Université Cheikh Anta DIOP de Dakar

Faculté des Sciences et Techniques

(F. S. T.)

Département de Mathématiques et Informatique

(D. M. I.)

Laboratoire d'Algèbre de Cryptologie de Géométrie Algèbrique et Applications

(L. A. C. G. A. A.)

TITRE :

CRYPTOGRAPHIE

Demba SOW

Docteur en Mathématiques et Cryptologie

Enseignant chercheur au Département de Mathématiques/Informatique

Email : demba1.sow@ucad.edu.sn, sowdembis@yahoo.fr

Année académique 2018-2019

Table des matières

Chapitre 1

Notions de Cryptographie

Introduction

- L'origine de la **cryptographie** remonte à **4000 avant J.C** en **Egypte pharaonique**.
- Jusqu'à une époque récente, la cryptographie était réservée aux **domaines militaire et diplomatique**.
- Par exemple, les spécialistes disent que, sans la **cryptanalyse** de la machine **Enigma** par l'équipe de **Alan Turing**, les **Alliers** gagneraient difficilement la **Deuxième Guerre Mondiale**.
- Initialement, la cryptographie s'occupait principalement des **problèmes de confidentialité** bien qu'il existait des **techniques (protocoles) d'authentification** entre autres.
- **Tous les algorithmes et protocoles inventés avant les années 70, ont été complètement cassés.**
- On se refère à cette période en parlant de période de la **cryptographie classique ou artisanale** (art des codes secrets) : 4000 ans avant J.C jusqu'en 1975/76.

1.1 Cryptographie moderne

- Dans les **années 70**, considérées comme le début de l'époque moderne, la cryptographie, devenue **la science des codes secrets**, s'est développée dans le monde civil surtout avec la prolifération des systèmes de communication et de nouveaux services des années 1990/2000 : **Internet, Commerce électronique**.
- Cette nouvelle cryptographie est dite moderne parce que entre autres :
  - toutes ses branches ont connues une modélisation mathématique plus cohérente ;
  - elle a produit des outils (algorithmes, protocoles,...) qui restent encore robustes malgrés le développement des techniques de cryptanalyse ;
  - elle prend en charge presque totalement tous les besoins de sécurité dans un schéma de communication ;

- ♦ son application dans le monde civil a permi le développement de nouveaux métiers ou services : **commerce électronique, e-banking, consultation de données personelles sur internet,...**

1.1.1 Terminologie cryptographique

- ■ La **cryptographie** utilise des concepts issus de nombreux domaines (Informatique, Mathématiques, Electronique).
- ■ Toutefois,les techniques évoluent et trouvent aujourd'hui régulièrement racine dans d'autres branches (**Biologie**, **Physique**, etc.)
- ♦ **Cryptologie** : Il s'agit d'une science mathématique comportant deux branches : la cryptographie et la cryptanalyse
- ♦ **Cryptographie** : c'est une discipline incluant les principes, les moyens et des méthodes de transformation des données, dans le but :
  - de masquer leur contenu (**Confidentialité**),
  - d'empêcher leur modification (**Intégrité**),
  - ou leur utilisation illégale (**Authentification et Non-Répudiation**).
- ♦ **Cryptanalyse** : c'est l'étude des faiblesses des outils de sécurité (algorithmes) produits par la cryptographie dans le but de les corriger ou nuire au système de communication .
- ♦ **Chiffrement** : Le chiffrement consiste à transformer une donnée (texte, message, ...) afin de la rendre incompréhensible par une personne autre que celui qui a créé le message et celui qui en est le destinataire.
- ♦ **Déchiffrement** : C'est la fonction permettant de retrouver le texte clair à partir du texte chiffré.
- ♦ **Texte chiffré** : Appelé également **cryptogramme**, le texte chiffré est le résultat de l'application d'un chiffrement à un texte clair.
- ♦ **Clé** : Il s'agit du paramètre impliqué et autorisant des opérations de **chiffrement** et/ou **déchiffrement**.
  - ► Dans le cas d'un algorithme symétrique, la clef est identique lors des deux opérations.
  - ► Dans le cas d'algorithmes asymétriques, elle diffère pour les deux opérations.
- ♦ **Cryptosystème** : Il est défini comme l'ensemble des clés possibles (espace de clés), des textes clairs et chiffrés possibles associés à un algorithme donné.

