Chapitre 3
Chiffrement symétrique et asymétrique

1. Les algorithmes de chiffrement symétrique

Les algorithmes de chiffrement symétrique sont une pierre angulaire de la cryptographie moderne, offrant une méthode efficace et rapide pour sécuriser les communications. Leur caractéristique principale est l'utilisation d'**une seule et même clé** pour les opérations de **chiffrement** (transformation du texte en clair en texte chiffré) et de **déchiffrement** (transformation du texte chiffré en texte en clair).

Autrement dit, l'expéditeur (Alice) et le destinataire (Bob) doivent **connaître et partager préalablement la même clé secrète**. Toute personne possédant cette clé peut à la fois chiffrer et déchiffrer les messages.

Principe de fonctionnement

– **Chiffrement** : l'expéditeur (Alice) prend le message original (texte en clair), applique l'algorithme de chiffrement et utilise la clé secrète partagée pour transformer le message en un format illisible (texte chiffré).

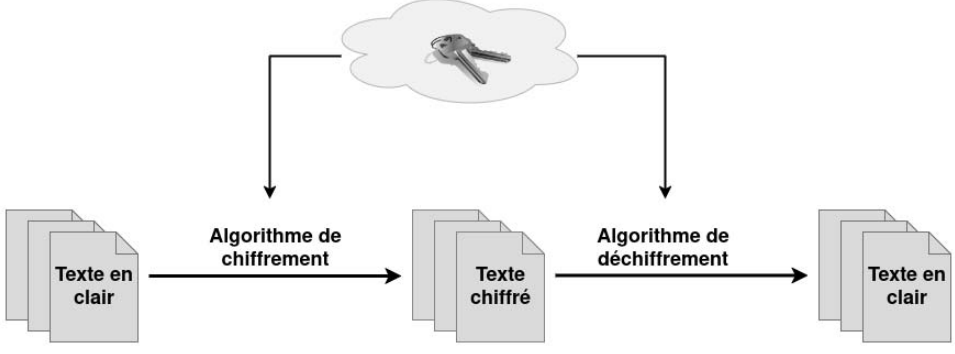
  $C = E_k(P)$ où :

  – C est le texte chiffré (ou *ciphertext* en anglais) ;
  – E est l'algorithme de chiffrement ;
  – K est la clé secrète partagée ;
  – P est le texte en clair (ou *plaintext* en anglais).

– **Déchiffrement** : le destinataire (Bob) reçoit le texte chiffré C, applique l'algorithme de déchiffrement et utilise la même clé secrète K pour retrouver le message original.

  $P = D_K(C)$ où D est l'algorithme de déchiffrement.

La sécurité repose entièrement sur le secret de cette clé secrète K. Si un attaquant parvient à obtenir la clé secrète, il peut facilement déchiffrer tous les messages chiffrés avec cette clé secrète.



Propriétés fondamentales

Les bons algorithmes symétriques vérifient quelques propriétés importantes :

- **Résistance à la cryptanalyse** : il doit être pratiquement impossible de retrouver le texte en clair ou la clé sans connaître cette dernière.
- **Diffusion** : une modification minime du texte en clair ou de la clé secrète entraîne un changement radical du texte chiffré.
- **Performance** : ils sont généralement **plus rapides** que les algorithmes asymétriques, en particulier pour chiffrer de grandes quantités de données.
- **Gestion des clés** : c'est le principal défi. La clé doit être partagée de manière sécurisée entre l'expéditeur (Alice) et le destinataire (Bob) avant toute communication chiffrée.

Types d'algorithmes de chiffrement symétrique

On distingue principalement deux grandes catégories :

- **Chiffrement par bloc** (*block ciphers*) : ces algorithmes divisent le texte en clair en blocs de taille fixe (par exemple : 64 bits, 128 bits) et chiffrent chaque bloc indépendamment.
  - **Fonctionnement** : chaque bloc de données est traité comme une unité et transformé en un bloc de texte chiffré de même taille. La transformation est répétée plusieurs fois (*rounds*) en utilisant des sous-clés dérivées de la clé principale.
  - **Modes d'opération** : pour chiffrer des messages plus longs que la taille d'un seul bloc, ou pour ajouter de la robustesse, différents modes d'opération sont utilisés (ex. : CBC, CTR, GCM). Ces modes définissent comment les blocs sont liés entre eux et comment l'initialisation vectorielle (IV) est utilisée.
  - **Exemples** : AES, DES, Blowfish, Twofish, etc.

– **Chiffrements par flux (*stream ciphers*)** : ces algorithmes chiffrent les données bit par bit ou octet par octet.
  – **Fonctionnement** : un générateur de flux de clés (*key stream generator*) produit une séquence pseudo-aléatoire de bits (le flux de clés). Ce flux de clés est ensuite combiné (généralement par une opération XOR) avec le texte en clair bit par bit ou octet par octet pour produire le texte chiffré.

