REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
Université de Bejaia

SECURITE INFORMATIQUE

Support de cours

Dr. FARAH Zoubeyr

Maitre de conférences à l'université de Bejaia.

Table des matières

Liste des figures

Liste des tableaux

Introduction

L'utilisation des nouvelles technologies prend de plus en plus d'ampleur dans notre quotidien du fait de leur accessibilité, notamment avec la prolifération des outils les intégrant (Smart phones, smart TV, objets connectés, outils de travail, systèmes d'accès …etc). Les utilisateurs de ces nouvelles technologies sont souvent novices, naïfs ou pas forcément conscients des risques qui existent dans le domaine de l'informatique.

En effet, la sécurité informatique ou *Cybersécurité*, implique la protection des utilisateurs et des systèmes contre d'éventuelles actions malveillantes. D'où la nécessité de mettre en place des démarches et des mécanismes pour évaluer les risques et définir les objectifs de sécurité à atteindre selon les contextes d'utilisation. Cela constitue un défi majeur pour les concepteurs de systèmes afin d'assurer la protection des utilisateurs sans pour autant les conditionner avec des détails techniques.

Ainsi, la sécurité informatique est un ensemble de moyens techniques, organisationnels, juridiques et humains utilisés pour garantir la sécurité des systèmes manipulés, notamment la sécurité des données et des communications.

Dans le but de fournir un support de base englobant des notions aussi bien théoriques que pratiques sur la sécurité informatique, nous avons élaboré ce cours qui considère le lecteur comme débutant dans le domaine de la sécurité. Le cours se focalise beaucoup plus sur la sécurité des données et des communications, qui est assurée principalement par l'utilisation de la Cryptographie (Science de codage), considérée comme étant le *noyau* de la sécurité informatique. L'usage de la cryptographie a débuté bien avant l'apparition des ordinateurs et était utilisée principalement pour produire des écritures incompréhensibles et pour faire passer secrètement des messages.
Avec l'avènement des ordinateurs, la cryptographie a connu une grande évolution en nombre d'algorithmes développés ainsi que de techniques d'attaques qui ne cessent d'accroître en efficacité. Par conséquent, la sécurité a connu l'émergence d'autres domaines en relation avec la cryptographie, tels que le hachage, la signature et la gestion de clés.

Le présent cours introduit les concepts de base de la sécurité appuyés par des exemples applicatifs afin de faciliter à l'étudiant sa compréhension. Il est destiné aux étudiants de Master 1 Informatique, mais aussi à tous ceux qui désirent s'initier à la sécurité informatique. Cependant, afin de bien cerner les notions introduites dans ce support, il est requis d'avoir, au préalable, des connaissances de base sur la théorie du codage en informatique [DUM 2013].

Ce support de cours est structuré en cinq chapitres : le premier chapitre introduit les notions et le vocabulaire de base liés au domaine de la sécurité informatique. Le deuxième chapitre décrit la cryptographie en détails. Le troisième chapitre présente les algorithmes de chiffrement moderne. Quant au quatrième chapitre, il est consacré à la signature numérique et aux fonctions de hachage, des mécanismes utilisés pour assurer les objectifs d'intégrité et d'authentification. Enfin, le chapitre cinq explique la gestion de clés ainsi que les différentes techniques de gestion existantes.

Chapitre I : Introduction à la sécurité

1. Introduction

Le domaine des systèmes informatiques connait, de nos jours, un fulgurant progrès en termes d'échange d'information et d'ouverture sur le monde extérieur. En effet, les systèmes informatiques sont de plus en plus accessibles depuis des machines de moins en moins contrôlées. De ce fait, la sécurité informatique a pour objectif principal la protection des données et des ressources en mettant en place des mécanismes de contrôle qui permettent d'assurer le bon fonctionnement du système utilisé.

Ce chapitre introduit les notions de base des aspects techniques et organisationnels de la sécurité informatique.

2. Vocabulaire de base

a) **Sûreté :** protection contre les actions non intentionnelles.

b) **Sécurité** : protection contre les actions intentionnelles malveillantes.

c) **Menace :** Événement, d'origine accidentelle ou délibérée, capable s'il se réalise de causer un dommage à un système donné.

d) **Vulnérabilité** : Une vulnérabilité est une faiblesse dans le système qui peut être exploitée par une menace.

e) **Risque** : Association d'une menace aux vulnérabilités qui permettent sa réalisation.

f) **Attaques** : elles représentent les moyens d'exploiter une vulnérabilité. Il peut y avoir plusieurs attaques pour une même vulnérabilité.

g) **Politiques de sécurité**

- Définition des autorités et des ressources,
- Organisation, règles d'usage,
- Spécification des droits.

h) Mécanismes de sécurité

- Moyens pour la mise en œuvre d'une politique. Parmi les moyens, on cite la protection physique, l'authentification par mot de passe, le chiffrement et les listes d'accès.

Exemple illustratif de quelques termes du vocabulaire de base dans une structure informatique composée d'un hôte connecté à un réseau :

***Tableau 1**:Menace, Vulnérabilité, Risque et Attaque sur un Hôte sous réseau*

Terme	***Action***
Menace	Suppression de fichiers importants
Vulnérabilité	Accès au PC sans protection (mot de passe faible)
Risque	Suppression de fichiers VIA un accès réseau non protégé
Attaque	-Utilisation d'un poste connecté au réseau -Logiciel d'intrusion

3. Objectifs de sécurité

Assurer la sécurité d'un système informatique revient à atteindre un ensemble d'objectifs en vue de garantir la protection des informations contre toute divulgation, altération ou destruction. L'accès à ces informations doit être également contrôlé par un accès protégé.

Dans cette section, un ensemble d'objectifs sont présentés.

- **Confidentialité** : empêcher la divulgation d'informations à des entités non autorisées à les connaître. Les entités peuvent être des sites, organisations, personnes, … etc.

- **Authentification :** Il existe deux types d'authentification :

  a. d'une information : prouver qu'une information provient de la source annoncée (auteur, émetteur).

b. d'une entité (ou groupe ou organisation) : prouver que l'identité est bien celle annoncée.

- **Intégrité des informations :** Assurer que les informations n'ont pas été modifiées (ou altérées) par des entités non autorisées ou inconnues. Généralement, l'intégrité est appliquée sur des données en transmission.

- **Signature :** Le moyen de lier une information à une entité.

- **Contrôle d'accès :** Limiter l'accès des ressources aux entités autorisées selon différents privilèges. En pratique, on trouve plusieurs modèles d'accès [GAD 2018].

- **Certification :** L'approbation de l'information par une entité de confiance.

- **Réception :** Approuver la réception de l'information.

- **Anonymat :** Cacher l'identité d'une entité impliquée dans un processus pour des fins de protection.

- **Non-répudiation :** Empêcher les entités de démentir (nier) leurs actions précédentes ou leurs engagements.

4 Menaces sur les systèmes informatiques

Dans un système informatique, les menaces peuvent toucher les composants matériels, logiciels ou les données. Il existe principalement deux types de menaces : les menaces non-intentionnelles (accidentelles) et les menaces intentionnelles.

1. ***Les menaces accidentelles*** : ne supposent aucune préméditation. Dans cette catégorie, sont repris les Bugs logiciels, les pannes matérielles, et autres défaillances "incontrôlables".
2. ***Les menaces intentionnelles :*** sont les actions d'un tiers désirant s'introduire pour relever des informations et/ou porter des modifications sur le système. Dans ce type de menace, on distingue deux sous-familles :

    a) ***menaces intentionnelles passives*** : l'intrus va tenter de dérober les informations par audit ou intrusion, ce qui rend sa détection relativement difficile. En effet, ses actions ne modifient pas le comportement du système.

    b) ***menaces intentionnelles actives*** : l'intrus modifie volontairement le comportement ou le contenu du système. La détection de ce type d'actions est facile, mais il peut être trop tard lorsque l'attaque a déjà eu lieu.

    Les menaces actives peuvent être des interceptions, des interruptions, des fabrications ou des modifications. Le Tableau 2 résume les types de menaces actives pour chaque type de support.

Tableau 2: Types de menaces actives sur un système Informatique

Support / **Menace**	**Données**	**Logiciel**	**Matériel**
Fabrication	Création de fausses données	/	/
Interception	Copie des données	Copies illicites	Vol du support matériel
Interruption	Suppression de données	Introduction de Bugs dans le code	Mise hors tension ou destruction
Modification	Changement de contenu	Introduction de virus dans le code	Changement de composants

Dans la famille des menaces actives, on distingue aussi les menaces qui sont liées aux échanges d'information. En pratique, les entreprises informatisées nécessitent un réseau

sécurisé pour le transfert de leurs données, que ce soit entre les machines de cette entreprise, ou avec des machines externes via le réseau Internet.

À l'origine, c'est Shannon qui, en 1948 puis en 1949 avec Weaver [SHA 1948], a défini les bases d'une transmission de données entre deux parties. Son idée est illustrée par la **figure 1**.

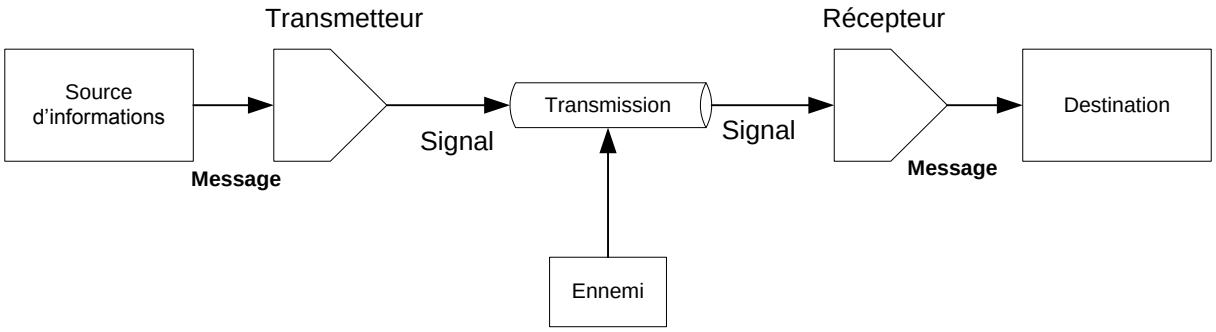

Figure 1: Schéma de transmission de données de Shannon

Dans une communication, nous trouvons quatre catégories de menaces intentionnelles :
(Illustrées à la figure 2 dans le cas d'une communication entre deux entités **A** et **B**) :

– Interruption = problème lié à la disponibilité des données

– Interception = problème lié à la confidentialité des données

– Modification = problème lié à l'intégrité des données

– Fabrication = problème lié à l'authenticité des données

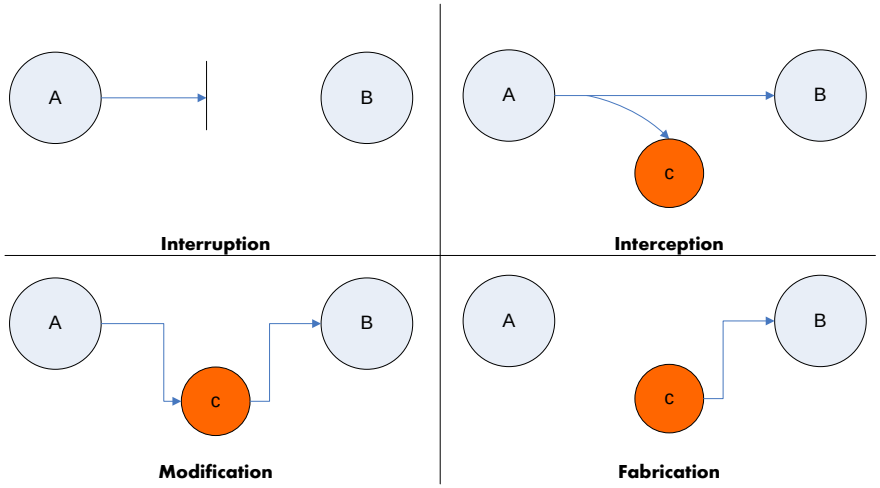

Figure 2: Menaces intentionnelles dans une communication

Chapitre II : Un outil de base pour la sécurité des données « La Cryptographie »

1. Introduction

La protection des systèmes repose principalement sur des techniques de chiffrement appelées *cryptographie*. Le présent chapitre donne les principes de cette science, ainsi que ses bases mathématiques.

2. Concepts de base

2.1 Définition

La *cryptographie* est la science qui utilise les mathématiques pour chiffrer et déchiffrer des données. La cryptographie permet la protection des données stockées ou transmises, à travers des réseaux non sûrs (comme Internet), de telle sorte qu'elles ne puissent être lues par personne à l'exception des destinataires convenus.

Cryptographie = "écriture cachée"
La cryptographie, dont l'etymologie est issue du grec ancien, est composée des deux mots : Kruptos qui signifie "cacher et "Graphein qui signifie "écriture"

Le but fondamental de la cryptographie est de respecter adéquatement les quatre objectifs majeurs de la sécurité suivants : ***la confidentialité, l'intégrité des données, l'authentification et la non-répudiation.***

2.2 Terminologie de base

La cryptographie est utilisée principalement pour assurer la sécurité des données et des communications. Cette section donne la terminologie à connaître dans ce domaine.

- **Communication :** la communication est l'action d'échanger une information entre deux ou plusieurs entités.