1.1.2 Modélisation des systèmes de chiffrement

Un système de cryptographie est composé d'un quintuplet $(\mathcal{P}, \mathcal{C}, C_k, D_{k'}, \mathcal{K})$ où :

- $\mathcal{P}$ est un ensemble appelé espace des textes clairs
- $\mathcal{C}$ est un ensemble appelé espace des textes chiffrés
- $\mathcal{K}$ est un ensemble appelé espace des clés
- $C_k : \mathcal{P} \to \mathcal{C}$ est une fonction inversible à gauche appelée fonction de chiffrement et qui dépend d'un parametre k appelé clé.
- $D_{k'} : \mathcal{C} \to \mathcal{P}$ est la fonction inverse à gauche de C_k (i.e $D_{k'} \circ C_k(m) = m, \forall m \in \mathcal{P}$) et est appelée fonction de déchiffrement (dépendant de la clé k').

A partir de ce modèle on a deux cryptosystèmes.

1.1.3 Systèmes de cryptographie

Il existe deux systèmes de cryptographie : le système symétrique ou cryptographie à clés secrètes et le système asymétriqu ou non-symétrique ou cryptographie à clés publiques.

1.1.3.1 Cryptographie à clé secrète

■ **Concept :**

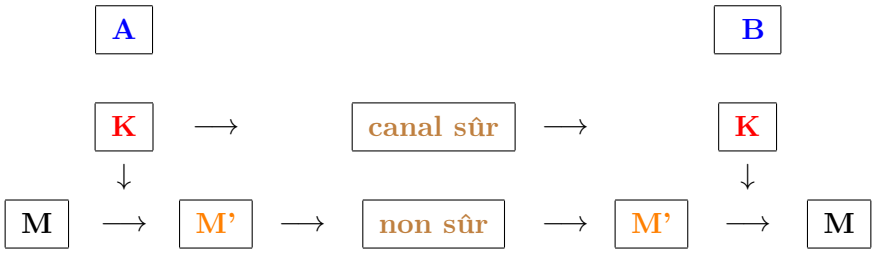
- Les clés de chiffrement (K_E) et de déchiffrement (K_D) sont identiques : $K_E = K_D = K$.
- La clé doit rester secrète.
- Les algorithmes les plus répandus sont le **DES**, **AES**, **3DES**, ...
- Au niveau de la génération des clés, elle est choisie aléatoirement dans l'espace des clés ;
- Ces algorithmes sont basés sur des opérations de transposition et de substitution des bits du texte clair en fonction de la clé ;
- La taille des clés est souvent de l'ordre de **128 bits**. Le **DES** en utilise **56**, mais l'**AES** peut aller jusque **256** ;
- L'avantage principal de ce mode de chiffrement est sa rapidité ;

■ **Limites :**

■ Le principal désavantage réside dans la distribution des clés :

- ♦ Pour une meilleure sécurité, on préfèrera l'échange manuel.
- ♦ Malheureusement, pour de grands systèmes, le nombre de clés peut devenir conséquent.
- ♦ C'est pourquoi on utilisera souvent des échanges sécurisés pour transmettre les clés.
- ♦ En effet, pour un système à N utilisateurs, il y aura $N \cdot (N-1)/2$ paires de clés.

■ **Schéma du système symétrique**



1.1.3.2 Cryptographie à clés publiques

■ **Concept :**

- Une clé publique K_{pub} (pour chiffrer ou vérifier une signature) ;
- Une clé privée K_{priv} (pour déchiffrer ou signer) ;
- Propriété : La connaissance de K_{pub} ne permet pas de déduire K_{priv} ;
- $D_{K_{priv}}(E_{K_{pub}}(M)) = M$;
- L'algorithme de cryptographie asymétrique le plus connu est le **RSA** ;

■ **Principe :**

- Le principe de ce genre d'algorithme est qu'il s'agit d'une fonction unidirectionnelle à trappe.
  - Une telle fonction a la particularité d'être facile à calculer dans un sens, mais difficile voire impossible dans le sens inverse.
  - La seule manière de pouvoir réaliser le calcul inverse est de connaître une trappe. Une trappe pourrait par exemple être une faille dans le générateur de clés.
  - Cette faille peut être soit intentionnelle de la part du concepteur (définition stricte d'une trappe) ou accidentelle.