    $C = P \oplus S$ où S est le flux de clés et $\oplus$ est l'opération XOR.
  – **Avantages** : très rapides et adaptés aux applications où les données arrivent en continu (ex. : communication vocale ou vidéo en temps réel).
  – **Inconvénients** : si le même flux de clés est réutilisé avec deux messages différents, cela compromet la sécurité des deux messages. Un vecteur d'initialisation (IV) est crucial pour éviter la réutilisation du flux de clés.
  – **Exemples** : RC4, ChaCha20, etc.

Avantages des algorithmes de chiffrement symétrique

– **Vitesse** : extrêmement rapides, ce qui les rend idéaux pour chiffrer de grandes quantités de données (trafic réseau, bases de données, disques durs entiers).
– **Faible consommation de ressources** : ils sont moins gourmands en CPU et en mémoire par rapport aux algorithmes asymétriques.
– **Simplicité de mise en œuvre (pour les opérations de chiffrement/déchiffrement)** : une fois la clé partagée, le processus est direct.

Inconvénients des algorithmes de chiffrement symétrique

– **Problème de la distribution des clés** : c'est le défi majeur. Comment deux parties (Alice et Bob) peuvent-elles établir et partager en toute sécurité une clé secrète sur un canal non sécurisé ? Si la clé est interceptée, toute la sécurité est compromise. Ce problème est souvent résolu en utilisant des algorithmes de chiffrement asymétrique pour établir initialement une clé symétrique (protocole d'échange de clés Diffie-Hellman par exemple ou ML-KEM) ou via des canaux sécurisés hors bande.
– **Authentification et non-répudiation** : les algorithmes symétriques ne fournissent intrinsèquement pas d'authentification de l'expéditeur ni de non-répudiation (la preuve qu'une personne spécifique a envoyé un message et ne peut le nier). Pour cela, des techniques supplémentaires comme les codes d'authentification de message (MAC) sont nécessaires.

Les algorithmes de chiffrement symétrique sont essentiels pour la performance et l'efficacité de la cryptographie moderne. Leur rapidité les rend indispensables pour la protection de volumes importants de données. Maintenant, voyons comment AES et DES fonctionnent.

1.1 Fonctionnement de l'AES

L'**AES** (*Advanced Encryption Standard*) est aujourd'hui **l'algorithme de chiffrement symétrique le plus utilisé dans le monde**. Adopté comme standard par le NIST en 2001, il est devenu incontournable dans les domaines de la sécurité des communications, du stockage sécurisé, des VPN, du chiffrement de fichiers, et bien d'autres.

AES a été conçu pour remplacer le vieillissant **DES**, dont la taille de clé (56 bits) n'était plus suffisante face à la puissance croissante des ordinateurs.

Le 3DES, qui appliquait DES trois fois, a prolongé sa durée de vie mais était plus lent et moins élégant.

En 1997, le NIST a lancé un appel à propositions pour un nouvel algorithme de chiffrement symétrique, le futur AES, qui devait répondre à plusieurs critères :

- **Sécurité** : l'algorithme devait être extrêmement résistant aux attaques connues.
- **Performance** : il devait être rapide et efficace, aussi bien en logiciel qu'en matériel, sur une large gamme de plateformes.
- **Flexibilité** : il devait supporter différentes tailles de clés et de blocs.
- **Coût** : il devait être libre de droits et disponible pour tous.

Parmi les quinze propositions initiales, cinq finalistes furent retenus : **MARS**, **RC6**, **Serpent**, **Twofish** et **Rijndael**. Après un examen public et des analyses approfondies par la communauté cryptographique mondiale, **Rijndael** fut finalement choisi en 2001 et normalisé sous le nom d'**AES (FIPS PUB 197)**.

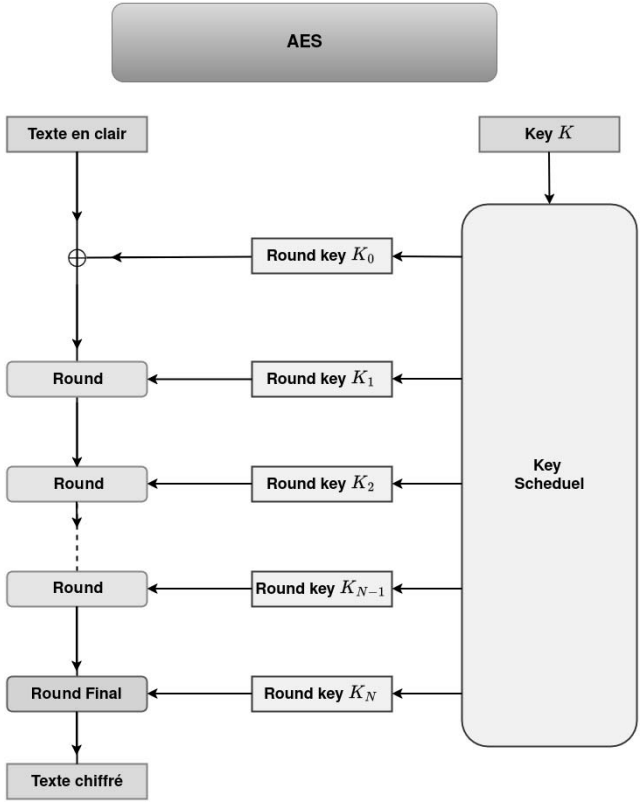
1.1.1 Principes et caractéristiques

Le principe fondamental de l'AES repose sur une série de transformations mathématiques et logiques appliquées de manière répétée à un bloc de données.