- **Message en clair** : information compréhensible par l'expéditeur et le destinataire.
- **Chiffrement** (encryption) : Le processus de transformation d'une information de manière à la rendre incompréhensible.
- **Message chiffré** (cryptogramme) : Résultat de l'opération de chiffrement.
- **Déchiffrement** (décryptage) : Processus de reconstruction du message clair à partir du texte chiffré.
- **Cryptographie :** L'art et la science de garder le secret des informations ou des messages, pratiquée par des ***Cryptographes***.
- **Cryptanalyse :** L'art de décrypter des informations chiffrées, pratiquée par des ***Cryptanalystes***.
- **Cryptologie :** La branche des mathématiques qui traite de la cryptographie et de la cryptanalyse, ses pratiquants sont appelés ***Cryptologues***.
- **Cryptosystème :** Un algorithme cryptographique en plus de toutes les clés possibles et tous les protocoles qui le font fonctionner.

3. Principe de la cryptographie

Transformation de messages clairs (informations) à l'aide d'informations gardées secrètes (les clés de chiffrement).

3.1 Modèle mathématique

Un cryptosystème est un 5-uple $\{P,C,K,E,D\}$ ayant les propriétés suivantes :

1- P *:* est un ensemble qui représente l'espace des messages en clair. Un élément de P s'appelle un "message en clair".
2- C *:* est un ensemble qui représente l'espace des messages chiffrés. Un élément de C s'appelle un "message chiffré" ou encore un "cryptogramme".
3- K *:* est un ensemble qui représente l'espace de clés, ses éléments sont les clés de chiffrement.

4- $E = \{E_K : k \in K\}$: est une famille de fonctions $E_K : P \rightarrow C$, ses éléments sont les fonctions de chiffrement.

5- $D = \{D_K : k \in K\}$:est une famille de fonctions $D_K : C \rightarrow P$, ses éléments sont les fonctions de déchiffrement.

6- À chaque clé k1 $\in K$ est associée une clé k2 $\in K$ telle que $D_{K2}(E_{K1}(M))=M$ pour tout message M $\in P$.

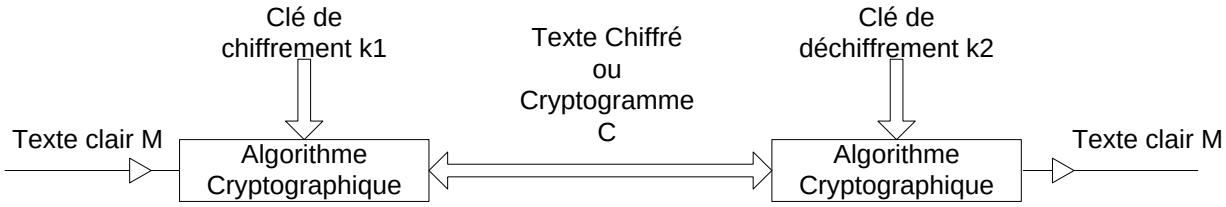

Figure 3: Principe de base du processus de chiffrement

3.2 Exemple d'un système de chiffrement basique

Afin de rendre une communication incompréhensible, on peut simplement changer l'ordre des lettres dans l'alphabet. Avec un décalage des lettres de deux positions vers la gauche, on obtient le tableau suivant :

Tableau 3: Alphabet décalé de deux positions

Alphabet	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
Alphabet décalé	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**	**a**	**b**

Le chiffrement du message « UNIVERSITE DE BEJAIA » donne le message chiffré « WPKXGTUKVG FG DGLCKC »

Exercice :

Déchiffrer le message suivant : « ANMINTQ»

Indication n°1 : les espaces n'ont pas été touchés.

Indication n°2 : l'alphabet a été décalé.

Clé : chaque lettre a été décalée d'un rang à droite par rapport à sa position dans l'alphabet.

Modèle mathématique du chiffrement par décalage :

Si on veut définir un modèle mathématique pour le chiffrement par décalage, on peut procéder comme suit :

a) Identification des lettres par des chiffres pour permettre le calcul.

b) Définition du modèle {*P,C,K,E,D*} avec les valeurs suivantes :

1- *P={0,1,2,3…25},* indices des lettres dans l'alphabet.
2- *C ={0,1,2,3…25},* indices des lettres dans l'alphabet décalé.
3- *K ={0,1,2,3…25}* indique le décalage à appliquer.
4- E ={ E_K :kϵ K } est une famille de fonctions E_K : $P \rightarrow C$, ses éléments sont les fonctions de chiffrement. Pour $x \in P \; et \; k \in K$ on aura : $E_k(x) = x + k \mod 26$, l'opération modulo permet de définir l'ensemble Z_{26}.
5- D ={ D_K :kϵ K } est une famille de fonctions D_K : $C \rightarrow P$,ses éléments sont les fonctions de déchiffrement. Pour $y \in P \; et \; k \in K$ on aura $D_k(y) = y - k \mod 26$, l'opération modulo permet de définir l'ensemble Z_{26}.

4 Cryptographie classique

Cette section présente quelques systèmes de chiffrement classique connus dans la littérature. Le mot classique vient du fait que la plupart de ces systèmes ont été développés avant l'apparition des ordinateurs.

4.1 Chiffrement par transposition

Lors d'un chiffrement par transposition, seul l'ordre des lettres du texte en clair est modifié, les lettres elles-mêmes n'étant pas remplacées par d'autres lettres ou d'autres symboles.

Le système de **transposition le plus simple** consiste à écrire le message en sens inverse. Ainsi, le message *MASTER* devient *RETSAM*. Cependant, ce système échoue avec le chiffrement des palindromes, tel que le mot ''RADAR''.

Les systèmes de transposition les plus utilisés sont décrits ci-dessous.

a) Transposition par blocs

On écrit le message en clair dans un tableau de dimension prédéfinie, on chiffre le texte en le lisant sur le tableau selon un procédé convenu.

Exemple : écriture du texte dans des blocs de **3*3** en ligne.

JE SUIS INFORMATICIEN

Tableau 4: tableau de chiffrement par transposition

J	**E**		**N**	**F**	**O**	**I**	**E**	**N**
S	**U**	**I**	**R**	**M**	**A**			
S		**I**	**T**	**I**	**C**			

La lecture par colonne donne le texte chiffré : **JSSEU IINRTFMIOACI E N**

Déchiffrement : Appliquer un cheminement inverse pour reconstruire le message clair.

b) Transposition avec clé simple

Un mot clé est utilisé pour définir l'ordre de prélèvement des lettres dans le tableau. Cette clé est obtenue en numérotant les lettres du mot clé selon l'ordre de leur apparition dans l'alphabet.

Le message est alors chiffré en l'écrivant dans un tableau dont le nombre de colonnes coïncide avec le nombre de lettres du mot clé et en recopiant ses colonnes dans l'ordre de la clé numérique.

Exemple :

Clé= MASTER

Numérotation selon l'ordre d'apparition dans l'alphabet

3 1 5 6 2 4

M A S T E R

Texte clair : **JE SUIS INFORMATICIEN**

Tableau 5: Tableau de chiffrement par clé simple

M	A	S	T	E	R
3	1	5	6	2	4
J	E		S	U	I
S		I	N	F	O
R	M	A	T	I	C
I	E	N			

Résultat : EUJI S FSOINMIRCATE I N

Déchiffrement :

Cheminement inverse, pour reconstruire le message clair.

4.2 Chiffrement par substitution

À la différence de la transposition, la substitution consiste à remplacer un symbole du texte clair, par un autre symbole, qui peut ne pas figurer dans le message clair.

a) Chiffrement de César

Chiffrement : chaque lettre du texte en clair est remplacée par la lettre située **n** rangs plus loin dans l'alphabet.

Déchiffrement : s'effectue en remplaçant les lettres du texte crypté par celles situées **n** rangs avant dans l'alphabet.

Pour un code décalé de ***trois positions***, le chiffrement est donc le suivant :

Tableau 6: Tableau de chiffrement par substitution

Clair	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Chiffré	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Le même tableau sert au chiffrement et au déchiffrement.

b) Masque Jetable (one-time pad)

L'algorithme consiste à additionner le rang de la lettre à chiffrer au rang de la lettre correspondante du masque, le résultat mod 26 donne le rang de la lettre chiffrée. Le destinataire dispose d'un bloc identique et utilise le masque de la même manière pour déchiffrer chaque lettre du message chiffré. Le masque est utilisé une seule fois, pour un seul message.

Exemple :

M A S Q U E J E T A B L E Texte
+
T B F R G F A R F M I L K Masque
=
F B X F A J J V Y M J V P Chiffré

Fonctionnement : On attribue aux lettres des indices : A=0, B=1, C=2…Z=25. Puis, on effectue des additions *mod 26*.

M+**T**=**F** ⇔**12**+**19**=31 mod 26=**5 …**

Pour le déchiffrement : On soustrait le rang de la lettre à déchiffrer au rang de la lettre correspondante du masque, le résultat mod 26.

c) Chiffrement de Hill (Lester S. Hill, 1891-1961)

Proposé en 1929 [LES 1931], c'est un chiffrement **polygraphique** : le texte n'est pas chiffré lettre par lettre, mais par blocs de lettres. On désigne par 2-chiffre de Hill, le chiffre obtenu en codant les lettres par blocs de deux, 3-chiffre de Hill celui obtenu en codant les lettres par blocs de trois, et ainsi de suite.

Dans une première phase, chaque lettre du texte à chiffrer est remplacée par une valeur numérique, celle de son rang dans l'alphabet :

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Les valeurs numériques des lettres du texte en clair sont ensuite codées par blocs de deux, en leur substituant les valeurs obtenues par le procédé suivant :

• On choisit une matrice A **, régulière**, de format 2x2 et à **coefficients dans Z_{26}** .

Cette matrice va constituer la clé de chiffrement.

A = $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$

• On code chaque paire de lettres du texte en clair *p1p2* à l'aide du produit

$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} ap_1 + bp_2 \\ cp_1 + dp_2 \end{pmatrix}$ (avec modulo 26)

(Si le nombre de lettres du texte en clair est impair, on ajoute une lettre supplémentaire choisie arbitrairement, la lettre x par exemple)

• On convertit les résultats obtenus en caractères alphabétiques.

Pour déchiffrer un message codé, il suffit d'appliquer la même méthode, en utilisant la matrice inverse $\mathbf{A^{-1}}$ de **A**.

Exemple :Déchiffrer le mot **YUNCNAQBEVTE** sachant qu'il est crypté avec le chiffrement de Hill en utilisant la matrice k= $\begin{matrix} 2 & 1 \\ 1 & 2 \end{matrix}$

Indication : $k^{-1}= \begin{matrix} 18 & 17 \\ 17 & 18 \end{matrix}$ avec un produit matriciel on obtient le mot SOIXANTE NEUF

Chapitre III Algorithmes de chiffrement moderne

1. Introduction

La cryptographie a connu d'importants progrès avec l'émergence des ordinateurs. En effet, l'automatisation des calculs, de plus en plus complexes, a permis à la cryptologie d'évoluer rapidement, donnant naissance à plusieurs algorithmes de chiffrement exécutables sur ordinateur. Ces algorithmes manipulent principalement des informations numériques (Bits).

On distingue deux grands types de chiffrement (ou algorithmes de chiffrement), les algorithmes à **clé secrète** et les algorithmes **à clé publique**. Ces deux classes sont détaillées dans ce chapitre.

2. Chiffrement à clé secrète

Les algorithmes de chiffrement à clé secrète (ou symétriques) sont ceux pour lesquels les interlocuteurs partagent une même clé secrète de chiffrement et de déchiffrement. L'emploi d'un algorithme à clé secrète nécessite l'échange préalable d'un secret entre les communicants.

Propriétés :

- Les clés sont identiques : $K_{Encryption} = K_{Decryption} = K$,
- La clé doit rester secrète,
- Opérations Chiff/Déchiff rapides
- Le principal désavantage réside dans la distribution de clés, c'est pourquoi on utilise souvent des échanges sécurisés pour transmettre les clés.

La cryptographie à clé secrète peut être classéee en deux catégories : Système de chiffrement par flot (de flux ou par stream) et système de chiffrement par blocs.
Les schémas de chiffrement par flot traitent l'information bit(ou octet) par bit(ou octet). Ils sont très rapides et légers, ils peuvent être traités avec des outils dotés de capacité mémoire

et processeur limités. Ce type de chiffrement est parfaitement adapté à la cryptographie en temps réel, la cryptographie militaire ou les communications téléphoniques sécurisées.

Les schémas par blocs sont plus lents et nécessitent plus de moyens mémoire/processeur que les schémas par flots. Mais ils sont bien adaptés à la cryptographie civile comme la sécurité des transactions bancaires. Dans un système par blocs, chaque clair est découpé en blocs de même longueur et chiffré bloc par bloc.