■ **Problèmes mathématiques :**

- Les algorithmes se basent sur des concepts mathématiques tels que
  - l'exponentiation de grands nombres premiers (RSA) ;
  - le problème des logarithmes discrets (ElGamal) ;
  - le problème du sac à dos (Merkle-Hellman).
  - le problème sur les réseaux arithmétiques (NTRU).
- La taille des clés s'étend de 1024 bits à 4096 bits en standard.
- Dans le cas du RSA, une clé de 1024 bits n'est plus sûre, mais est toujours utilisable de

particulier à particulier.

- Au niveau des performances, le chiffrement par voie asymétrique est environ 1000 fois plus lent que le chiffrement symétrique.

Fonctionnement :

- Cependant, à l'inverse du chiffrement symétrique où le nombre de clés est le problème majeur, ici, seules n paires sont nécessaires.
- En effet, chaque utilisateur possède une paire (K_{pub}, K_{priv}) et tous les transferts de message ont lieu avec ces clés.
- La distribution des clés est grandement facilitée car l'échange de clés secrètes n'est plus nécessaire.
- Chaque utilisateur conserve sa clé secrète (privée) sans jamais la divulguer. Seule la clé publique devra être distribuée.

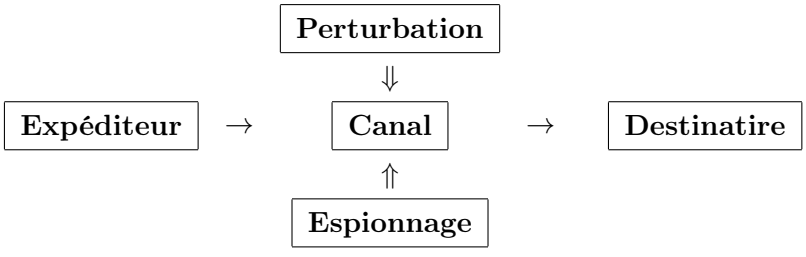
Schéma du système asymétrique (non symétrique)

1.2 Services de la cryptographie

1.2.1 Schéma de communication

⇒ **Un schéma de communication** (**échange d'informations entre entités distantes**) **fait internir :**

- **un expéditeur** (**personne, machine,...**),
- **un destinataire** (**personne, machine,...**),
- **un canal de transmission** (**ligne téléphonique, fibre optique, systèmes de communication sans fils,...**)
- **et un message** (**information : texte, son, image, vidéo,...**).



1.2.2 Besoins de sécurité lors d'une communication

Quand deux interlocoteurs communiquent plusieurs besoins de sécurité se manifestent. Supposons que Alice envoie un message à Bob.

- Peut-on assurer, même si le message envoyé par Alice est intercepté par un ennemi Charlie, que sa signification reste cacher ?
  Confidentialité
- Si le message envoyé par Alice est modifié en cours de la transmission, est-ce que Bob peut le détecter ?
  Intégrité
- Bob, peut-il être sûr d'avoir reçu un message provenant d'Alice ?
  Authentification
- Alice, peut-elle nier avoir envoyé un message à Bob ?
  Non répudiation

1.2.3 Services de sécurité

- **Confidentialité :** Propriété qui assure que l'information n'est pas rendue disponible ou révélée à des personnes, entités ou processus non autorisés.
- **Intégrité** : Propriété qui assure que des données n'ont pas été modifiées ou détruites de façon non autorisée lors de leur traitement, stockage et/ou transmission.
- **Non répudiation / Signature** : Fonction qui permet de garantir qu'aucun des partenaires d'une transaction ne pourra nier d'y avoir participé.
- **Authentification** : Fonction permettant de prouver l'identité des partenaires (hommes, machines, réseaux,...) d'une communication ou l'origine des données lors d'un échange ou la validité d'un object cryptographique (clé, carte à puce, certificat/signature,...).

