**RESUME SUR LA CRYPTOGRAPHIE**

## Définition:

La cryptographie est essentiellement la science qui utilise une logique mathématique pour maintenir l'information sécurisée. Elle permet de stocker de manière sécurisée des informations sensibles ou de transmettre des informations de manière sécurisée à travers des réseaux peu sûrs pour éviter qu'ils ne soit piraté, masqué ou modifié. Il existe différentes terminologies utilisées en cryptographie. Ces terminologies sont les suivants :

- **Plaintext** (Texte en clair) : c'est l'information qu'un expéditeur veut transmettre à un récepteur.
- **Encryption** (Cryptage) : le cryptage est la procédure d'encodage de messages (ou d'informations) de telle sorte que les écoutes ou les pirates ne peuvent pas le lire, mais les parties autorisées peuvent. Dans un schéma de cryptage, le message ou l'information (c'est-à-dire le texte en clair) est crypté à l'aide d'un algorithme de cryptage, ce qui le transforme en un texte chiffré illisible.
- **Ciphertext** (Texte chiffré) : le texte chiffré est le résultat du cryptage effectué en texte clair à l'aide d'un algorithme appelé un chiffrement.
- **Cipher** (chiffrement) : un chiffrement est un algorithme pour l'exécution du cryptage ou du décryptage - une série d'étapes bien définies qui peuvent être suivies comme une procédure.
- **Decryption** (Décryptage) : il s'agit du processus de décodage du texte chiffré et de le récupérer dans le format en texte clair.
- **Cryptographic key** (Clé cryptographique): Généralement, une clé ou un ensemble de clés est impliqué dans le cryptage d'un message. Une clé identique ou un ensemble de clés identiques est utilisé par la partie légitime pour décrypter le message. Une clé est une information (ou un paramètre) qui détermine la sortie fonctionnelle d'un algorithme ou d'un chiffrement cryptographique.

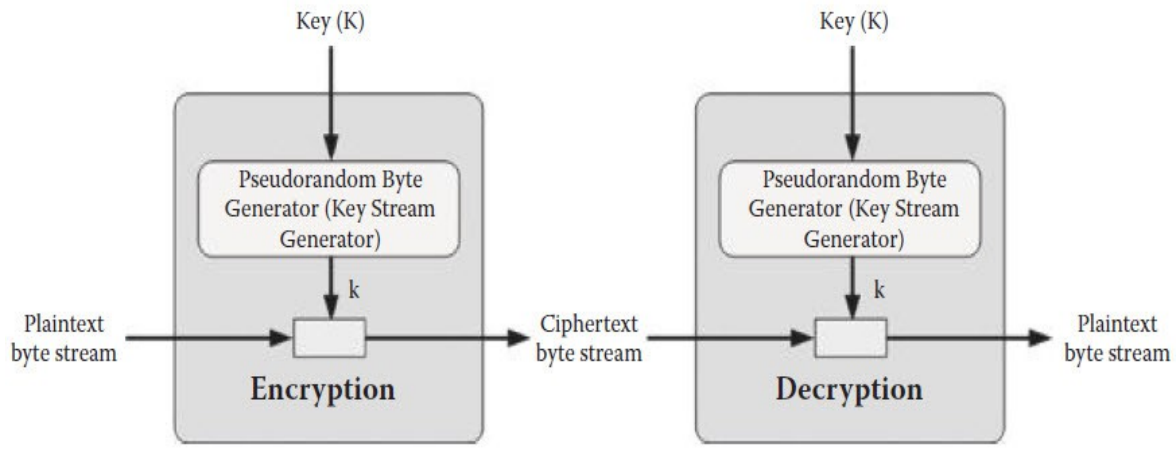



**Figure 3.1 Diagramme opérationnel pour un chiffrement de flux**

• **Stream cipher** (chiffrement de flux) : un chiffrement de flux est une méthode de cryptage de texte (pour produire du texte chiffré) dans laquelle une clé et un algorithme cryptographique sont appliqués à chaque chiffre binaire dans un flux de données, un bit à la fois. Cette méthode n'est pas très utilisée dans la cryptographie moderne. Un diagramme de flux opérationnel typique du chiffrement du flux est illustré dans la Figure 3.1.