2.1 RC4, un système de chiffrement par flot

Conçu par *Ronald Rivest* de RSA Security en 1987 [PAU 2011], officiellement nommé **Rivest Cipher 4**. Il était parmi les systèmes de chiffrement par flot les plus utilisés. On le retrouve notamment dans le standard SSL/TLS, dans Oracle Secure SQL ou encore dans le protocole WEP (Wired Equivalent Privacy, de la norme 802.11).

a. Fonctionnement du RC4

- Chiffrement par flot sur les **octets**
- Utilise une clé **K** de taille comprise entre 1 et 256 Octets
- Un tableau **T[]** de taille 256 Octets
- Un tableau **S[]** de taille 256 Octets

La clé RC4 permet d'initialiser le tableau (T) de 256 octets en répétant la clé autant de fois que nécessaire pour remplir le tableau. Par la suite, des opérations très simples sont effectuées : les octets sont déplacés dans le tableau, des additions sont effectuées, etc. Le but de ces opérations est d'avoir une répartition aussi aléatoire que possible des valeurs du tableau T.
L'algorithme agit en trois phases :

- **Initialisation**

Initialement, les cellules de **S** reçoivent une valeur égale à leur position (i.e., S[0]=0, S[1]=1, ...) Le vecteur temporaire de longueur T est destiné à recevoir la clé. Si la longueur de la clé K

est égale à 256 octets, K est simplement transféré dans T. Si K est inférieur à 256 octets, il est recopié dans **T** jusqu'à atteindre la taille de **T[]**. L'algorithme 1 illustre cette étape d'initialisation.

Algorithme 1 : Initialisation RC4

Soit une clé **K** de longueur : **Longueur_clé**

S[] , **T[]** de taille **256**

pour **i** de **0** à **255**

S[i] := i;

T[i] := K[i mod longueur_clé];

finpour

- **Permutation**

Le vecteur temporaire T[] est ensuite utilisé pour produire la permutation initiale de S. Pour chaque cellule S[i] de S, celle-ci sera échangée avec une autre cellule de S selon un calcul basé sur la valeur comprise dans la cellule T[i] correspondante. L'Algorithme 2 illustre cette phase :

Algorithme 2 ; Permutation RC4

j := 0;

pour **i** de **0** à **255**

j := (j + S[i] + T[i]) mod 256;

Permuter(S[i], S[j]);

finpour

- **Génération du flux de chiffrement**

À partir de cette étape, la clé d'entrée n'est plus utilisée. Pour chaque S[i], on procèdera à un échange avec un autre octet de S, selon un schéma basé sur la configuration courante de S.

Une fois arrivé à S[255], le processus redémarre à la cellule S[0] (on parle de PRGA pour Pseudo-Random Generation Algorithm). La valeur de k est alors utilisée pour le chiffrement par une opération XOR avec le prochain octet de texte clair, ou pour le déchiffrement par une opération XOR avec le prochain octet de texte chiffré.
L'algorithme 3 illustre cette phase :

Algorithme 3 : Chiffrement RC4

i := 0; j := 0 ;

tant_que (**Vrai**) // il y a des Octets à chiffrer

i := (i + 1) mod 256;

j := (j + S[i]) mod 256;

Permuter(S[i], S[j]) ; // encore une dernière permutation

Octet_chiffrement = S[(S[i] + S[j]) mod 256] ;

result_chiffré = Octet_chiffrement XOR Octet_message;

fintant_que

b. Robustesse du RC4

Dans RC4, le chiffrement d'un message en clair M, en utilisant une clé K, produit un message chiffré C, selon la formule : C=M **XOR** Suite-Octets-Pseudo-Aléatoire.
La faiblesse d'une telle formule réside dans le fait que l'utilisation de la même clé K dans le chiffrement de plusieurs messages peut révéler certaines informations sur les messages chiffrés.
Supposons l'échange sécurisé de deux messages M1 et M2 en utilisant le chiffrement RC4, on aura les deux formules suivantes :

chiffrement du premier message C1=RC4(M1,K) = M1 XOR SOPA(k)

chiffrement du deuxième message C2=RC4(M2,K) = M2 XOR SOPA(k)

On peut facilement remarquer qu'un attaquant disposant de C1 et C2 peut retirer les parties pseudo aléatoires de C1 et C2 par une simple opération XOR comme suit :

C1XORC2= M1 XOR SOPA(k)XOR M2 XOR SOPA(k)=M1XORM2.

Cela vient du fait que la sécurité repose exclusivement sur la clé. En effet, c'est uniquement cette dernière qui détermine le flux de sortie du générateur.

En 2001, Fluhrer, Mantin et Shamir [**FLU 2001**] remarquèrent que les **premiers octets** en sortie du RC4 n'étaient pas tout à fait aléatoires. Comme conséquence, un attaquant peut tirer des informations importantes sur la clé d'initialisation depuis ses premiers Octets.

2.2 DES (Data Encryption Standard), un système de chiffrement par blocs

Ce système de chiffrement à clé privée par blocs est le plus connu. Il a été adopté comme standard américain en 1977 (standard FIPS 46[**FIP 1999**]) pour les communications commerciales, puis en 1991 par l'ANSI (American National Standards Institute). DES est basé sur les structures de Feistel [**FEI 1973**]

a) Algorithme général

Ce système fonctionne par blocs de 64 bits en utilisant une clé de 56 bits(la clé compte 64 bits dont 8 bits pour le contrôle de parité). A partir de la clé initiale on génère 16 sous clés de taille 48 bits chacune.

L'algorithme compte trois étapes :

Entrée : Un bloc de 64 bits ***p*** ; Une clé de 64 bits **K** ;

1- On applique sur ***p,*** une permutation initiale **IP()** , prédéfinie, pour obtenir une chaine initiale notée $\mathbf{P_0}$. On aura :

$\mathbf{P_0=IP(}\boldsymbol{p}\mathbf{)=L_0.R_0}$ // L_0 sont les 32 premiers bits de P_0 et R_0 les 32 bits restants ;

2- Appliquer seize itérations (ou tours) d'une certaine fonction ***f***, suivant les règles :

Pour 1<=i<=16 **faire**

$L_i=R_{i-1}$

$R_i=L_{i-1} \oplus f(R_{i-1},K_i)$ // les K_i sont obtenues à partir de la clé initiale K

FinPour

3- L'inverse de IP (noté IP^{-1}) est appliqué au dernier bloc de la dernière itération (tour) $R_{16}||L_{16}$ pour obtenir le texte chiffré **C** ;

$C=IP^{-1}(R_{16}.L_{16})$ // noter l'inversement dans l'ordre de L_{16} et de R_{16} au dernier tour.

L'algorithme est illustré par le schéma suivant :

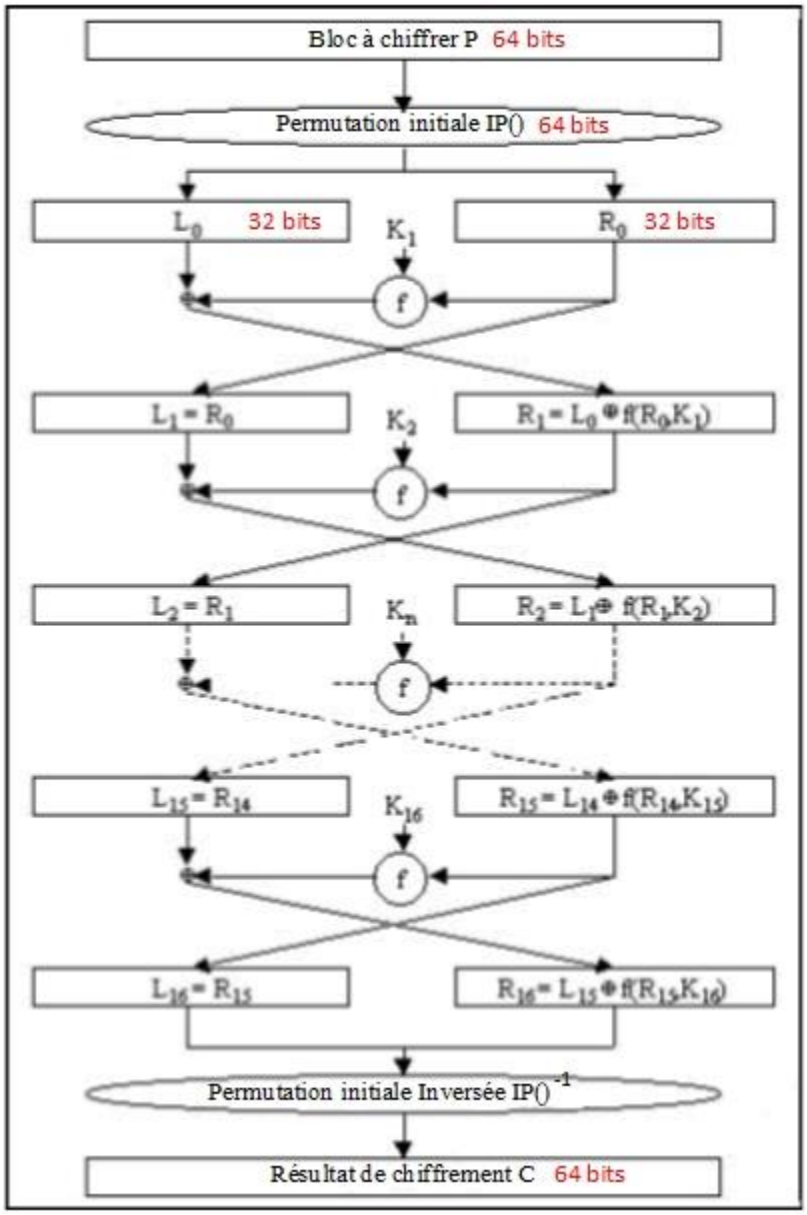

Figure 4: Schéma de chiffrement DES

Dans ce qui suit, on détaille les différentes fonctions utilisées dans l'algorithme.

- **Les permutations IP et IP^{-1}**

Elles opèrent sur l'ordre des bits du bloc comme suit :

Permutation initiale

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Permutation initiale inverse

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Figure 5: Fonction IP() et $IP()^{-1}$

Les tableaux se lisent de gauche à droite et de haut en bas. Cela signifie que dans le calcul de IP(*p*), le bit 58 de du bloc *p* devient le premier, le bit 50 devient le deuxième …etc.
IP^{-1} est la matrice inverse de IP.

- **fonction *f***

Au départ, l'argument de gauche à 32 bits est expansé en 48 bits en redoublant certains bits (fonction E() de la figure7). Ensuite, on applique un XOR de cet argument expansé avec le deuxième argument qui est la sous clé K_i (48 bits). Le résultat possède 48=8*6 bits est transformé en une chaine de 32 =8*4 bits en utilisant des dispositifs appelés boîtes-S (Figure 8) qui calculent un bloc de 4 bits à partir d'un bloc de 6 bits. Enfin, on applique la permutation finale P, décrite par la figure7 à ces 32 bits pour obtenir le résultat final de ***f*** sur 32 bits.

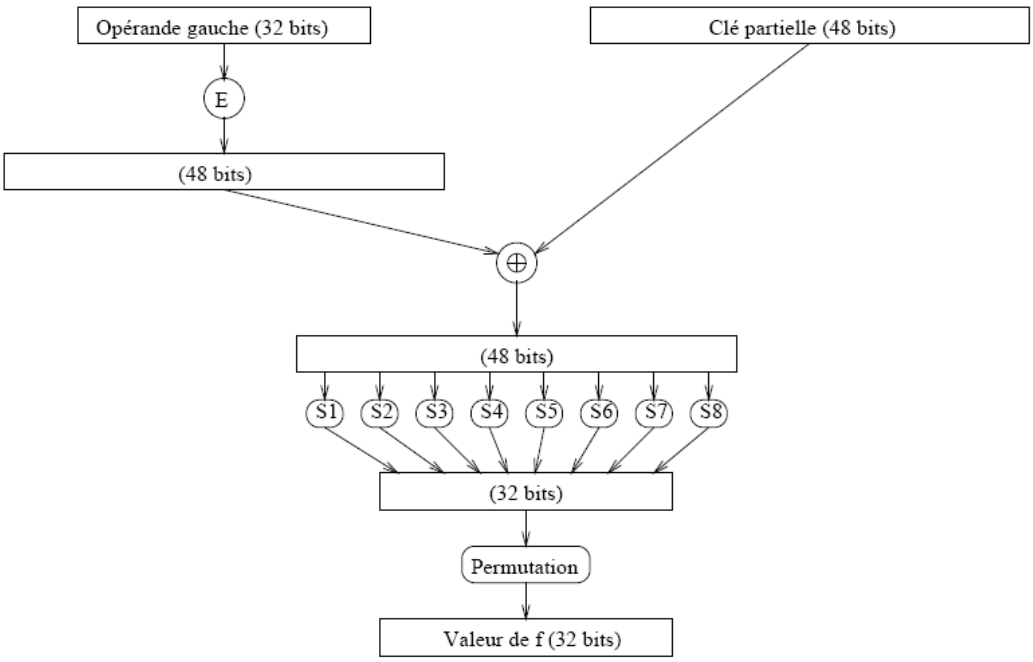

Figure 6: Fonction f

Fonction E d'expansion

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Permutation P finale

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Figure 7: Expansion E et Permutation P

- **Boites-S**

Ce sont des tableaux à 2 lignes et 16 colonnes. Les premiers et derniers bits de l'entrée déterminent une ligne du tableau, les autres bits déterminent une colonne. La valeur numérique trouvée à cet endroit indique la valeur des quatre bits de sortie.

S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figure 8: Boites-S

➢ **Génération des sous clés K_i (diversification)**

Le principe de diversification de la clé initiale K est schématisé par la figure 9.