Remarques

- L'**authentification** n'est pas seulement pris en charge par des techniques de cryptographie et il existe plusieurs variantes et niveaux d'authentifications.
- Dans un schéma de communication, l'**authentification** est le service le plus usité et peut être le plus fondamental.
- **En général, plusieurs types d'authentifications sont couplés pour une sécurité avec plusieurs niveaux (barrières)** : on l'appelle **l'authentification forte** ou **multifacteurs.**

- **Il y'a trois facteurs fondamentaux :**
  - **Ce que l'on sait** (mot de passe, phrase secrète, ...)
  - **Ce que l'on a** (clé, token usb, carte magnétique, carte à puce,...)
  - **Ce que l'on est** (Biométrie : empreintes digitales, voix, scanner rétinien, ADN/Chromosomes, ...)
  - **Ce que l'on sait faire** (calculer une opération, lire le contenu d'une image, ...)

1.3 Primitives de la cryptographie

■ Pour réaliser les fonctions et objectifs de sécurité , la cryptographie utilise plusieurs outils de bases appelés **primitives de cryptographie**.

■ Mais aussi des compositions plus complexes parfois que sont **les protocoles de cryptographie**.

■ Parmi ces primitives ou algorithmes, on peut citer :

- **Algorithmes (pour le chiffrement) :**
  - **à clé secrète** : DES, AES, RC4, etc.
  - **à clé publique** : RSA, Mc-Eliece, El Gamal etc.
  - **chiffrement basé sur les réseaux arithmétiques** : NTRU-Encrypt, ect.
- **Fonctions de hachage :(pour garantir l'intégrité par la création d'empreinte)** MD5, SHA-1, SHA-2, SHA-3 etc.
- **Schémas de signatures numériques :(pour la non répudiation)** : DSA, EC-DSA, PSS-RSA etc.
- **Signature basée sur les réseaux arithmétiques** : NTRU-Sign, ect.

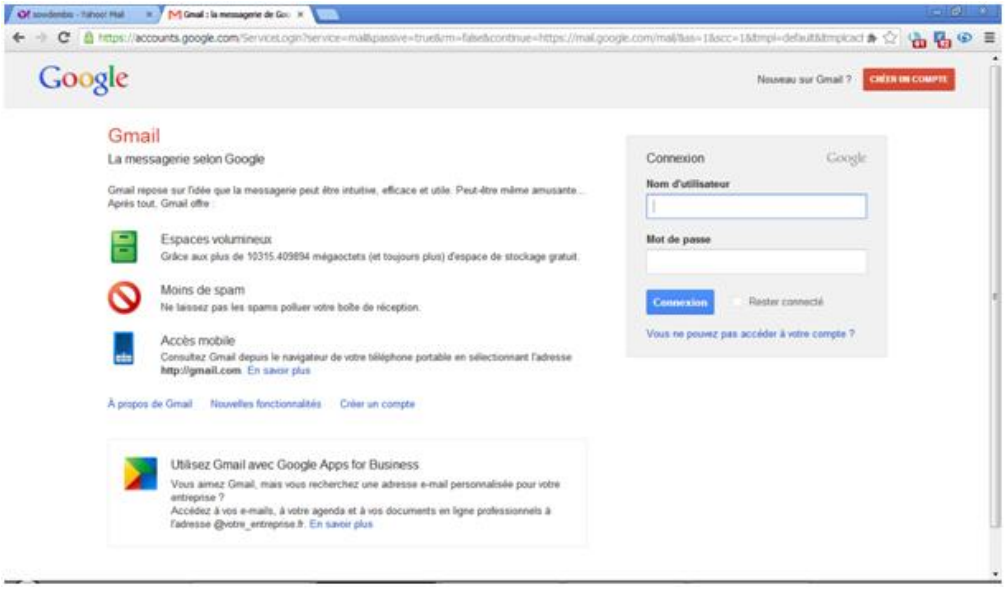
■ Parmi ces protocoles, on peut citer :

- **Diffie-Hellman (pour l'échange de clés).**
- **Fiat-Shamir (pour l'identification).**
- **Needham-Schrôder (pour l'authentification mutuelle).**

1.4 Où applique t-on la cryptographie ?

Internet : confidentialité, anonymat, authentification.