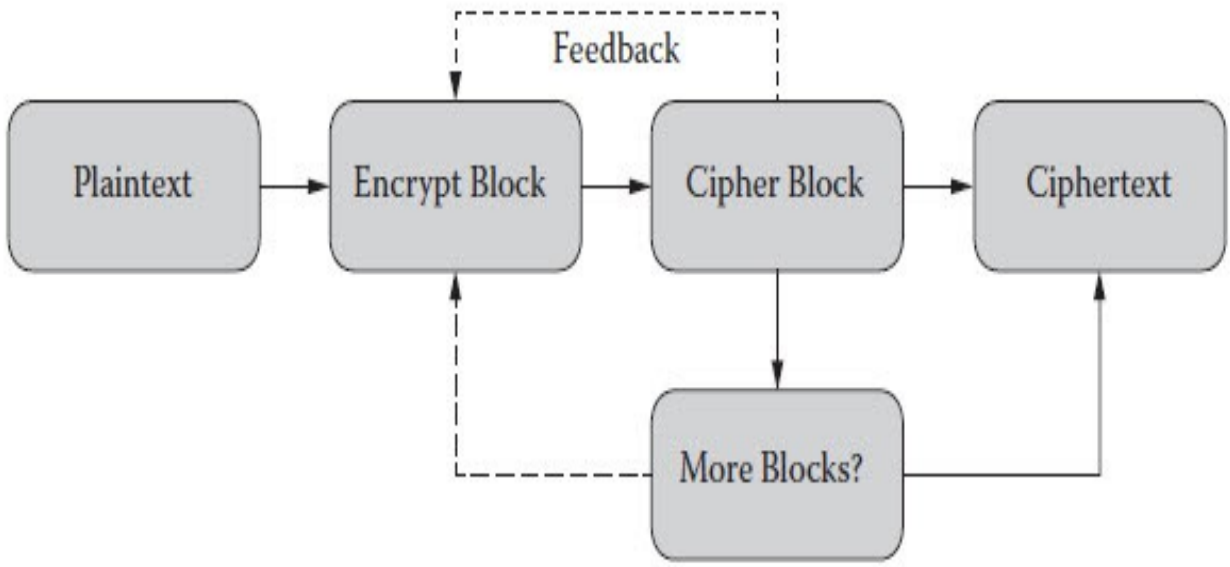



**Figure 3.2 Diagramme opérationnel d'un chiffrement de bloc**

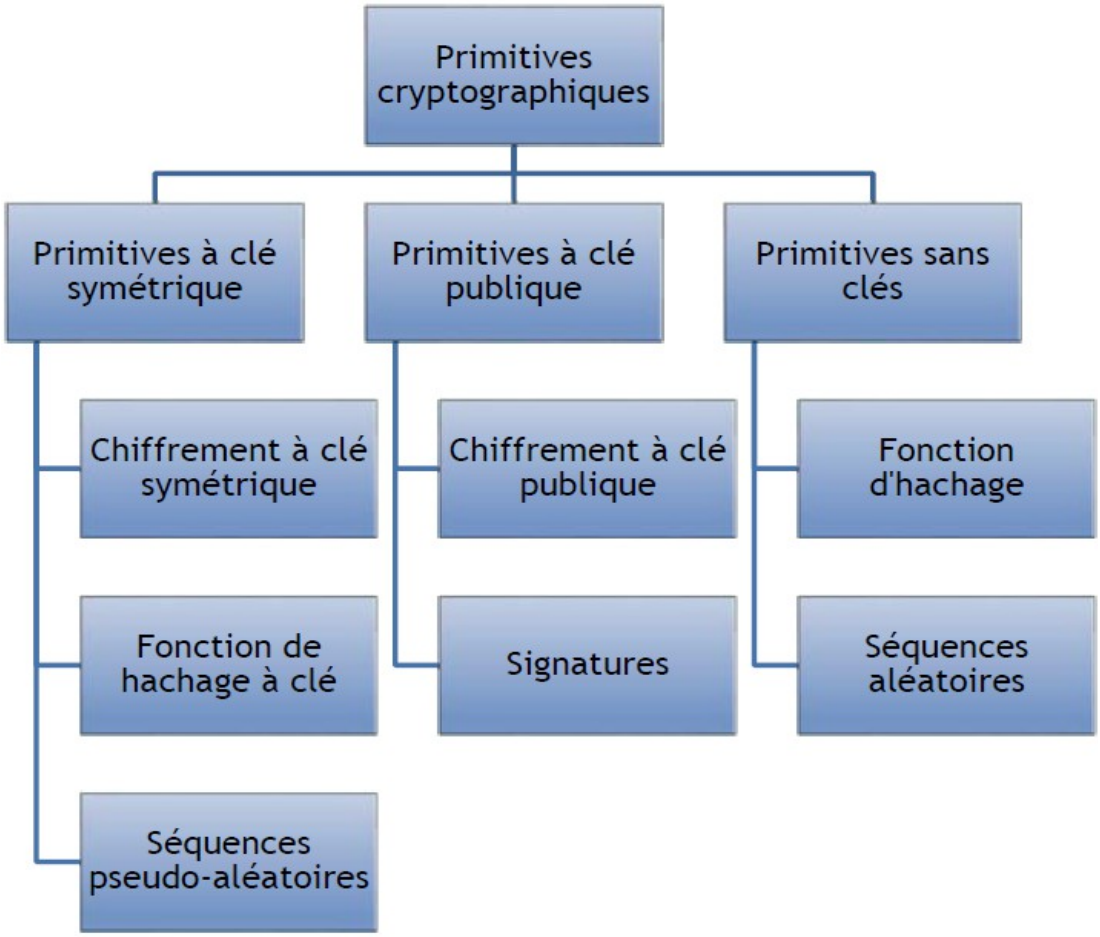



**Figure 3.3. Classification des primitives cryptographiques**

• **Block cipher** (Chiffre de bloc) : Un chiffrement de bloc est une méthode de cryptage de texte (pour produire un texte chiffré) dans lequel une clé et un algorithme cryptographique sont appliqués à un bloc de données (par exemple, 64 bits) à la fois en tant que groupe plutôt qu'un bit à un temps. Un schéma pour une opération de chiffrement de bloc est illustré dans la Figure 3.2.