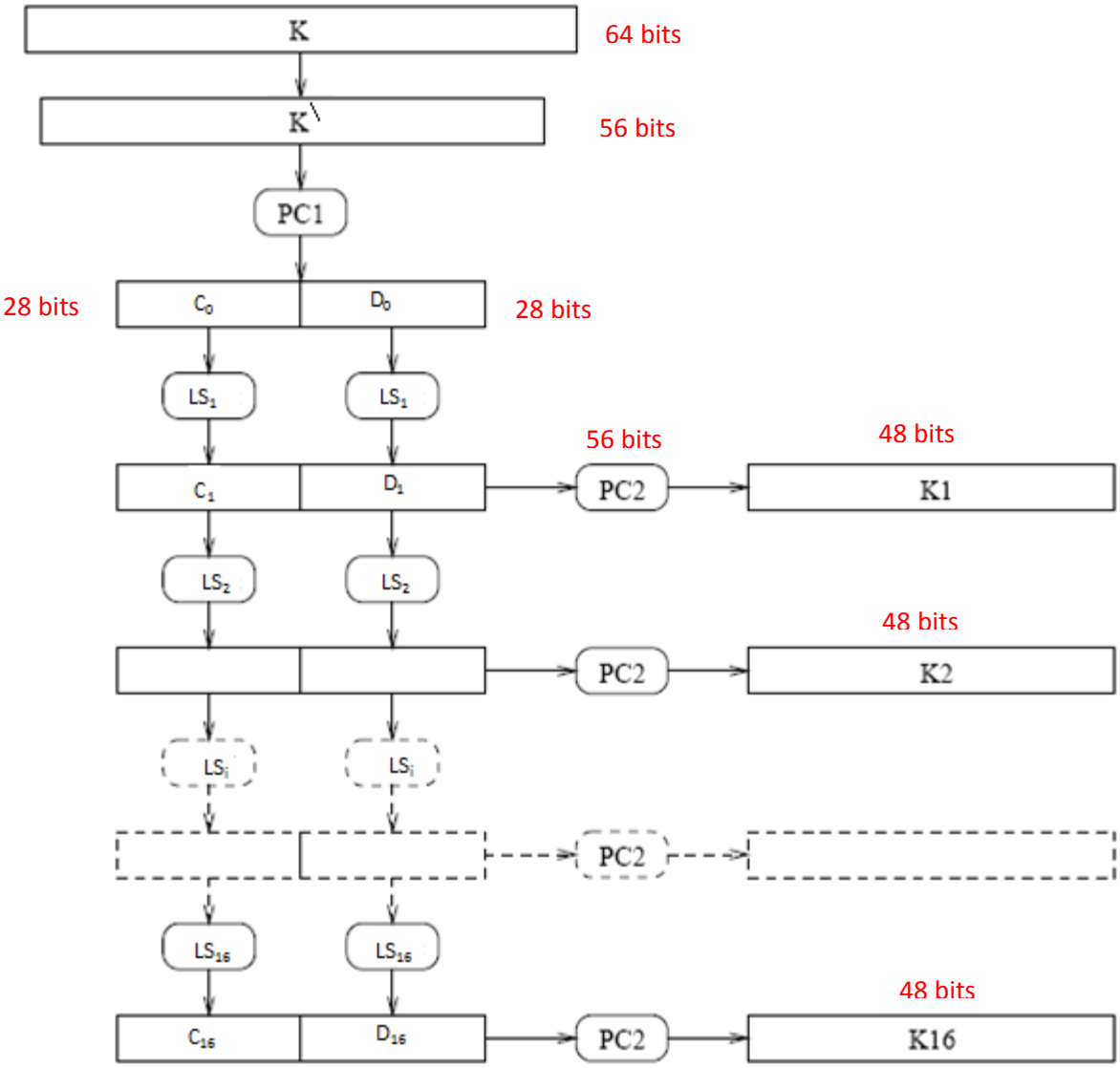

Figure 9: Diversification de la Clé K

On applique d'abord une permutation PC1 à K. puis, à chacune des 16 étapes, chaque moitié de la chaine de 56 bits obtenue subit une rotation à gauche, d'un cran aux étapes 1,2,9,16 et de deux crans aux autres étapes. Après chacune de ces étapes, on obtient une clé partielle de 48 bits en appliquant la règle d'extraction PC2. La permutation PC1 et la règle PC2 sont

détaillées par la figure 12. Les 56 bits de K y sont numérotés de 1 à 64 en évitant les multiples de 8, puisque dans la pratique, ces positions sont des bits de parité.

Permutation PC_1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Règle d'extraction PC_2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Figure 10: PC1 et PC2

b) Déchiffrement en DES

Il suffit de prendre Le même **schéma de chiffrement** avec l'ordre Inversé des clés.

Le schéma de déchiffrement est illustré par la figure11.

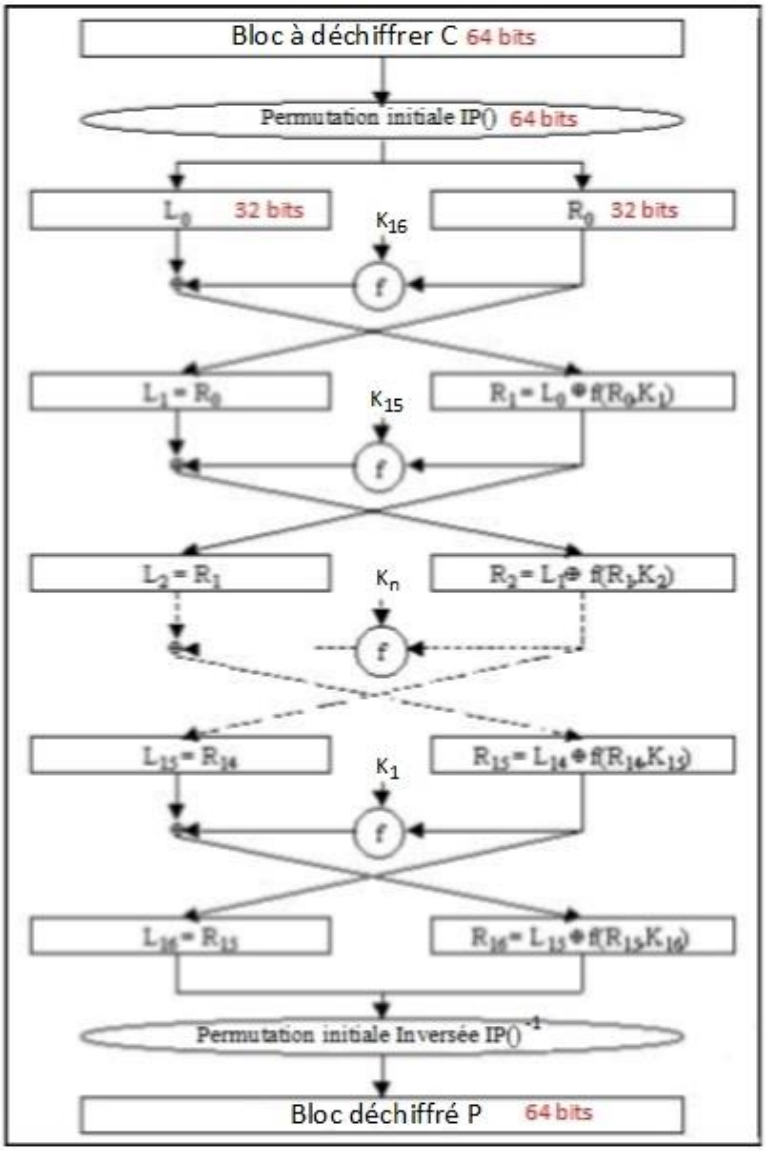

Figure 11: Déchiffrement DES

c) Evolution de DES

Avec l'évolution rapide des ordinateurs, les clés de 56 bits sont devenues vulnérables aux attaques exhaustives. Il a été Remplacé par le Triple DES à deux clés (112 bits), le 3DES est défini par la formule : $DES_{K1}(DES^{-1}_{K2}(DES_{K1}(M)))$.

Actuellement remplacé par l'AES (Advanced Encryption Standard) qui utilise une Clé de longueur modifiable pour assurer une résistance suffisante à l'attaque exhaustive.

3 Chiffrement à clé publique (Asymétrique)

Dans un système de chiffrement asymétrique, chaque entité possède deux clés distinctes, une nommée publique et l'autre privée (secrète). La clé privée est utilisée exclusivement pour le déchiffrement, elle réside chez son propriétaire et elle n'est jamais communiquée. La clé publique est utilisée exclusivement pour le chiffrement, elle est communiquée à toutes les entités désirant l'utiliser pour chiffrer des données à destination de son propriétaire. La clé publique est distribuée librement du fait qu'il est impossible de déduire la clé privée à partir de la clé publique.

Le schéma général d'un chiffrement asymétrique est illustré par la figure 12. La clé publique est noté P_k, la clé secrète est noté S_k.

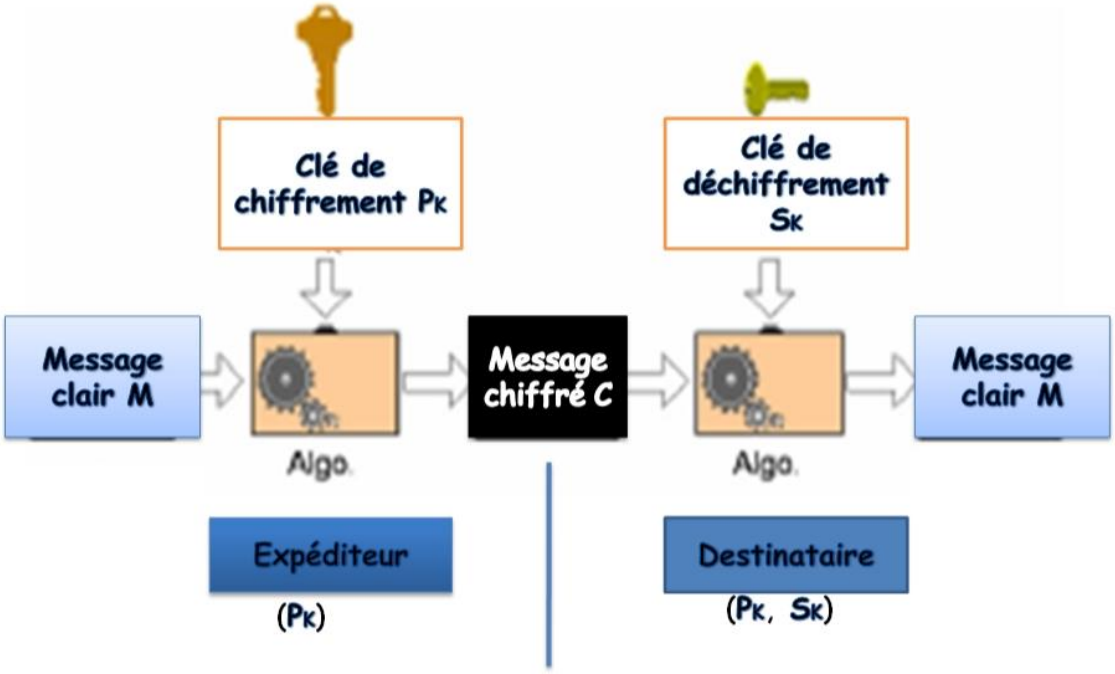

Figure 12: Chiffrement à clé publique

3.1 Propriétés du chiffrement asymétrique

- Clé de chiffrement ≠Clé de déchiffrement.
- Pas besoin de communiquer les clés privées (clés de déchiffrement).
- Opérations Chiffrement/Déchiffrement lentes .
- Besoin d'une infrastructure pour la gestion des clés publiques

Le concept de clé publique date officiellement de 1976 de Diffie et Hellman [**DIF 1976**]. La première implémentation a eu lieu en 1978 par Rivest, Shamir et Adleman [**RIV 1978]** sous la forme de l'algorithme RSA.

La sécurité de tels systèmes repose sur des problèmes calculatoires :

- RSA : factorisation de grands entiers.
- ElGamal : logarithme discret [ELG 1985].
- Merkle-Hellman : problème du sac à dos (knapsacks) [MER 1978].

3.2 RSA : Rivest - Shamir - Adleman

Il s'agit du système de chiffrement asymétrique le plus utilisé, Il exploite le problème de factorisation. Sa sécurité repose sur le principe suivant :

Le calcul du produit de deux nombres premiers est aisé. La factorisation d'un nombre en ses deux facteurs premiers est beaucoup plus complexe.

Avant de détailler RSA, on donne un rappel de quelques notions mathématiques qui sont nécessaires à la compréhension de l'algorithme.

a. Complément mathématique

Dans ce cours, l'ensemble des relatifs est symbolisé par $\mathbb{Z}$; pour un entier naturel *n, l'ensemble des entiers positifs est noté* $\mathbb{Z}_n$*={0,1,2,3..n-1}.*

Définition 1. Soient **a** et **b** deux entiers relatifs ; on dit que **b** divise **a** et on note **b|a** s'il existe un entier **c** tel que **a = c.b**

Définition 2. Un entier positif est premier s'il n'admet pas d'autres diviseurs que 1 et lui-même. 1 n'est pas considéré comme premier.

Exemple : 2,3,5 et 7 sont des nombres premiers.