S'agit-il bien de ma banque ?



Signature électronique : vérifiable, authentique, non-répudiation (je n'ai jamais signé ce texte ...)

Vote électronique : le résultat reflète le vote, chaque vote est confidentiel, on ne peut pas connaître des résultats partiels, seuls les électeurs peuvent voter et une seule fois

Paiement par carte bancaire : est-ce qu'il s'agit d'une vraie carte ? Est-ce que le montant débité sera égal au montant crédité ? Est-ce que le code secret est bien protégé ?

Décodeurs : vérification de l'abonné, impossibilité de retransmettre les données décodées à une tierce personne, mise à jour de l'abonnement ...

Porte monnaie électronique : pas de création de fausse monnaie, pas de création de faux porte-monnaie ...

Bases de données sécurisées : (ex : carte vitale, seules les personnes habilitées ont accès à la vue partielle à laquelle elles ont droit, les données peuvent être échangées entre un médecin, un laboratoire, un hôpital, mise à jour possible des données)

1.5 Notion de cryptanalyse

■ **La cryptanalyse** a pour principal objet, d'étudier les faiblesses des outils de sécurité produits par la cryptographie dans le but de les corriger ou nuire au système de communication .

■ **Attaquant** : Entité [nommée **Charlie**] suceptible d'agir sur un schéma de communication dans le but de nuire c'est à dire :

- ♦ de violer la confidentialité des données,
- ♦ de détourner et de modifier des données ou de récupérer des informations,
- ♦ d'usurper l'identitité de l'un des partenaires de la communication.

■ **Types d'attaques**

Il y'a deux classes d'attaques :

■ **Attaques passives** : l'attaquant écoute seulement. Donc, c'est une attaque qui cible la confidentialité.

■ **Attaques actives** : l'attaquant agit sur le système de commnucation pour :

- ♦ détourner et modifier des données ou récupérer des informations,
- ♦ usurper l'identitité de l'un des partenaires de la communication.

$\Longrightarrow$ Donc les attaques actives ciblent toutes les fonctions de sécurité : confidentialité, intégrité, authentification-identification et non répudiation.

■ Pour celà, l'attaquant cherche généralement à mettre à défaut l'une des primitives de cryptographie.

■ Par exemple :

- ♦ **pour les algorithmes** : récupérer les clés de chiffrement, déchiffrer les cryptogrammes ou de casser complètement les algorithmes utilisés ;
- ♦ **pour les protocoles** : détourner l'objectif d'un protocole, compromettre le déroulement d'un protocole ;
- ♦ **pour le hachage** : fabriquer de fausses empreintes,
- ♦ **pour les signatures** : falsifier les signatures.

■ Ces attaques peuvent être dirigées :

- ♦ sur les modèles mathématiques utilisés pour fabriquer les primitives de cryptographie ;
- ♦ sur l'implémentation matérielle et/ou logicielle des pritmitives de cryptographie ;
- ♦ sur les entités propriétaires (légitimes) de données ou d'objets cryptographiques (secrets) ;
- ♦ sur les acteurs légitimes d'un scénarios de communication ;
- ♦ sur la gestion (fabrication, distribution, stockage, tests de validité,...) de données ou d'objets cryptographiques ;

1.5.1 Sécurité d'un chiffrement

■ **Un système de chiffrement est dit sûr si la probabilité d'obtenir une information sur le texte clair ou la clé de déchiffrement à partir du chiffré est presque nulle**.

■ Celà veut dire que :

- ♦ toute information qu'on peut tirer du texte clair sera si faible qu'elle ne permettra pas de violer sa confidentialité partielle ou complète ;
- ♦ ou que toute information qu'on peut extraire de la clé sera si faible qu'elle ne permettra pas de la reconstituée substantiellement.

1.5.2 Taille des paramètres de sécurité

■ Un PC à $1GHz = 10^9 Hz$ effectue 10^9 opérations élémentaires par seconde.

■ ***Opérations élémentaires :*** affectation, instructions de contrôle, calcul binaire,...