• **Cryptanalyse :** Cryptanalyse se réfère à l'étude des codes, du texte chiffré ou des crypto systèmes (c'est-à-dire des systèmes de code secret) dans le but de trouver des faiblesses qui permettraient de récupérer le texte clair du texte chiffré, sans nécessairement connaître la clé ou l'algorithme utilisé durant le chiffrement.

• **Signature numérique :** une signature numérique est une signature électronique qui peut être utilisée pour authentifier l'identité de l'expéditeur d'un message ou le signataire d'un document, en outre pour s'assurer que le contenu original du message ou du document qui a été envoyé est inchangé. Les signatures numériques sont généralement

facilement transportables, ne peuvent être imitées par quelqu'un d'autre et peuvent être automatiquement horodatées.

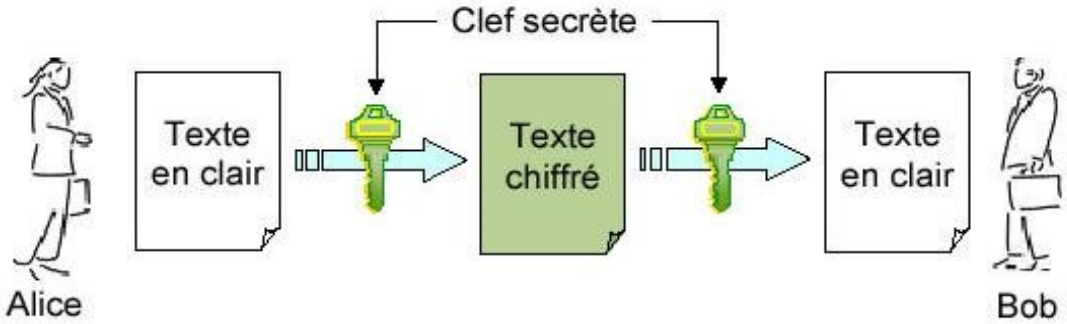
• **Certificat numérique :** il existe une différence entre la signature numérique et le certificat numérique. Un certificat numérique fournit un moyen de prouver l'identité de quelqu'un dans les transactions électroniques. La fonction de celui-ci pourrait être considérée comme un passeport ou un permis de conduire dans les interactions en face à face. Par exemple, un certificat numérique peut être une «carte de crédit» électronique qui établit les informations d'identification de quelqu'un lors de transactions commerciales ou autres via le Web. Il est délivré par une autorité de certification (CA). En règle générale, une telle carte contient le nom de l'utilisateur, un numéro de série, des dates d'expiration, une copie de la clé publique du titulaire du certificat (utilisé pour chiffrer des messages et des signatures numériques) et la signature numérique de l'autorité émettrice de certificat afin qu'un destinataire puisse vérifier que le certificat est réel.

• **Autorité de certification** (CA) : une autorité de certification est une autorité dans un réseau qui émet et gère les informations de sécurité et les clés publiques pour le chiffrement des messages.

Comme présenté dans la Figure 3.3, **les primitives cryptographiques peut être classées en trois catégories, à savoir, primitives à clé symétrique, primitives à clé publique, et primitives sans clés.**

# I- La cryptographie symétrique

La cryptographie symétrique (ou le cryptage des clés symétriques) est une classe d'algorithmes de cryptographie qui utilisent les mêmes clés cryptographiques pour le cryptage du texte clair et le décryptage du texte chiffré. Figure 3.4 montre l'aperçu des étapes de la cryptographie Figure




**3.4 Modèle opérationnel de la cryptographie symétrique**

# A) Le chiffrement AES

Advanced Encryption Standard ou AES [18], aussi connu sous le nom de Rijndael, est un algorithme de chiffrement symétrique où il est le nouveau standard de chiffrement pour les organisations du gouvernement des États-Unis. Il a été approuvé par la NSA (National Security Agency) dans sa suite B1 des algorithmes cryptographiques. Il est actuellement le plus utilisé et le plus sûr. l'AES remplace le DES (choisi comme standard dans les années 1970) qui de nos jours devenait obsolète, car il utilisait des clefs de 56 bits seulement. L'AES est défini dans chacun de :

- FIPS PUB 197: Advanced Encryption Standard (AES)7
- ISO/IEC 18033-3: Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers8

## A1. Aperçu de l'Algorithme AES

Le chiffrement AES est presque identique au bloc de chiffrement Rijndael. Le bloc **Rijndael** et la taille des touches varient entre 128, 192 et 256 bits. Toutefois, la norme AES ne requiert qu'une taille de bloc de 128 bits. Par conséquent, seul **Rijndael** avec une longueur de bloc de 128 bits est connu sous le nom de l'algorithme AES. Dans ce chapitre, nous ne discutons que la version standard de Rijndael avec une longueur de bloc de 128 bits.