Théorème 1. (Décomposition en produit de facteurs premiers)
Tout nombre entier naturel s'écrit comme un produit de nombres premiers ; cette décomposition en facteurs premiers est unique (exceptée par réarrangement des facteurs).

Exemple : $\mathbf{1\,200 = 2^4 \times 3 \times 5^2}$ et il n'admet aucune autre factorisation sous forme de produits de nombres premiers, excepté par réarrangement des facteurs 2,3 et 5.

La décomposition en facteurs premiers est aisée pour de petits nombres, mais elle est très longue pour de grands nombres ; La sécurité du système asymétrique RSA est basée sur la **difficulté de la factorisation de grands nombres.**

Définition 3. *(PGCD)*
Le PCGD de deux entiers naturels **a** et **b** est le plus grand élément de l'ensemble de leurs diviseurs communs ; on le note PGCD(a,b).

L'Algorithme 3 permet le calcul du PGCD de deux entiers.

Algorithme 3 (EUCLIDE).

Soient **a** et **b** deux entiers naturels avec **a ≥ b**

Tant que b ≠ 0 *faire*
r ←a mod b ; a←b ; b← r.
FinTantque
*retourne(***a***)*

Si PGCD(a,b) = 1 alors **a** et **b** sont premiers entre eux.

Le PGCD de deux entiers relatifs est celui de leurs valeurs absolues.

Exemple : PGCD(26,3)=1

Théorème 2 *(Identité de BACHET-BÉZOUT).*
*Soient **a** et **b** deux entiers non nuls. Il existe deux entiers relatifs **u** et **v** tels que **: au + bv = PGCD(a,b)**.*

L'algorithme 3 peut être étendu pour fournir, outre le PGCD de **a** et **b**, deux entiers premiers entre eux **u** et **v** tels que **au + bv = PGCD(a,b)** (***Théorème 2***).

Algorithme 4 (EUCLIDE étendu) : Soient **a** et **b** deux entiers tels que **a ≥ b ≥ 1**.

Entrée : **a**, **b** entiers (naturels)
Sortie : **r** entier (naturel) et **u**, **v** entiers relatifs tels que **r = pgcd(a, b)** et **r = a*u+b*v**

Initialisation : **r := a**, **r' := b**, **u := 1**, **v := 0**, **u' := 0**, **v' := 1**
q, **rs**, **us**, **vs** quotient et variables de stockage intermédiaires

les égalités **r = a*u+b*v** et **r' = a*u'+b*v'** sont des invariants de boucle.

Tant que (**r' ≠ 0**) *faire*
q := r÷r'
rs := r, us := u, vs := v
r := r', u := u', v := v'
r' := rs -q *r', u' = us - q*u', v' = vs -q*v'
FinTantque
retourne (**r**, **u**, **v**)

Certains éléments de $\mathbb{Z}_n$ n'ont pas d'inverses ; il existe même des éléments non nuls dont le produit est nul.

Théorème 3. Conditions d'invisibilité dans $\mathbb{Z}_n$:
– k est inversible dans $\mathbb{Z}_n$ implique PGCD(k ,n) = 1.
– Si PGCD(k ,n) ≠ 1 il existe p dans $\mathbb{Z}_n$ tel que k × p = 0.
– Si n est un nombre premier, tous les éléments de $\mathbb{Z}_n$ sont inversibles sauf 0.
La détermination de l'inverse de k dans $\mathbb{Z}_n$ s'effectue grâce à l'algorithme d'EUCLIDE étendu.

On aura dans ce cas-là : PGCD(k ,n)=u×k+v×n (bézout), et on lit : **u** inverse de **k** modulo **n**

Exemple : calculer l'inverse de 3 en $\mathbb{Z}_{26}$

On met a=K et n=b, le résultat de l'application d'Euclide Etendu est présenté dans le tableau 7.

Tableau 7: calcul de l'inverse par Euclide Etendu

r									**=**	**u**	**×**	**a**	**+**	**v**	**×**	**b**
26									=	1	×	26	+	0	×	3
3									=	0	×	26	+	1	×	3
2	=	26	-	8	×	3			=	1	×	26	+	-8	×	3
1	=	3	-	1	×	**2**	=	1×3-1×(1×26 - 8×3)	=	-1	×	26	+	9	×	3

On aura donc, PGCD(a,b)=au + bv
a=3, b=26, u=9, v=-1
On lit : 9 inverse de 3 en $\mathbb{Z}_{26}$

Définition 4. (Fonction totient d'Euler)
La fonction d'Euler définie par : **φ(n) =#{m≤n|PGCD(m,n) =1}**

φ(n) est l'indicateur d'Euler, c'est-à-dire le cardinal des entiers inversibles de **n**.
Il faut lire : la fonction **φ** du nombre *n* a pour résultat un nombre **m** inférieur ou égal à **n** et tel que le plus grand commun diviseur de **n** et **m** soit 1.

- Si **n** est premier, il vient **φ(n) =n−1**. La fonction φ(n) donne le nombre d'entiers positifs plus petits ou égaux à n relativement premiers à n.
- De plus, soient **p** et **q** deux nombres premiers et **n=pq**. Il vient

$$\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1)$$

b. **Principe du RSA**

Dans ce qui suit, on utilisera de petits nombres pour faciliter les calculs. En pratique, RSA manipule des nombres qui ont au moins 100 chiffres décimaux.

Soit **p** et **q** deux nombres premiers dont le produit est **n=p.q**.

2.2.1 On calcule φ(n) tel que φ(n) = (p-1) * (q-1).

2.2.2 Puis on choisit un nombre e tel que e soit premier avec φ(n).

2.2.3 On cherche ensuite le nombre **d** qui est l'inverse de **e** modulo **φ(n)** (e.d≡1mod(Φ(n))).

Le couple **(e,n)** représente la clé publique.

Le couple **(d,n)** représente la clé privée.

Le chiffrement se fait selon la formule : $\mathbf{C = M^e \bmod n}$ et le déchiffrement par $\mathbf{M = C^d \bmod n}$

Exemple

Soient p = 31, q = 53

n=1643.

Φ(n) = 1560.

Soit e = 11 (PGCD(e,Φ(n))=1).

On détermine **d** par l'application d'Euclide Etendu (inverse modulaire de **e** sur $Z_{\Phi(n)}$), on trouve **d=851**.

La clé publique **:(11,1643)**

La clé privée est: **(851,1643).**

Chiffrement du mot « **ONE** »

1. D'abord on produit un codage par la position dans l'alphabet du mot **ONE** (a=01,b=02,c=03,..z=25), on obtient : **15 14 05**

2. Découpage en morceaux de même taille 3 : **151 405**

3. Chiffrement des blocs sur quatre positions :

151^{11} mod 1643 =1453

405^{11} mod 1643=0374

Résultat de chiffrement : **14530374**

Déchiffrement de **14530374**

1. Découpage en morceaux de même taille 4 : **1453 0374**

2. Déchiffrement des blocs sur trois positions :

1453^{851}mod 1643 =151

0374^{851} mod 1643=405

Résultat de déchiffrement : **151 405**

3. Découpage en deux positions pour la reconstitution du mot : **15 14 05=ONE**

On note que lors de l'utilisation de RSA, en plus de la clé privée, il faut connaitre le processus de découpage utilisé.

c. Conseils d'utilisation du RSA

Pour garantir une bonne sécurité, il faut respecter certains règles telles que :

– Ne jamais utiliser de valeurs **n** trop petites,

– N'utiliser que des clés fortes (p-1 et q-1 ont un grand facteur premier),

– Ne pas chiffrer de blocs trop courts,

– Ne pas utiliser de n communs à plusieurs clés,

– Si (d,n) est compromise ne plus utiliser n.

d. Exponentiation rapide

Plusieurs algorithmes ont été proposés dans la littérature afin de faciliter les calculs d'exponentiation. On présente ici l'algorithme ***square and multiply.***

Algorithme square and multiply (bits forts → bits faibles)
Entrée *: x et n*
Sortie *: x^n*
y=1, n=$\sum_0^k b_i 2^i$ (écriture binaire de n)
pour *i=k,……,0* ***faire***
y=y^2
si *b_i=1* ***alors***
y=y.x
FinPour
Retourne *y*

Exemple **:** calculer 9^{107} mod 187

On commence par écrire la puissance en binaire : $107=(1101011)_2$

Après on déroule l'algorithme comme illustré dans le Tableau 8.

Tableau 8: déroulement de l'algorithme square and multiply

i	b_i	Etapes du calcul
6	1	$1^2 \times 9$
5	1	$9^2 \times 9 = 729 = 168 \mod 187$
4	0	$168^2 = 28224 = 174 \mod 187$
3	1	$174^2 \times 9 = 272484 = 25 \mod 187$
2	0	$25^2 = 625 \mod 187$
1	1	$64^2 \times 9 = 36864 = 25 \mod 187$
0	1	$25^2 \times 9 = 5635 = 15 \mod 187$

Le résultat final est celui de la dernière ligne du tableau : 9^{107} mod 187=**15**.

Chapitre IV : Signature numérique et fonctions de hachage

1. Introduction

Les fonctions de hachage et la signature numérique assurent les objectifs de sécurité les plus importants. Elles sont utilisées dans le cadre du contrôle de l'intégrité de données et dans le cadre du contrôle de l'authentification et de la non-répudiation.
Ce chapitre introduit les bases du hachage et de la signature numérique, en introduisant leur principe de fonctionnement et leur utilisation en pratique.

2. Fonctions de hachage

Une fonction de hachage produit des suites binaires de tailles fixes à partir de chaines de taille quelconques.

2.1 Utilisation des fonctions de hachage

Les fonctions de hachage sont utilisées dans le cadre du contrôle de l'intégrité de données et dans le cadre de la signature numérique.

- ✓ L'intégrité est assurée dans la mesure où les fonctions de hachage permettent d'obtenir facilement une empreinte (haché, condensé, résumé) d'une donnée (fichier, message…), cette empreinte est utilisée comme l'identification unique de cette donnée pour contrôler d'éventuelles altérations ou modifications lors de la transmission.
- ✓ Dans la signature numérique, et pour des raisons d'efficacité, on signe l'empreinte de la donnée à transmettre et pas la donnée elle-même.

2.2 Notions sur les fonctions de hachage

Les fonctions de hachage manipulent des informations binaires dans l'ensemble {0,1}.

a. ***Définition***

Une fonction de hachage est une application h : $\{0,1\}^*$ ----> $\{0,1\}^n$ $n \epsilon N$

En clair, une fonction h associe à des suites binaires de longueurs arbitraires des suites binaires de longueurs fixées.

Exemple :

L'application qui associe b_1 xor b_2 xor b_3 …….. xor b_k au mot binaire $b_1 b_2$ b_3 ……..b_k dans $\{0,1\}^*$ est une fonction de hachage. Pour une suite b, elle renvoie 1 si le nombre de 1 dans b est impair et 0 sinon.

Exemple : pour la suite 01101 on a bien **h**(10100)=0.

En pratique, les fonctions de hachage sont produites à partir de fonctions de compression à sens unique. Dans ce qui suit on donne les définitions correspondant à ces deux types de fonctions.

b. Fonction de compression

Une fonction de compression est une application, **$g : \Sigma^m$ ----> Σ^n m,nϵN et m>n**

Autrement dit, une fonction de compression produit une suite binaire de longueur fixée à partir d'une autre suite de longueur fixée plus courte.

c. Fonction à sens unique

C'est une fonction relativement facile à calculer mais considérée plus difficile à inverser. En d'autres termes, étant donné un x, il est facile de calculer h(x), mais étant donné h(x) il est très difficile de calculer x. Le mot difficile, veut dire qu'il faudrait beaucoup de temps en terme de calcul processeur, et beaucoup d'espace mémoire pour arriver à calculer l'inverse de h(x).

d. Collision dans une fonction de hachage

Une fonction de hachage n'est jamais injective, de ce fait, une collision se présente lorsque pour deux entrées différentes **x** et **x'** , on a : **h(x)=h(x')**

Exemple : une collision dans la fonction XOR de l'exemple précédent est un couple formé de deux chaines de bits distinctes, ayant toutes les deux un nombre pair de 1 :

x=01101, **x'=**01101, on obtient **h**(01101)=1 et **h**(10100)=1

2.3 Propriétés des fonctions de hachages

Soit **h** une fonction de hachage, on note **y=h(x)**, alors **x** est appelé ***préimage*** de **y**.
Les propriétés à vérifier dans une fonction de hachage **h** à utiliser pour la cryptographie sont :

1-Propriété de *compression* et de facilité de calcul ;

2-*Résistance à la préimage* : étant donné y, il est ***calculatoirement*** difficile de trouver un x tel que y=h(x) (fonction à sens unique) ;

3-*Résistance à la deuxième préimage* : étant donné y et un x (spécifié) tel que y=h(x), il est ***calculatoirement*** difficile de trouver un autre x' ≠x tel que y=h(x') ;

4- *Résistance à la collision* : il est ***calculatoirement*** difficile de trouver x et x' tel que: x ≠x' et h(x)=h(x').

2.4 Construction d'une fonction de hachage

Merkel et *Damgard* [DAM 1989], proposent de construire une fonction de hachage à base d'une fonction de compression. Ils assurent aussi que si la fonction de compression est résistante aux collisions alors il est de même pour la fonction de hachage résultante. C'est sur ce principe que sont basées les fonctions de hachage les plus utilisées dans la pratique comme **MD5** et **SHA**.