■ PUISSANCE DE CALCUL DES MACHINES

Temps	Nbre operations / 1 PC	Nbre operat / 10^{18} PC
1s	10^9	...
1 an	$3,1.10^{16}$	$3,1.10^{34}$
1000 ans	$3,1.10^{19}$	$3,1.10^{37}$
10^9 ans	...	...
15.10^9 ans	$46,5.10^{25}$	$46,5.10^{43}$

■ La vitesse de la lumière est de $300000kms/s = 3 \times 10^8 m/s$ donc elle traverse une pièce de 3 métres de largeur en un dix milliardième de seconde.

■ Pendant ce temps, un PC à $1GHz$ peut effectuer 10 opérations élémentaires !.

■ Un PC à $1GHz = 10^9 Hz$ effectue 10^9 opérations ou instructions élémentaires par seconde.

■ Combien de temps faut-il pour qu'il puisse casser une clé de taille m par force brute ?

■ Pour cela la machine doit tester toutes les 2^m clés possibles ! On suppose que le PC peut tester une clé par instruction c'est à dire 10^9 clés par seconde.

■ Comme $10^9 = 2^{30}$ et $1an = 31536000s \cong 2^{25}s$, alors le PC peut tester 2^{55} clés par an d'où on a le tableau suivant :

■ SECURITE SUR LA TAILLE DES DONNEES "SECRETES"

taille m	Temps pour 1 PC	Temps pour $10^{18} = 2^{60}$ PCs
56	$2^{26}s = 2ans$	$2^{-59}s$=...
64	$2^{34}s = 2^9 ans = 512ans$	$... = 2^{-51}ans$
128	$2^{98}s = 2^{73}ans$	$2^{13}ans =$ **8192ans**
256	$2^{226} = 2^{201}ans$	$2^{141}ans \gg$ **age Univers**
1024	...	...
2048	$2^{2018} = ...$	$2^{1993} = ...$

■ NIVEAU DE SECURITE

♦ La taille des clés, des données chiffrées et des valeurs aléatoires secrètes doivent au moins être de taille **128**

♦ S'il n'y a pas d'autres attaques à prendre en compte en dehors de l'attaque par force brute, il n'y a pas de raison de choisir des des données sensibles de taille supérieur à **256**

■ **NB : Avec les ordinateurs quantiques, il faut doubler la taille de la clé.**

■ SECURITE DES MOTS DE PASSE

■ **Mots de passe de 8 caractères**

Alphabet	Alphabet et chiffre	Alphanumerique
$26^8 \sim 2^{38}$	$36^8 \sim 2^{42}$	$256^8 \sim 2^{64}$

■ Mots de passe de 22 caractères

Alphabet	Alphabet et chiffre	Alphanumerique
$26^{22} \sim 2^{104}$	$36^{22} \sim 2^{118}$	$256^{22} \sim 2^{176}$

■ Ainsi :

- ♦ Un mot de passe qui protège des données pas trop précieuse doit au moins être de 8 caractères alphanumériques.
- ♦ Un mot de passe qui joue le role d'une clé de chiffrement doit au moins etre de 22 caractères alphanumériques.

Chapitre 2

Cryptographie à clés secrètes

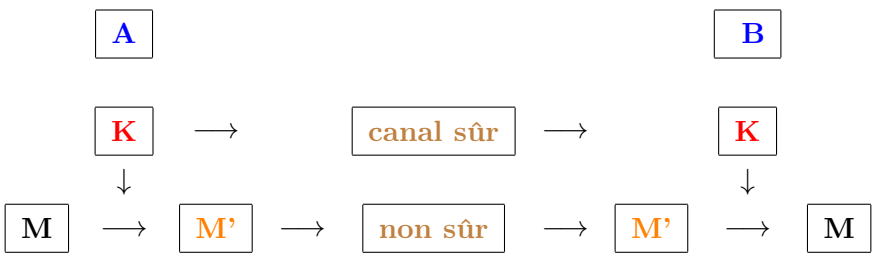
■ **Concept**

- Les clés de chiffrement (K_E) et de déchiffrement (K_D) sont identiques : $K_E = K_D = K$.
- La clé doit rester secrète.
- Les algorithmes les plus répandus sont le **DES**, **AES**, **3DES**, ...
- Au niveau de la génération des clés, elle est choisie aléatoirement dans l'espace des clés ;
- Ces algorithmes sont basés sur des opérations de transposition et de substitution des bits du texte clair en fonction de la clé ;
- La taille des clés est souvent de l'ordre de **128 bits**. Le **DES** en utilise **56**, mais l'**AES** peut aller jusque **256** ;
- L'avantage principal de ce mode de chiffrement est sa rapidité ;