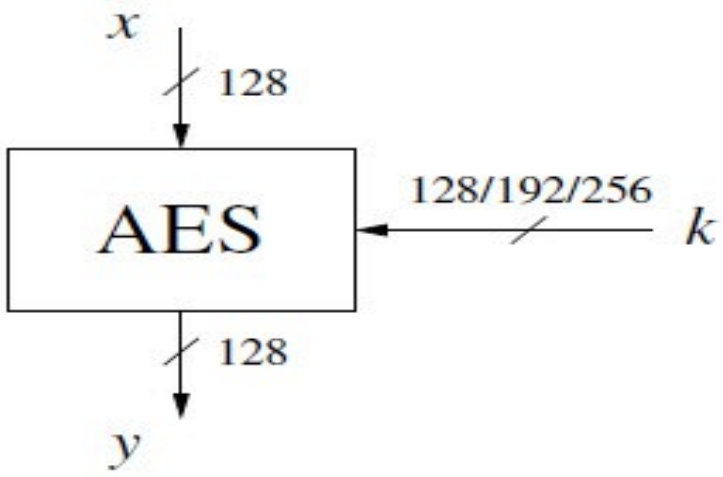



**Figure 3.5 Paramètres d'entrée / sortie AES**

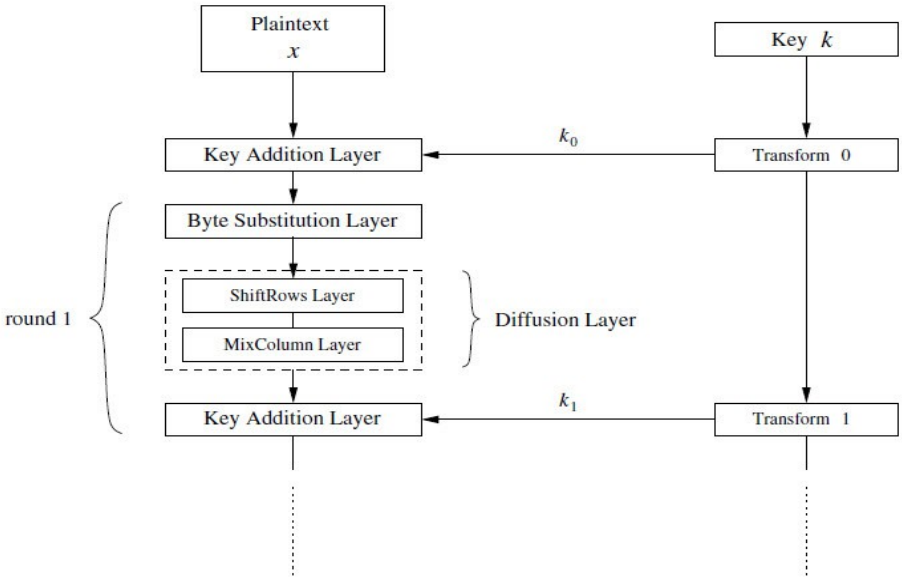
Comme mentionné précédemment, trois longueurs de clés doivent être supportées par Rijndael car il s'agissait d'une exigence de conception **NIST**(National Institute of Standards and Technology)Le nombre de cycles internes du chiffre est une fonction de la longueur de la clé selon le Tableau 3.1.

| Longueur des clés | # tours = n |
|---|---|
| 128 bits | 10 |
| 192 bits | 12 |
| 256 BITS | 14 |

**Tableau 3.1 Longueurs et nombre de tours pour AES**

**Standard, N. F. (2001). Announcing the advanced encryption standard (AES). Federal Information**
**Processing Standards Publication, 197, 1-51.**
**https://www.iso.org/standard/54531.html**
**Rijndael est le nom de l'algorithme de chiffrement symétrique employé par le standard AES.**
**NIST: National Institute of Standards and Technology (www.nist.gov)**

AES se compose de couches où chaque couche manipule tous les 128 bits du chemin de données. Le chemin de données est également appelé l'état de l'algorithme. Il n'y a que trois types de couches différentes. Chaque tour, à l'exception de la première, se compose des trois couches, comme le montre la Figure 3.6 : le texte clair est désigné par x le texte chiffré par y et le nombre de tours comme n Par ailleurs, le n du dernier cycle ne fait pas appel à la transformation MixColumn, ce qui rend le schéma de cryptage et de décodage symétrique.