Dans ce qui suit nous utilisons un cryptosystème qui manipule des informations binaires. Le résultat du hachage est dans l'ensemble $\{0,1\}^n$. On prend n>=128, pour éviter l'attaque des anniversaires qu'on va décrire après.

e. On dispose d'une fonction de compression **g : $\{0,1\}^m$----> $\{0,1\}^n$** pour n∈N et m-n>1
f. On va utiliser la fonction **g** pour en définir une fonction de hachage **h()**
   h : $\{0,1\}^*$----> $\{0,1\}^n$ pour n∈N

Etapes de construction :

1-Normalisation du message à hacher :

- Soit **Mϵ{0,1}*** le message à hacher. **M** est de longueur arbitraire **L** (|M|=L).
- Soit **r=m-n.**
- Si le nombre de bits du message **M** n'est pas multiple de **r**, on rajoute des **0** au début du message **M** pour qu'il le soit ; on note **u=0^i.M ,**tel que **|u|≡0 mod r** .
- Si le nombre de bits de **L** n'est pas multiple de **r-1**, on rajoute des **0** en début de **L**. on note **y= 0^i.L**, tel que **|y|≡0 mod r-1.**
- Découper le mot **y** en blocs de taille **r-1** chacun.
- Rajouter un **1** au début de chacun des blocs obtenu précédemment ; le mot **y** devient un mot de blocs de taille **r** chacun, le nouveau mot obtenu est noté **y'**.
- ✓ En fin, concaténer **u** avec **r** zéro avec **y'** pour obtenir le message normalisé **M'= u0^{r}y' ;** on obtient un message de **k** blocs **M'= M'$_1$ M'$_2$ M'$_3$….. M'$_k$,** chacun des blocs est de taille **r** (**M'$_i\epsilon${0,1}^{r} 1≤ i ≤k)**

Exemple:

Soit à normaliser le message M=11101 avec une définition g : {0,1}^{132}----> {0,1}^{128}

r=132-128=4, M=11101

L=5=101

u=0^3.M=0001 1101

y=101 ; y'=1.y=1101

M'=u.0^4.y'=0001 1101 <u>0000</u> 1101 (M'$_1$ =0001, M'$_2$ =1101, M'$_3$ =<u>0000</u>, M'$_4$=1101)

2-Hachage :

Le hachage de **M** est obtenu en appliquant le schéma dans la figure suivante sur **M'**, avec un vecteur d'initialisation **IV=0^n** (**n** bits de valeurs **0**).

h$_0$=IV ; h$_i$=g(h$_{i-1}$M'$_i$) 1≤ i ≤k

h(M)=h$_k$ (résultat du hachage).

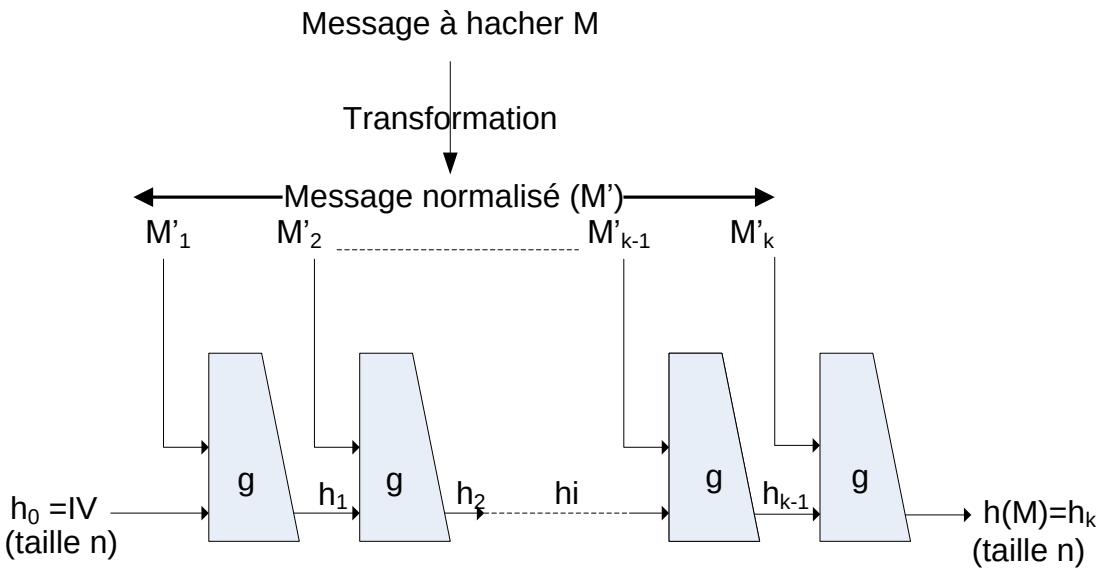

Figure 13: Etapes de hachage du message M

IV(Initial Value) est une chaine de taille fixée(n) mais publique. Merkle-Damgard assurent que si **g** est résistante aux collisions, il en est de même pour **h**.

2.5 Attaque sur les fonctions de hachage

Les fonctions de hachage sont attaquées en utilisant une technique appelée "Paradoxe des anniversaires" [SHE 2002]. Ce paradoxe répond à la question suivante : combien de personnes faut-il dans un groupe pour que la probabilité d'avoir deux personnes ayant la même date de naissance atteigne 0.5 ?

Définition du problème

Pour ce problème, il faut ignorer le 29 février et les années bissextiles. Définissons P(n,k)=Pr[il y a au moins deux éléments communs dans le sous-ensemble de k éléments, et la répartition est uniforme dans l'intervalle 1 à n].

On cherche la valeur de **k** telle que **P(365, k) >= 0, 5.**

Dans un premier temps, on cherche la probabilité qu'il n'y ait pas d'éléments communs, ce que nous noterons **Q(365, k)**. Pour ce faire, on note par N, le nombre de manières de tirer k valeurs sans élément commun (arrangement A^k_n). Il vient N =A^k_{365}=365!/(365 − k)!

Si on enlève la restriction des doublons, il y aura 365^k possibilités. Donc, la probabilité de ne pas avoir de doublons est donnée par : $Q(365, k) = N/365^k = 365!/(365 - k)!(365)^k$

Par conséquent, $P(365, k) = 1 - Q(365, k) = 1 - 365!(365 - k)!(365)^k$

Pour **k = 23**, on a **P(365, k) = 0.5073**.

Projection du paradoxe des anniversaires sur les fonctions de hachage :

Soit une fonction de hachage **h : {0,1}*----> {0,1}^{n}** pour nϵN. h peut générer 2^n hachés différents. K le nombre de chaines à choisir pour avoir une probabilité supérieur à 0,5 de trouver une collision.

En appliquant le Paradoxe des anniversaires on obtient $k=2^{n/2}$ chaines. Donc, en calculant un peu plus que $2^{n/2}$ valeurs hachées, l'attaque des anniversaires trouve une collision avec une probabilité supérieure à 0.5.

Pour une protection contre l'attaque des anniversaires, n doit être choisi de façon que le calcul de $2^{n/2}$ valeurs hachées soit ***calculatoirement*** impossible.

3. Signature numérique

La signature numérique est utilisée pour prouver l'identité du signataire, lier une donnée à son auteur et garantir la non-répudiation.

La signature numérique est caractérisée par les propriétés suivantes :

1-Elle assure l'Authentification : elle permet de convaincre le destinataire de l'identité de l'auteur du document.

2-Elle assure la Non répudiation : elle est la preuve que le signataire a généré ou bien a accepté le document.

3-Un document signé est inaltérable : une fois le document signé, il ne peut être modifié.

4-Une signature n'est pas réutilisable : elle fait partie du document signé et ne peut être déplacée sur un autre document.

3.1 Utilisation de la signature numérique

Une des propriétés les plus importantes dans un système de chiffrement à clé publique est la possibilité de signer avec la clé secrète et de pouvoir vérifier cette signature en utilisant la clé publique correspondante.

La signature numérique peut être mise en œuvre par un système de chiffrement à clé publique combiné avec des fonctions de hachage. Le principe général de la signature d'un document, en combinant ces deux objets, est illustré dans le schéma de la figure 14.

Le principe est illustré à travers un échange sécurisé d'un message M entre deux communicants (Alice et Bob). Initialement, chaque entité doit disposer d'un couple de clés publique et privée (P_k, S_k). Après, on passe à l'échange des clés publiques entre les entités (étape 2). La signature est appliquée sur le haché du message pour plus d'efficacité (étape 5). En effet, un temps précieux serait perdu dans la signature du message complet M.

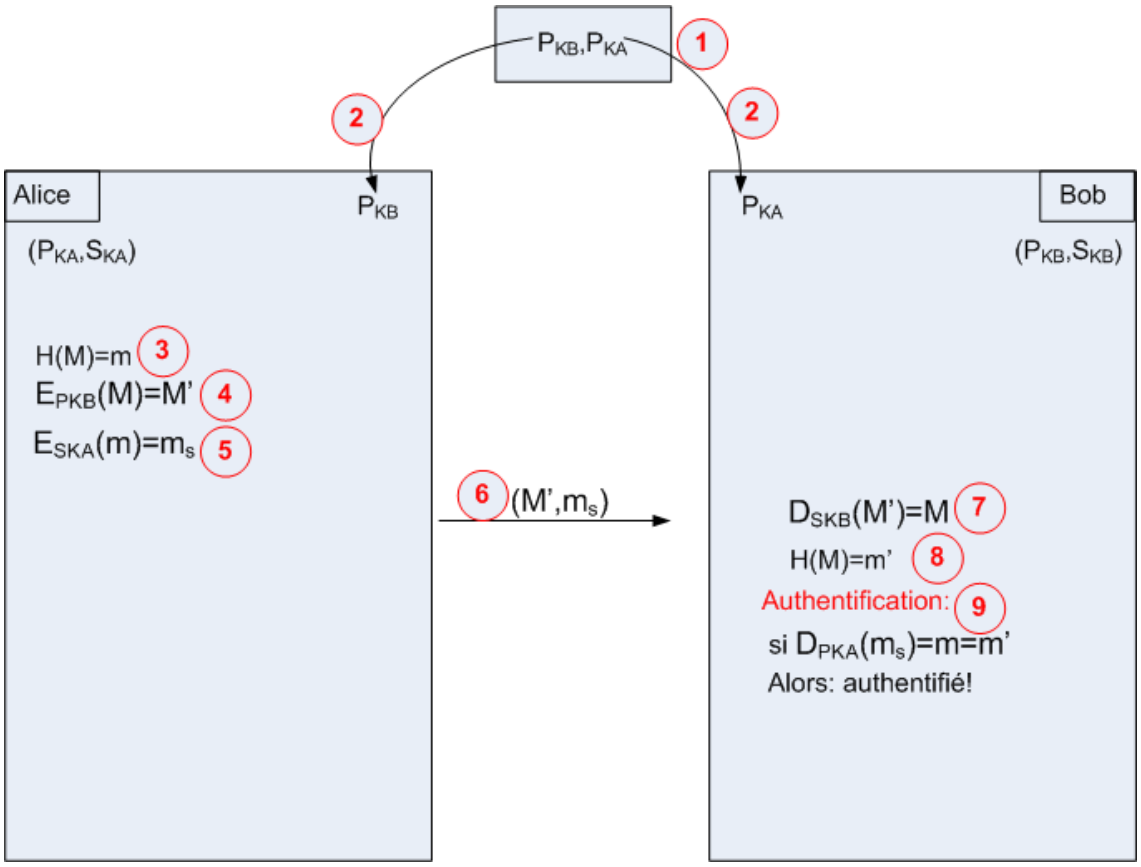

Figure 14: Principe de la signature numérique

Une fois le message chiffré, Alice envoie le couple message chiffré et signature du message à Bob (étape 6). Comme la clé privée d'Alice est utilisée pour générer la signature, on garantit que seule Alice (qui possède la clé privée) a pu signer le message M. Réciproquement, c'est sa clé publique (connue par Bob) qui permet la vérification de la signature d'Alice (étape 9).

3.2 Mise en œuvre de la signature RSA

Le procédé de génération des paramètres est identique à celui utilisé pour l'algorithme de cryptage RSA (Chapitre 3, section 1.2). Alice disposera donc d'une clé publique (e,n) et d'une autre clé secrète (d,n).

Un message à signer **m** est un entier naturel dans $\mathbb{Z}_n$. La signature de **m** par Alice est simplement : $\mathbf{s=m^d \bmod n}$

Pour vérifier la signature d'Alice, Bob récupère le message **m** et la signature **s** et utilisera la clé publique d'Alice pour vérifier l'identité suivante :

> **Si** $m=s^e \bmod n$ alors **signature valide**
> **Sinon**, signature **non valide.**

Chapitre V : La Gestion de clés

1. Introduction

La sécurité des systèmes de chiffrement actuels, à clé publique ou secrète, repose principalement sur le secret de la clé. Cela constitue un principe dans la sécurité informatique.

Les techniques de gestion de clés permettent d'assurer une bonne gestion des clés cryptographiques, depuis la phase de génération (choix des clés) jusqu'à la phase d'expiration.

