Rijndael*** . Si la communauté des cryptologues est restée sceptique face à cette attaque, elle a eu le mérite de relancer les doutes au sujet de la simplicité algébrique de ***AES***. *Don Coppersmith* affirma au sujet de cette attaque :

Je crois que le travail de Courtois-Pierprzyk [Cou] *a un défaut. Les auteurs surestiment le nombre d'équations linéairement indépendantes. En conséquence, ils n'ont pas assez d'équations linéaires à disposition pour résoudre le système et la méthode ne casse pas* ***Rijndael.*** *La méthode a un certain mérite et il est intéressant de l'examiner plus en avant mais elle ne casse pas* ***Rijndael*** *en l'état actuel*

BIBLIOGRAPHIE

[Bih] **Biham E. & Shamir A.** Differential cryptanalysis of the full16-round DES *Advances of Cryptology, proceedings of CRYPTO 91*. [4] Preprint, Dec.1991

[Cho] **Chor B. & Rivest R.L.** A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Trans Inform Theory* 34 (1998): 901-909

[Cou] **Courtois N.** & **Pieprzyk J.** (2002). "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations". *LNCS* **2501**: 267–287.

[Dae] **Daemen J. & V. Rijmen V.**. *The Design of Rijndael, AES- The Advanced Encryption Standard;* Springer-Verlag, 2002.

[Den] **Denning D**. & **Sacco G**. Timestampes in key distributed protocols. *Communication of the ACM,* 24(8): 533-535, 1981

[Dif] **Diffie W. & Hellman M.E.**, New Directions in Cryptography, *IEEE Transactions on Information Theory*, Vol. IT-22, Nov. 1976, pp: 644–654.

[Fei] **Feige U., Fiat A. & Shamir A.** Zero-knowledge proofs of identity. *Journal of Cryptology,* 1:77-94, 1988.

[Fer] **Fergusson N., Kelsey J., Lucks S., Schneier B., Stay M., Wagner D. &Whiting D.** The best known attacks against AES/Rijndael. In *Fast Software Encryption*, *Proceedings FSE 2000,* pp. 213–230, Springer Verlag, 2000.

[Kar] **Karn P**. & **Simpson W**., Photuris: Session-Key, Management Protocol, March 1999.

[Ker] **Kerckhoffs A.** La cryptographie militaire. *Journal des sciences militaires.* Vol. IX, pp. 5-83 (Janvier 1883) et pp.161-191 (Février 1883)

[Knu] **Knudsen Lars R.** &, **Mathiassen John E**.: On the Role of Key Schedules in Attacks on Iterated Ciphers. *ESORICS 2004:* 322-334

[Lan] **Langford S.K.** & **Hellman M.E.** , "Linear-Differential Cryptanalysis," in *Advances in Cryptology: Proceedings of Crypto 94*, Yvo G. Desmedt, Editor, Springer-Verlag, Berlin, Lecture Notes in Computer Science #839, pp. 17-25, 1994.

[**Mat**] **Matsui, M.** & **A. Yamagishi**. 1994. A New Cryptanalytic Method for FEAL Cipher.

IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science. E77-A(1): 2-7.

[Mer] **Merkle R.** & **Hellman M**., Hiding Information and Signatures in Trapdoor Knapsacks, *IEEE Trans. Information Theory*, 24(5), September 1978, pp.525–530.

[Mur] **Murphy S.** The cryptanalysis of FEAL 4 with twenty chosen plaintexts. EUROCRYPT 1992: 81–91; J. Cryptology 2(3): 145–154 (1990).

[Nec] **Nechvatal *et al.*** *Report on the development of the Advanced Encryption Standard* (*AES*). Computer Security Division / Information Technology Laboratory National Institute of Standards and Technology (NIST); U.S. Department of Commerce October 2, 2000

[Oor] **Diffie W., van Oorschot P.C., Wiener M.J**. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, vol.2 (1992), pp.107-125.

[RSA Lab] RSA Laboratories, What is cipher block chaining mode? *Home: Standards Initiatives: Public-Key Cryptography Standards (PKCS): PKCS #6: Extended-Certificate Syntax Standard: 2.1 Cryptographic Tools: 2.1.4 What is a block cipher?*

[Sha] **Shannon C.E.,** "Communication Theory of Secrecy Systems," *Bell System Technical Journal,* Vol. 28, No. 4 (October 1949), pp. 656-715.

[Vau] **Vaudenay S.,** Cryptanalysis of of the Chor-Rivest cryptosystem. *Advances in Cryptology - CRYPTO '98 / Lecture Notes in Computer Science*. Volume 1462/1998, pp. 243-256

[Ver] **Vernam Gilbert Sandford**, "Cipher Printing Telegraph Systems For Secret Wire and Radio Telegraphic Communications", Journal of the IEEE, Vol 55, pp109-115 (1926)

[Xue] **Xuejia L.** & **Massey J.L.** A proposal for a new block encryption standard. *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology.* Aarhus-Denmark, 1991: 389-404, Springer-Verlag New York, Inc., New York, NY, USA