AES est un **chiffrement par bloc** basé sur un réseau de substitution-permutation (SPN, *Substitution-Permutation Network* en anglais). Cela signifie qu'il opère sur des blocs de données de taille fixe et utilise une série de substitutions (remplacement d'éléments par d'autres) et de permutations (réarrangement d'éléments) pour transformer le texte clair en texte chiffré.

Ses principales caractéristiques sont :

– **Taille de bloc fixe** : AES opère sur des blocs de données de **128 bits** (16 octets).
– **Tailles de clé variables** : il supporte trois tailles de clé :
  – **AES-128** : clé de 128 bits pour 10 tours (rounds)
  – **AES-192** : clé de 192 bits pour 12 tours (rounds)
  – **AES-256** : clé de 256 bits pour 14 tours (rounds)
  – Chaque tour applique une série de transformations pour augmenter la diffusion et la confusion.
– **Structure itérative** : le chiffrement et le déchiffrement sont basés sur une répétition de rounds identiques, à l'exception du dernier round.



1.1.2 AES et GF(2^8)

Les blocs de 128 bits (16 octets) de données, ainsi que la clé, sont manipulés comme des tableaux 2D de 4x4 octets, appelés des **états** (*states* en anglais).

Par exemple, un bloc de 16 octets $a_0, a_1, a_2, a_3, \ldots, a_{15}$ est arrangé dans l'état comme suit :

$$\begin{matrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{matrix}$$

Pour effectuer des opérations bien définies, inversibles et sécurisées sur ces octets, AES utilise des opérations **algébriques dans le corps fini** :

$$GF(2^8) = F_{256} = F_2[X]/(X^8 + x^4 + X^3 + X + 1)$$

c'est-à-dire un ensemble de 256 éléments, muni de **règles bien définies pour l'addition et la multiplication**. GF signifie *Galois Field* (corps de Galois).

Représentation des octets dans GF(2^8)

Chaque octet est vu comme un polynôme binaire de degré ≤ 7. De manière générale, un octet (8 bits) $(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$ est représenté par le polynôme suivant :

$$p(X) = b_7X^7 + b_6X^6 + b_5X^5 + b_4X^4 + b_3X^3 + b_2X^2 + b_1X + b0$$

Par exemple si $a_0 = 0x57 = (01010111)_2$ alors le polynôme correspondant est :

$$p(X) = 0X^7 + X^6 + 0X^5 + X^4 + 0X^3 + X^2 + X + 1$$

Ainsi :

$$p(X) = X^6 + X^4 + X^2 + X + 1$$

Addition dans GF(2^8)

L'addition dans $GF(2^8)$ est simplement un XOR bit à bit, ou simplement une addition de deux polynômes à coefficients dans $GF(2) = \{0,1\}$.

Il n'y a pas de retenue dans $GF(2) = \{0,1\}$ ($1 + 1 = 0$ dans GF(2)).

Soient $a_0 = 0x53$ et $a_0 = 0xCA$ deux octets :

- $a_0 = 0x53 \Rightarrow (01010011)$
- $a_1 = 0x53 \Rightarrow (11001010)$
- $a_0 + a_1 = 01010011 \oplus 11001010 = 10011001$
- $a_0 + a_1 = 0x99$

Multiplication dans GF(2^8)

La multiplication est un plus complexe. Elle se fait en deux étapes :

- **Multiplication polynomiale classique** : on multiplie les deux polynômes, comme on le ferait en algèbre, en utilisant les règles de $GF(2)$.
- **Réduction modulaire** : le résultat de la multiplication peut avoir un degré supérieur à 7. Pour que le résultat reste dans $GF(2^8)$, on effectue une division euclidienne par le **polynôme irréductible d'AES** ($m(X) = X^8 + x^4 + X^3 + X + 1$) et on prend le reste.

Exemple : multiplions $a_0 = 0x4D$ par $a_1 = 0xE11$ dans $GF(2^8)$:

- $a_0 = 0x4D \Rightarrow (01001101) \Rightarrow X^6 + X^3 + X^2 + 1$
- $a_1 = 0xE1 \Rightarrow (11100001) \Rightarrow X^7 + X^6 + X^5 + 1$
- $a_0 \times a_1 = (X^6 + X^3 + X^2 + 1)(X^7 + X^6 + X^5 + 1)$
- $a_0 \times a_1 = X^{13} + X^{12} + X^{11} + X^{10} + X^5 + X^3 + X^2 + 1$

- Il faut le réduire modulo $m(X)$. Une méthode simple pour faire cette réduction est de remplacer les termes X^8 par $x^4 + X^3 + X + 1$, car $m(X) \bmod m(X) = 0 \Rightarrow X^8 = x^4 + X^3 + X + 1 \bmod m(X)$.
- D'où $a_0 \times a_1 \bmod m(X) = X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$.
- $a_0 \times a_1 = 0x7F$ dans $GF(2^8)$.