■ Le principal désavantage réside dans la distribution des clés :

- Pour une meilleure sécurité, on préfèrera l'échange manuel.
- Malheureusement, pour de grands systèmes, le nombre de clés peut devenir conséquent.
- C'est pourquoi on utilisera souvent des échanges sécurisés pour transmettre les clés.
- En effet, pour un système à N utilisateurs, il y aura $N \cdot (N-1)/2$ paires de clés.

■ **Schéma du système symétrique**



2.1 Chiffrement par blocs

- On **divise le message en blocs de taille fixe** et on chiffre chacun séparément (mais pas forcément indépendamment).
- **Un chiffrement par blocs est une fonction**

$$E : \{0,1\}^M \times \{0,1\}^N \longrightarrow \{0,1\}^N$$

$$(K, x) \longmapsto E_K(x)$$

(K = la clé, x = le message ou clair, et $E_K(x)$ = chiffré) telle que $\forall\ K$ la fonction $x \mapsto E_K(x)$ soit une permutation.

- Il existe donc une fonction de déchiffrement

$$E^{-1} : \{0,1\}^M \times \{0,1\}^N \longrightarrow \{0,1\}^N$$

$$(K, y) \longmapsto E_K^{-1}(y)$$

avec $E_K^{-1}(E_K(x)) = x$.

- Ici, M = longueur de clé, et N = taille de bloc.
- Exemples typiques : **AES** a $N = 128$ et $M \in \{128; 192; 256\}$.
- $E_K(x)$ doit être facile/rapide à calculer si on connaît K (et x), et E.
- Surtout si on ne change pas K entre plusieurs valeurs de x.
- Exemple : ~ 20 cycles par octet pour chiffrer du **AES** à clé fixée sur un processeur **Intel moderne** en mode **64-bits** (soit $\sim 120 Mo/s$ sur un processeur à $2,4 GHz$, à peu près la vitesse d'un Ethernet gigabit).
- Si on ne connaît pas K, la fonction E_K doit résister à différentes sortes d'attaques.
- Idéalisation mathématique : E doit se comporter comme si chacune des permutations E_K avait été tirée au hasard une fois pour toutes, indépendamment les unes des autres, parmi toutes les permutations possibles.

Conception des chiffrements par blocs

■ **Concevoir un chiffrement qui soit rapide ou sûr est facile.**

■ **La difficulté est d'en concevoir un qui soit rapide et sûr !**

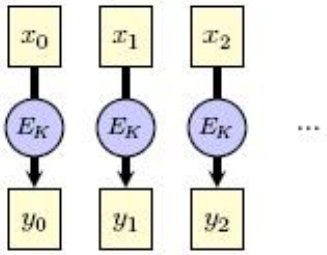
1. Combiner des opérations simples (**XOR** ($\oplus$), addition modulo 2^N ($\boxplus$), autres opérations logiques, rotations de bits, permutations d'octets ou de mots, **S-box** = **boîtes de substitution**) pour obtenir quelque chose de compliqué.
2. **Appliquer plusieurs fois successivement la même fonction simple** (**fonction de tour**) en utilisant des **sous-clés dérivées de la clé** K. Chacune de ces applications s'appelle un **tour**.

3. Rechercher les propriétés suivantes sur la fonction de tour (**Shannon**) :
   - **diffusion** = propagation rapide de l'information des bits du clair ou de la clé vers les bits du chiffré ;
   - **confusion** = absence de relations simples entre les bits du clair ou de la clé et du chiffré.