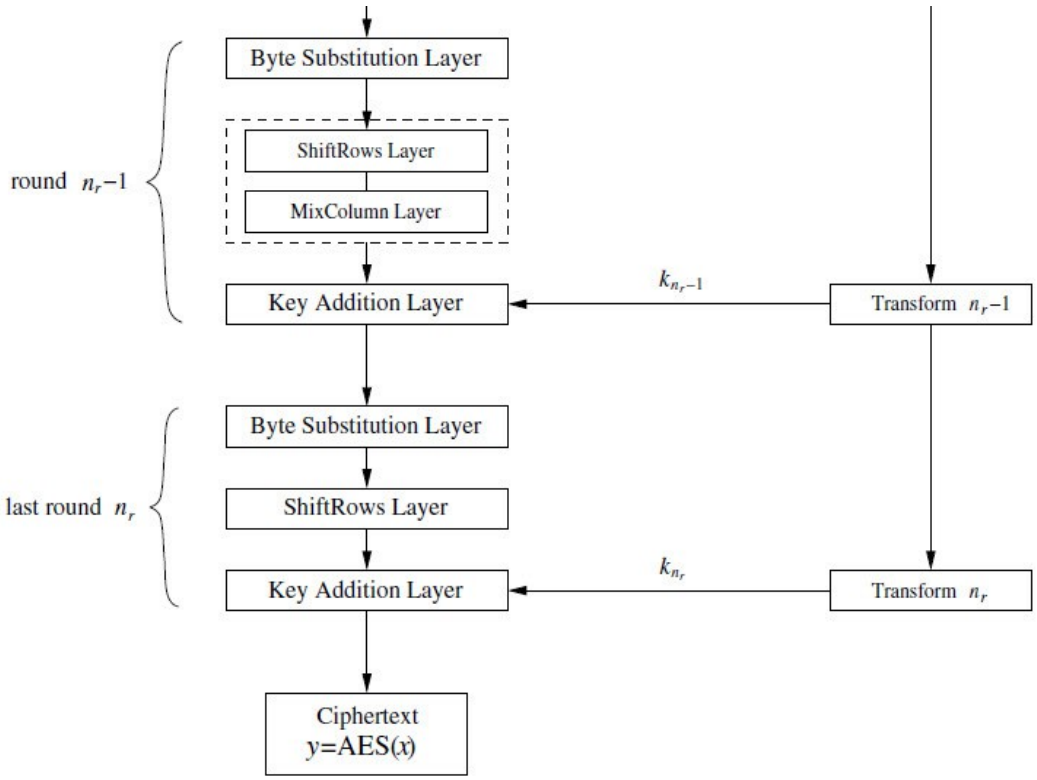



**Figure 3.6 Schéma de bloc du cryptage AES**

## II Le cryptographie asymétrique

La cryptographie à clé publique (PKC), également appelée cryptographie asymétrique, se réfère à un algorithme cryptographique qui nécessite deux clés distinctes, dont l'une est secrète (ou privée) et l'autre public. Bien que différentes, les deux parties de cette paire de clés sont liées mathématiquement. La Figure 3.20 montre une vue d'ensemble des opérations PKC.

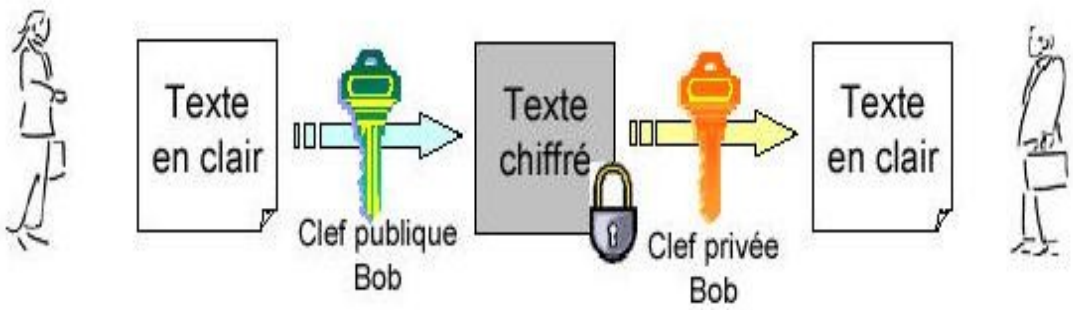



**Figure 3.20 Modèle opérationnel de la cryptographie asymétrique (PKC)**

La cryptographie à clé publique permet le cryptage et décryptage. Ces deux opérations permettent à deux parties communicantes de déguiser les données qu'elles se transmettent. L'expéditeur crypte les données avant de les envoyer via un support de communication . Le récepteur décrypte ou

déchiffre les données après leur réception. Tandis que pendant la transmission, les données cryptées ne sont pas comprises par un tiers illégitime.

a) Le chiffrement RSA

Le chiffrement RSA est un algorithme de cryptographie asymétrique, qui est très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman.

- Le chiffrement RSA utilise une paire de clés (des nombres entiers) composée d'une clé publique pour chiffrer et d'une clé privée pour déchiffrer des données confidentielles.
- Les deux clés sont créées par une personne, souvent nommée par convention Alice, qui souhaite que lui soient envoyées des données confidentielles.
- Alice rend la clé publique accessible. Cette clé est utilisée par ses correspondants (Bob, etc.) pour chiffrer les données qui lui sont envoyées. La clé privée est quant à elle réservée à Alice, et lui permet de déchiffrer ces données. La clé privée peut aussi être utilisée par Alice pour signer une donnée qu'elle envoie, la clé publique permettant à n'importe lequel de ses correspondants de vérifier la signature.

## A. Fonctionnement RSA

L'algorithme RSA se base sur trois étapes, à savoir,

1) création des clés,
2) Chiffrement du message, et
3) Déchiffrement du message.

## A.1. Création des clés

L'étape de création des clés est à la charge d'Alice. Elle n'intervient pas à chaque chiffrement car les clés peuvent être réutilisées, la difficulté première, que ne règle pas le chiffrement, est que Bob soit bien certain que la clé publique qu'il détient est celle d'Alice. Le renouvellement des clés n'intervient que si la clé privée est compromise, ou par précaution au bout d'un certain temps.