2. Cycle de vie des clés de chiffrement

La gestion des clés doit permettre la génération sûre, l'activation, la distribution et la destruction pour assurer le cycle de vie des clés qui est décrit dans la figure suivante :

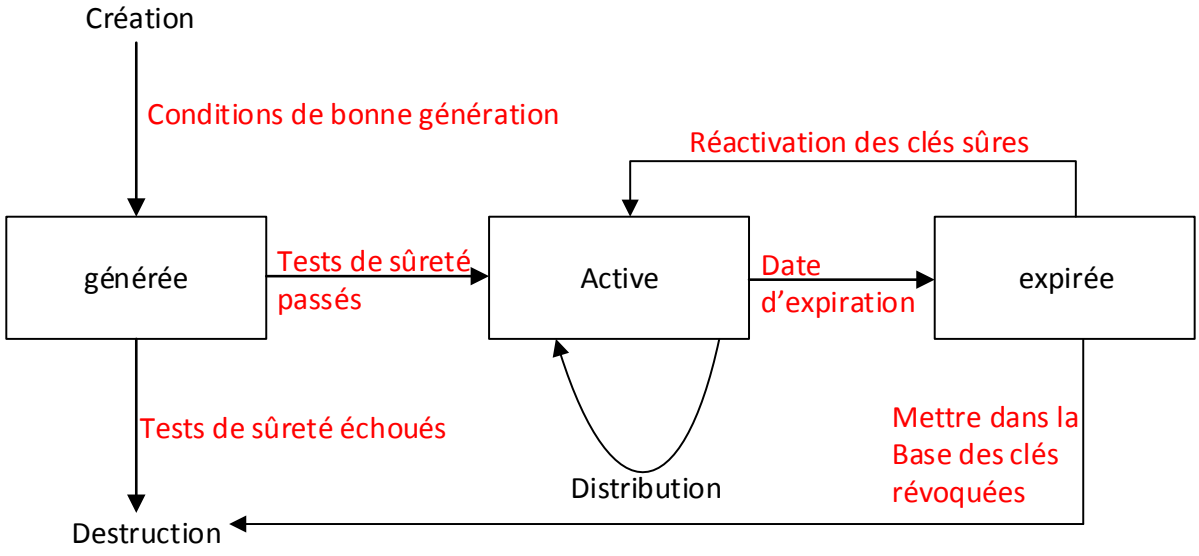

Figure 15: Cycle de vie d'une clé de chiffrement

La création : le processus de création doit respecter les règles de génération de clés fortes ; une série de tests est appliquée pour vérifier l'adéquation de la clé avec les règles pour éviter, par exemple, des clés faibles. Toute clé qui ne répond pas aux tests de sûreté sera détruite. Les clés qui sont bien générées passent en état générées.

L'activation : cette étape rend la clé utilisable pour les opérations de chiffrement, une série de tests est appliquée sur les clés générées avant leur activation. En pratique, une clé a une durée de vie limitée dans le temps et en nombre d'utilisations.

Expiration : la durée de vie d'une clé est définie par le temps nécessaire pour la cryptanalyse des messages chiffrés et/ou des clés. La durée définit la sécurité calculatoire du système de chiffrement.

Les clés compromises doivent être désactivées, détruites et signalées dans un annuaire de clés révoquées. Cet annuaire regroupe toutes les clés dont l'usage est interdit.

Une clé périmée peut être réactivée si cette dernière répond toujours aux règles de sécurité définies lors de sa création.

3. Distribution de clés

Une des tâches les plus importantes dans le cycle de vie d'une clé est le processus de distribution de cette dernière. En effet, lors de la distribution, on doit éviter la modification de la clé, sa destruction malintentionnée et son rejeu par un tiers non autorisé.
On peut distinguer plusieurs approches de distribution de clés selon le type de la clé. S'il s'agit de clés publiques, il n'est à priori pas nécessaire de les chiffrer pour les transmettre. En revanche, les clés symétriques doivent être transmises d'une manière sécurisée (Enveloppe chiffrée ou canal sécurisé).

3.1 Distribution des clés symétriques

La distribution de clés symétriques permet le partage sécurisé des clés entre les entités désirant entrer en communication chiffrée. Pour ce faire, on peut faire recours à des techniques à clé symétrique ou à clé asymétrique.

A)Par mécanisme symétrique

La distribution des clés symétriques au moyen de mécanismes symétriques utilise des clés secrètes partagées au préalable entre les entités. Un mécanisme de distribution symétrique peut faire appel ou non à un centre de distribution de clés(KDC). On donne un exemple de mécanisme pour chacun des types de distribution.

Mécanisme sans centre de distribution de clés (KDC) :

Le mécanisme de base est illustré dans la figure 16.

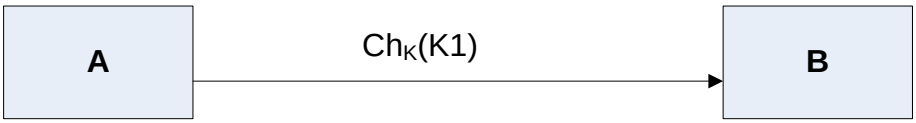

Figure 16: Échange de clés secrètes par un mécanisme symétrique sans KDC

Dans ce cas on suppose qu'une clé K est partagée au préalable entre les communicants A et B et que A est le générateur de la clé K1 à partager.

Ce mécanisme de base ne procure qu'une authentification implicite de A. Néanmoins, il est possible de l'améliorer pour obtenir une authentification de la clé en garantissant sa fraîcheur. Pour ce faire, il suffit de marquer la clé à échanger par une estampille ou une référence.

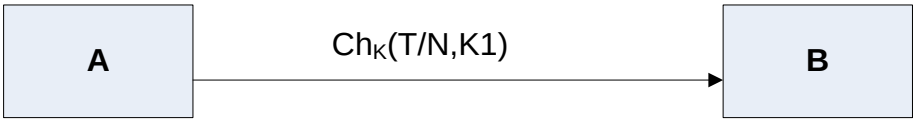

Figure 17: Partage de clés secrètes par un mécanisme symétrique sans KDC avec garantie de fraîcheur.

A envoie à B une estampille T ou un numéro de séquence N et la clé à échanger K1 le tout est chiffré par la clé symétrique K préalablement partagée.
B déchiffre le message et obtient la clé K1 et vérifie sa fraîcheur grâce à l'estampille T ou le numéro de séquence N.
Le fait de marquer la clé permet d'éviter l'attaque **par rejeu de la clé**

Mécanisme avec centre de distribution de clés (KDC) :

La distribution de clés secrètes au moyen d'un mécanisme symétrique avec un centre de distribution de clés utilise une clé partagée entre le demandeur de la communication (**A**) et

le centre KDC et une autre clé partagée entre le destinataire (**B**) et le KDC. L'objectif est de permettre à **A** de retransmettre une clé secrète $\mathbf{K_{AB}}$ engendrée par le KDC.

Fonctionnement :

1- **A** adresse une requête au **KDC** contenant son identité ($\mathbf{ID_A}$), l'identité du destinataire ($\mathbf{ID_B}$) et un jeton **N** .{ **N,** $\mathbf{ID_A}$**,** $\mathbf{ID_B}$ } ;
2- Le **KDC** prépare un message chiffré au moyen de la clé $\mathbf{K_A}$ qui contient le jeton de **A** (**N**), une confirmation de l'identité de **B** ($\mathbf{ID_B}$) , la clé secrète à utiliser entre **A** et **B** ($\mathbf{K_{AB}}$) et un message à transmettre à **B** chiffré avec $\mathbf{K_B}$. le message chiffré contient la clé partagée $\mathbf{K_{AB}}$ et l'identité de A. {$\mathbf{Ch_{KA}(N, ID_B, K_{AB}, Ch_{KB}(ID_A,K_{AB}))}$ };
3- **A** extrait { $\mathbf{Ch_{KB}(ID_A,K_{AB})}$ } issu du **KDC** et le transmet à **B**;
4- **B** reçoit le $\mathbf{Ch_{KB}(ID_A,K_{AB})}$ envoyé par **A**, extrait la clé $\mathbf{K_{AB}}$ et l'identité de son correspondant ; il renvoie alors à **A** un accusé de réception ($\mathbf{N_B}$) chiffré avec la clé $\mathbf{K_{AB}}$. { $\mathbf{Ch_{KAB}(N_B)}$ } ;
5- **A** accuse bonne réception en effectuant une incrémentation sur le jeton $\mathbf{N_B}$ ($\mathbf{N_B+1}$). { $\mathbf{Ch_{KAB}(N_B+1)}$}.

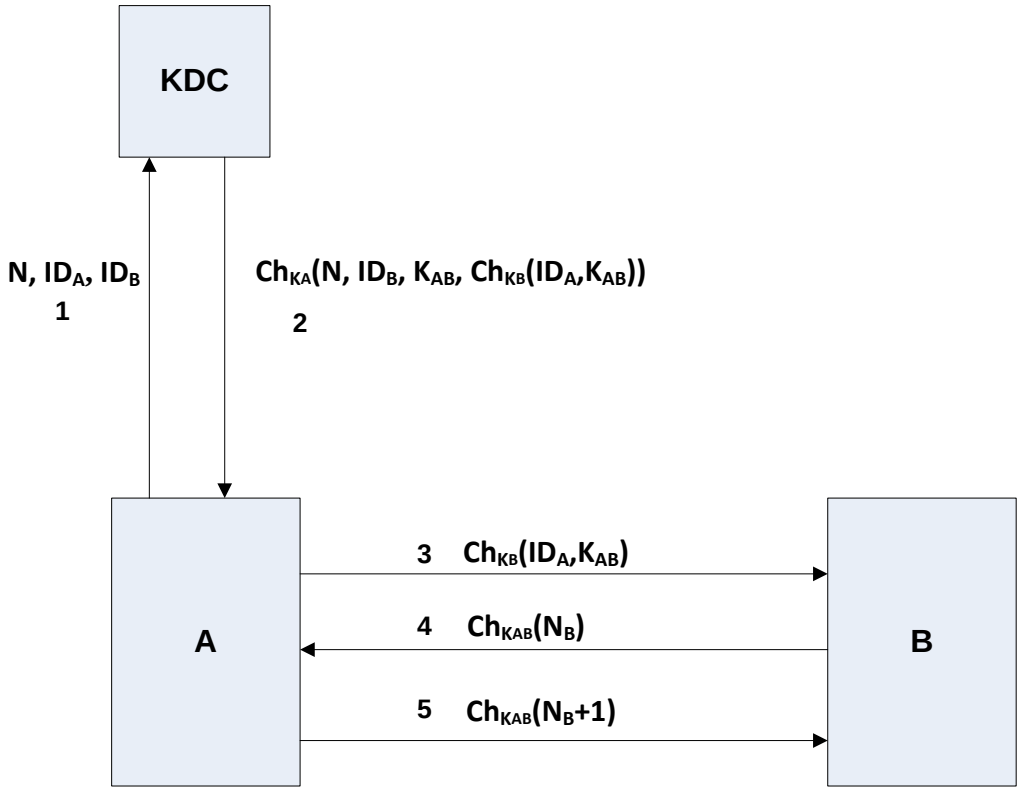

Figure 18: Distribution de clés secrètes par un mécanisme symétrique avec KDC

B)Par mécanisme Asymétrique

La solution consiste à utiliser un système cryptographique Asymétrique (Chapitre 3, section 3) afin d'assurer un partage sécurisé de la clé symétrique. Le principe est illustré par la figure 19.

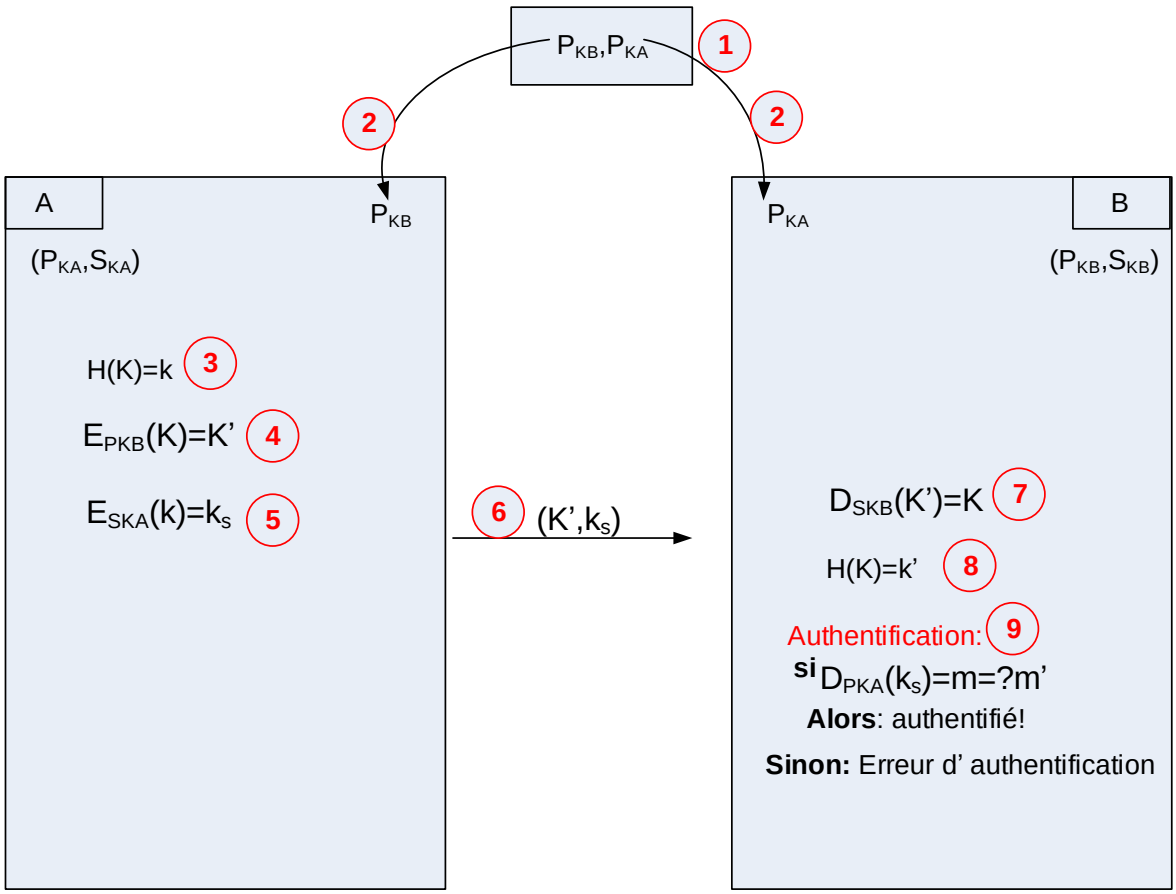

Figure 19: Partage d'une clé symétrique à base d'un système asymétrique

Initialement, chaque entité doit disposer de son couple de clés de chiffrement asymétrique. Une fois les clés publiques partagées (étapes 1 et 2), l'entité A génère une clé symétrique K à partager. Le processus de partage inclut les opérations de hachage, signature et chiffrement (étapes 3,4 et 5) afin de garantir l'intégrité, l'authentification et la confidentialité (étapes 7,8 et9) de la clé K.