1. Choisir p et q, deux nombres premiers distincts ;
2. calculer leur produit n = pq, appelé module de chiffrement ;
3. calculer $\varphi(n) = (p - 1)(q - 1)$ (c'est la valeur de l'indicatrice d'Euler en n) ;
4. choisir un entier naturel e premier avec $\varphi(n)$ et strictement inférieur à $\varphi(n)$,

appelé exposant de chiffrement ;

5. calculer l'entier naturel d, inverse de e modulo $\varphi(n)$, et strictement

inférieur à φ(n),
appelé exposant de déchiffrement ; d peut se calculer efficacement par l'algorithme d'Euclide étendu.
Le couple (n,e) est la clé publique du chiffrement, alors que le nombre d est sa clé privée,
sachant que l'opération de déchiffrement ne demande que la clef privée d et l'entier n, connu par la clé publique (la clé privée est parfois aussi définie comme le triplet (p, q, d)).

## A.2. Chiffrement du message

Si M est un entier naturel strictement inférieur à n représentant un message, alors le message chiffré sera représenté par

$$C \equiv M^e (mod\ n)$$

l'entier naturel C étant choisi strictement inférieur à n.

## A. 3. Déchiffrement du message

Pour déchiffrer C, on utilise d, l'inverse de e modulo (p – 1)(q – 1), et l'on retrouve le message clair M par

$$M \equiv C^d (mod\ n)$$

## 3.4. Les fonctions de hachage cryptographique

Les fonctions de hachage sont extrêmement utiles et apparaissent dans presque toutes les applications de sécurité de l'information. Une fonction hash est une fonction mathématique qui convertit une valeur d'entrée numérique en une autre valeur numérique compressée.
L'entrée de la fonction hash est de longueur arbitraire, mais la sortie est toujours de longueur fixe. Les valeurs retournées par une fonction hash sont appelées digest de message ou simplement valeurs de hash.

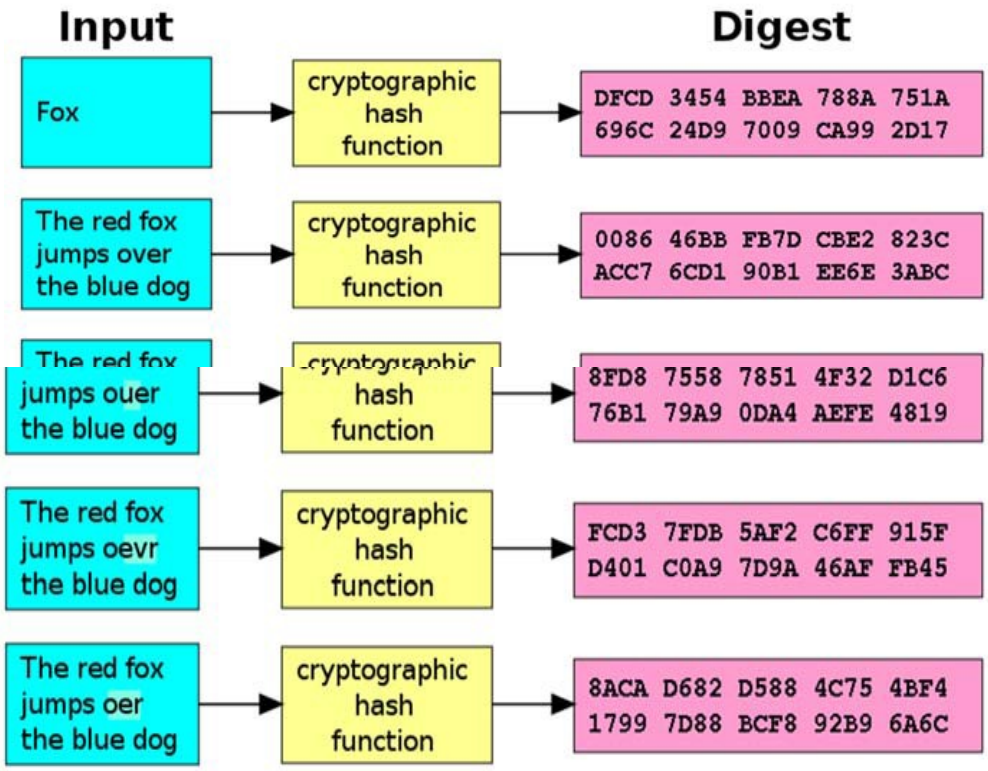



Figure 3.21 Une fonction de hachage cryptographique (spécifiquement, SHA-1) en action

## a) Caractéristiques des fonctions de hash

Les caractéristiques typiques des fonctions de hachage sont :

**Sortie de longueur fixe**

• La fonction Hash recouvre des données de longueur arbitraire à une longueur fixe. Ce processus est souvent appelé hachage des données.

• En général, le hash est beaucoup plus petit que les données d'entrée, par conséquent, les fonctions hash sont parfois appelées fonctions de compression.

• Étant donné qu'un hash est une représentation plus petite d'une donnée plus grande, elle est également appelée Digest.

• La fonction Hash avec sortie n bit est appelée une fonction hash n-bit. Les fonctions populaires de hachage génèrent des valeurs comprises entre 160 et 512 bits.

**Efficacité de l'opération**

• Généralement pour toute fonction h avec entrée x, le calcul de h(x) est une opération rapide.

• Les fonctions de hash par calcul sont beaucoup plus rapides qu'un cryptage symétrique.

## b) Propriétés des fonctions de hachage

Pour être un outil cryptographique efficace, la fonction d'hachage doit avoir les propriétés suivantes :

- **résistance à la préimage:** pour toute valeur de hachage $h$, il devrait être difficile de trouver un message $m$ tel que $h = hash(m)$ ; cette notion est liée à la notion de fonction à sens unique ; les fonctions qui n'ont pas cette propriété sont vulnérables aux attaques de préimage ;
- **résistance à la seconde préimage:** pour toute entrée $m_1$, il devrait être difficile de trouver une entrée différente $m_2$ telle que $hash(m_1) = hash(m_2)$ ; les fonctions qui n'ont pas cette propriété sont vulnérables aux attaques de seconde préimage;
- **résistance aux collisions :** il doit être difficile de trouver deux messages différents $m_1$ et $m_2$ tels que $hash(m_1) = hash(m_2)$ ; une telle paire de messages est appelée une collision de hachage cryptographique ; pour obtenir cette propriété, il faut une valeur de hachage au moins deux fois plus longue que celle requise pour obtenir la résistance à la préimage ; si la clé n'est pas assez longue, une collision peut être trouvée par une attaque des anniversaires.

### c) L'algorithme HMAC

Un HMAC [21] (keyed-hash message authentication code- code d'authentification d'une empreinte cryptographique de message avec clé), est un type de code d'authentification de message calculé en utilisant une fonction de hachage cryptographique en combinaison avec une clé secrète. Il peut être utilisé pour vérifier simultanément l'intégrité de données et l'authenticité d'un message. N'importe quelle fonction itérative de hachage, comme MD5 ou SHA-1, peut être utilisée dans le calcul d'un HMAC ; le nom de l'algorithme résultant est HMAC-MD5 ou HMAC-SHA-1. La qualité cryptographique du HMAC dépend de la qualité cryptographique de la fonction de hachage et de la taille et la qualité de la clé.

La fonction HMAC est définie comme suit :

$$HMAC_K(m) = h\Big((K \oplus opad)||h\big((K \oplus opad)||m\big)\Big)$$

avec :

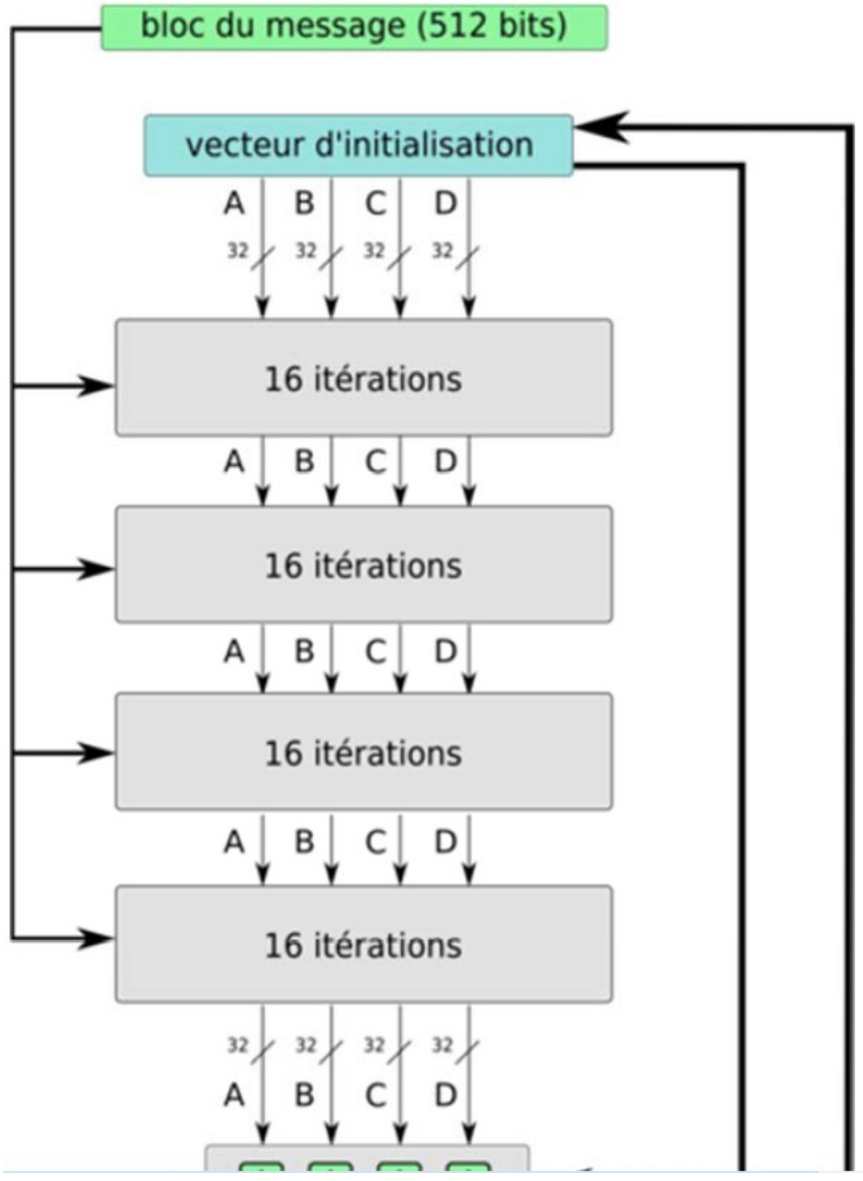
- *h* : une fonction de hachage itérative,
- *K* : la clé secrète complétée avec des zéros pour qu'elle atteigne la taille de bloc de la fonction h
- *m* : le message à authentifier,
- "||" désigne une concaténation et "⊕" un « ou » exclusif,
- *ipad* et *opad*, chacune de la taille d'un bloc, sont définies par : *ipad* = 0x363636...3636 et *opad* = 0x5c5c5c...5c5c. Donc, si la taille de bloc de la fonction de hachage est 512