3.2 Distribution de clés asymétriques

La gestion de clés asymétriques implique la génération et le stockage des paires de clés (publique privée), la protection, la distribution et la révocation de la clé publique. Le processus de gestion de clés est la partie la plus délicate d'un système asymétrique.

Il existe deux approches de gestion de clés asymétriques : celle basée sur une PKI (Public Key Infrastructure) et celle utilisant un réseau de confiance.

A) Approche basée sur une PKI (Public Key Infrastructure)

Dans ce modèle, la gestion de clés est assurée par une infrastructure de gestion de clés **PKI** (Public Key Infrastructure).

La première tâche d'une PKI est la génération d'une paire de clés. En fonction de l'organisation de la **PKI**, ceci peut être fait localement ou par un serveur central de distribution de clés **KDC**(key distribution Center).

La clé privée est stockée chez l'utilisateur et le mieux est de la chiffrer et de la conserver sur un media non accessible à travers un réseau.

La clé publique doit être enregistrée et validée par une autorité de certification, **CA**(Certificate authority). La CA fournit à l'utilisateur un certificat attestant du lien entre lui et la clé demandée.

Le processus de gestion de clés par une PKI est illustré par la figure 20.

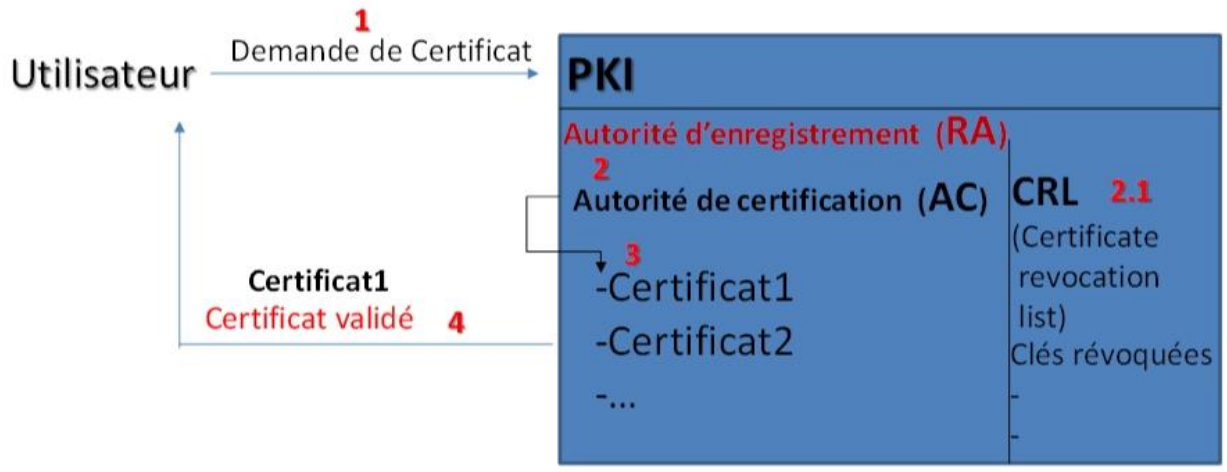

Figure 20: Gestion de clés par PKI

Autorité de certification (AC)

L'AC fait partie d'une PKI et avec sa signature, elle certifie les clés publiques à leurs propriétaires. C'est un organisme auquel tout le monde fait confiance. Les utilisateurs ont tous la clé publique de l'AC, elle leur permet de vérifier, par la suite, les signatures faites par l'AC.

Principalement, une AC assure les tâches suivantes :

a-Enregistrement des utilisateurs : Quand un utilisateur demande une clé d'un système à clés publiques, il est enregistré par l'AC du système. L'utilisateur doit communiquer son identité et d'autres informations personnelles. L'AC vérifie ces informations et ajoute l'utilisateur au système en faisant appel à l'autorité d'enregistrement.

b-Fabrication de clés : Le couple de clés (clé publique, clé privée) peuvent être fabriquées soit chez l'utilisateur soit chez l'AC. La clé privée reste chez l'utilisateur par contre la clé publique est mise dans un annuaire de l'AC.

c-Certification de la clé publique : L'AC génère un document (certificat) qui associe une clé publique à une entité (personne, serveur, site….). Une fois généré, le certificat est toujours signé par la clé privée de l'AC.

Un certificat contient principalement les informations suivantes :

- ✓ Identité du propriétaire de la clé ;
- ✓ La clé publique ;
- ✓ Le nom de l'algorithme de chiffrement utilisé ;
- ✓ Le numéro de série du certificat ;
- ✓ Date de début et date de fin de validité du certificat ;
- ✓ Le nom de l'AC qui la généré ;
- ✓ Les restrictions qui s'appliquent à l'usage du certificat.

Le certificat est ensuite stocké dans un annuaire.

d-Révocation de certificats : La **CRL** (Certificate revocation list) contient la liste des identifiants des certificats qui ont été révoqués ou invalidés. Un certificat peut être révoqué avant même sa date d'expiration. La **CRL** (Certificate revocation list) est mise à jour régulièrement par l'autorité de certification. Tout expéditeur voulant utiliser une clé publique doit contrôler tout d'abord que le certificat qui atteste de la validité de la clé publique n'existe pas dans la **CRL**.

Chaine de certificats [MAR 2004]

En pratique on trouve généralement une autorité de certification racine et des sous autorités de certifications. Il peut y avoir plus de deux niveaux de certificats : celui qui garantit la relation entre l'identité de l'utilisateur et sa clé publique et celui qui garantit la relation entre l'identité de l'AC et sa clé publique. Dans ce cas, on parle de *chaine de certification* où chaque certificat va garantir l'authenticité du certificat précédent. Le certificat de l'autorité racine est auto signé par la clé de cette dernière.

Exemple : soit un système de certification à trois niveaux (figure 21), une autorité racine AC_r, une autorité d'un premier niveau AC_1 et une autre autorité de deuxième niveau AC.

Afin de vérifier la validité du certificat des entités A et B, il suffit de disposer de la clé publique (certificat) de l'autorité racine AC_r. Alors que la vérification de la validité du certificat de C nécessite l'application de trois vérifications successives : vérifier la signature du niveau AC, puis remonter au niveau de AC1 et enfin le niveau racine ACr.

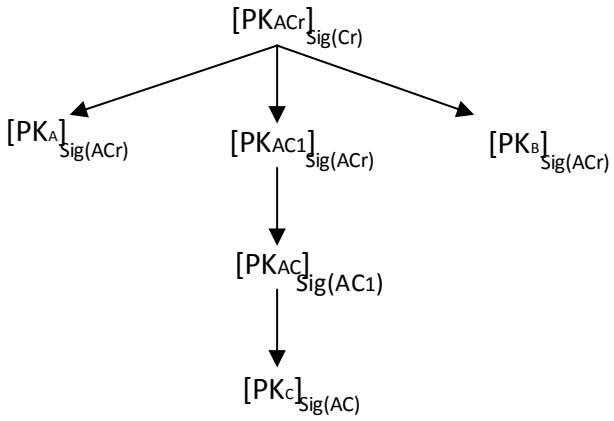

Figure 21: Chaine de certificats

B) Approche basée sur les réseaux de confiance

Le concept de réseau de confiance a été élaboré initialement par Phil Zimmermann [ZIM 1999], créateur de PGP (Pretty Good Privacy). Il existe plusieurs réseaux de confiances compatibles avec PGP (GnuPG, OpenPGP,.. etc.). Le réseau de confiance permet la création de relations de confiances entre les clés publiques et leurs propriétaires.

C'est un modèle de confiance décentralisé, il vient comme alternative aux modèles de confiance centralisés avec PKI. Au lieu de faire passer la relation de confiance par une autorité de certification (CA ou hiérarchie de CA), un réseau de confiance est construit à base de relations définies par les utilisateurs entre eux. En effet, un utilisateur peut choisir lui-même les tiers à qui il fait confiance.

En pratique, il existe de nombreux réseaux de confiance indépendants qui offrent une libre adhésion. Un utilisateur peut en faire partie via son propre certificat, et peut même devenir un lien entre différents réseaux.

Fonctionnement

Dans PGP, les certificats de clés publiques sont validés par la vérification des signatures numériques d'autres utilisateurs du réseau de confiance sur ces clés. Ces utilisateurs, en signant ces certificats, peuvent renforcer l'association entre les clés publiques et leurs propriétaires annoncés par ces certificats.

En règle générale, un certificat reçu par un utilisateur est considéré comme valide :

- Si cet utilisateur lui-même l'a déjà signé.
- Ou, si le certificat est signé par une personne à laquelle cet utilisateur a donné sa confiance totale.
- Ou, si le certificat est signé par trois personnes auxquelles cet utilisateur a donné sa confiance partielle.

On note que ces paramètres sont ajustables par l'utilisateur, en fonction de ses besoins.

Conclusion

Ce présent support de cours donne l'essentiel de la sécurité informatique, en commençant par introduire la cryptographie et les systèmes de chiffrement, en passant par les fonctions de hachage et signature numérique jusqu'à la gestion de clés. Nous avons essayé de simplifier les notions introduites pour ne pas encombrer l'étudiant avec beaucoup de détails qu'il pourra acquérir dans l'avenir en formation de spécialité.

Les concepts introduits par ce cours sont généralement appuyés par des exemples afin de faciliter leur compréhension par l'étudiant.

Enfin, Nous espérons que ce support de cours va répondre aux attentes des étudiants et qu'il sera un bon guide pour comprendre les bases de la sécurité informatique.

Références bibliographiques

[DAM 1989] Damgard, I. B. (1989, August). A design principle for hash functions. In Conference on the Theory and Application of Cryptology (pp. 416-427). Springer, New York, NY.

[DIF 1976] Diffie, W., & Hellman, M. (1976). New directions in cryptography. IEEE transactions on Information Theory, 22(6), 644-654.

[DUM 2013] Dumas, J. G., Roch, J. L., Tannier, E., & Varrette, S. (2013). Théorie des Codes: compression, cryptage, correction (p. 384). Dunod.

[ELG 1985] ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory, 31(4), 469-472.

[FEI 1973] Feistel, H. (*1973*) Cryptography and Computer Privacy. Scientific American, 228, 15-23

[FIP 1999] FIPS Publication 46-3, Data Encryption Standard, https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf

[FLU 2001] Fluhrer, S., Mantin, I., and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4", Selected Areas of Cryptography: SAC 2001, Lecture Notes in Computer Science Vol. 2259, pp 1-24, 2001.

[GAD 2018] Gadouche, H., Farah, Z., & Tari, A. (2018, October). A Correct-by-Construction Model for Attribute-Based Access Control. In International Conference on Model and Data Engineering (pp. 233-247). Springer, Cham.

[LES 1931] Lester S. Hill, « Concerning certain linear transformation apparatus of cryptography », American Mathematical Monthly, vol. 38, 1931, p. 135-154.

[MAR 2004] Martin, B. (2004). Codage, cryptologie et applications. PPUR presses polytechniques.

[MER 1978] Merkle, R., & Hellman, M. (1978). Hiding information and signatures in trapdoor knapsacks. IEEE transactions on Information Theory, 24(5), 525-530.

[PAU 2011] Paul, G., & Maitra, S. (2011). RC4 stream cipher and its variants. CRC press.

[RIV 1978] Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.

[SHA 1948] Shannon, C. E. (1948). A mathematical theory of communication. Bell system technical journal, 27(3), 379-423.

[SHE 2002] Sheldon, R. (2002). A first course in probability. Pearson Education India.

[ZIM 1999] Zimmermann, Philip R. (1999). "Why I Wrote PGP". Essays on PGP. Philip Zimmermann.