bits, *ipad* et *opad* sont 64 répétitions des octets, respectivement, 0x36 et 0x5c.

HMAC-SHA-1 et HMAC-MD5 sont utilisés dans les protocoles IPsec et TLS.

### 3.4.4 L'algorithme MD5

MD5 (Message Digest 5) est une fonction de hachage cryptographique qui calcule, à partir d'un fichier numérique, son empreinte numérique (en l'occurrence une séquence de 128 bits ou 32 caractères en notation hexadécimale) avec une probabilité très forte que deux fichiers différents donnent deux empreintes différentes.



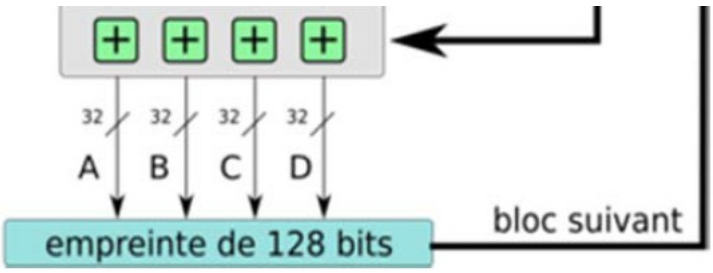



**Figure 3.22 Vue générale de MD5**

Comme présenté dans la figure 3.22, MD5 travaille avec un message de taille variable et produit une empreinte de 128 bits. Le message est divisé en blocs de 512 bits, on applique un remplissage de manière à avoir un message dont la longueur est un multiple de 512. Le remplissage se présente comme suit :

• on ajoute un 1 à la fin du message ;

- on ajoute une séquence de '0' (le nombre de zéros dépend de la longueur du remplissage nécessaire) ;
- on écrit la taille du message, un entier codé sur 64 bits.

L'algorithme principal travaille avec un état sur 128 bits. Il est lui-même divisé en 4 mots de 32 bits : A, B, C et D. Ils sont initialisés au début avec des constantes. L'algorithme utilise ensuite les blocs provenant du message à hacher, ces blocs vont modifier l'état interne. Les opérations sur un bloc se décomposent en quatre rondes (étapes), elles-mêmes subdivisées en 16 opérations similaires basées sur une fonction non linéaire F qui varie selon la ronde, une addition et une rotation vers la gauche. Les quatre fonctions non linéaires disponibles sont :

- $F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$
- $G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$
- $H(B,C,D) = B \oplus C \oplus D$
- $I(B,C,D) = C \oplus (B \wedge \neg D)$

MD5 peut s'écrire sous cette forme en pseudo-code.

```
//Définir r comme suit :
var entier[64] r, k
r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}

//MD5 utilise des sinus d'entiers pour ses constantes :
pour i de 0 à 63 faire
   k[i] := floor(abs(sin(i + 1)) × 2^32)
fin pour

//Préparation des variables :
var entier h0 := 0x67452301
var entier h1 := 0xEFCDAB89
var entier h2 := 0x98BADCFE
var entier h3 := 0x10325476
```

```
//Préparation du message (padding) :
ajouter le bit "1" au message
ajouter le bit "0" jusqu'à ce que la taille du message en bits soit égale à 448 (mod 512)
ajouter la taille du message codée en 64-bit little-endian au message

//Découpage en blocs de 512 bits :
pour chaque bloc de 512 bits du message
   subdiviser en 16 mots de 32 bits en little-endian w[i], 0 ≤ i ≤ 15
```

```
//initialiser les valeurs de hachage :
var entier a := h0
var entier b := h1
var entier c := h2
var entier d := h3
```

```
    //Boucle principale :
    pour i de 0 à 63 faire
        si 0 ≤ i ≤ 15 alors
            f := (b et c) ou ((non b) et d)
            g := i
        sinon si 16 ≤ i ≤ 31 alors
            f := (d et b) ou ((non d) et c)
            g := (5×i + 1) mod 16
        sinon si 32 ≤ i ≤ 47 alors
            f := b xor c xor d
            g := (3×i + 5) mod 16
        sinon si 48 ≤ i ≤ 63 alors
           f := c xor (b ou (non d))
           g := (7×i) mod 16
        fin si
        var entier temp := d
        d := c
        c := b
        b := ((a + f + k[i] + w[g]) leftrotate r[i]) + b
        a := temp
    fin pour

    //ajouter le résultat au bloc précédent :
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
fin pour
```