2.2 Modes Opératoires

2.2.1 Modes opératoires : ECB

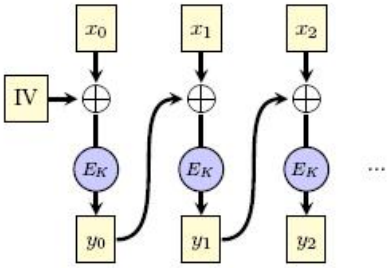
- **Problème :** on veut chiffrer un message de **longueur** $> N$, arbitrairement grande.
- **Idée la plus évidente :** "**electronic codebook mode**" (**ECB**) : découper le message en blocs de longueur N, chiffrer chacun indépendamment.



- **Avantages :** simple, parallélisable, possibilité d'accès aléatoire aux données déchiffrées, pas de propagation des erreurs.
- **Inconvénients :** les motifs répétés du clair sont apparents ;
  pas de garantie d'intégrité : un attaquant pourrait facilement modifier l'ordre des blocs, ou en répéter.

2.2.2 Modes opératoires : CBC

"**Cipher block chaining mode**" (**CBC**) : faire le **XOR** de chaque bloc de clair avec le bloc chiffré précédent : $y_i = E_K(x_i) \oplus y_{i-1}$.



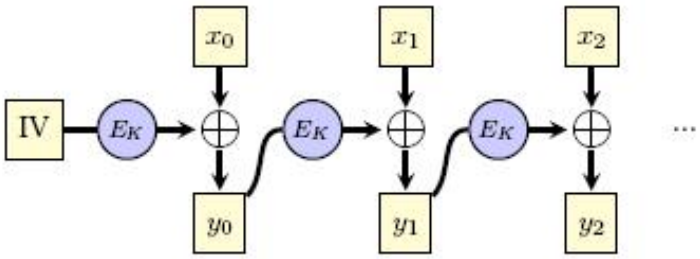
IV (pour "**initialization vector**") est une valeur qui n'est pas secrète dont le rôle est d'éviter que le chiffrement de deux clairs identiques soit identique.

- **CBC** est le mode le plus utilisé (mais de moins en moins).
- **Propriétés :** les motifs du clair n'apparaissent pas dans le chiffré, une erreur de transmission du chiffré affecte deux blocs du message déchiffré.

– **Pas de garantie d'intégrité**, mais un attaquant peut plus difficilement en tirer avantage.
– Le chiffrement ne peut se faire que séquentiellement, mais le déchiffrement peut se faire par accès aléatoire.

2.2.3 Modes opératoires : CFB

"**Cipher feedback mode**" (**CFB**) : on fait un **XOR** de chaque bloc d'entrée avec le chiffré de la sortie précédente : $y_i = x_i \oplus E_K(y_i - 1)$

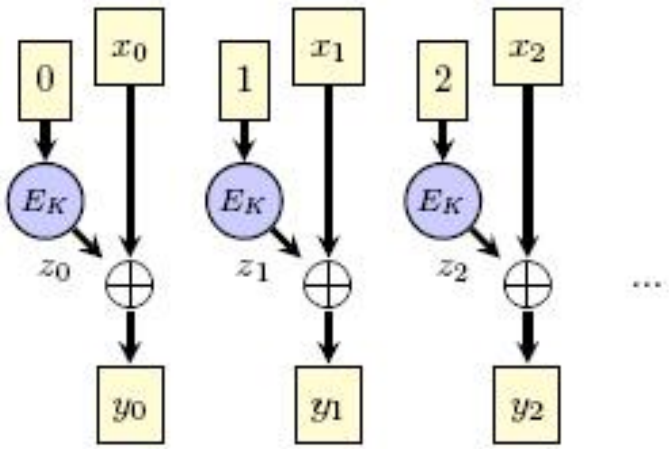


Peu de différence avec **CBC** si utilisé sur des blocs entiers.

Remarque : Le déchiffrement se fait simplement avec la fonction E_K.

2.2.4 Modes opératoires : compteur (CRT)

"**Counter mode**" (**CTR**) : on fait un **XOR** de chaque bloc d'entrée avec une valeur qui est le chiffré d'un simple compteur : $y_i = x_i \oplus z_i$ où $z_i = E_K(i)$.



Autrement dit, les $E_K(i)$ sont utilisés comme un **masque jetable** (**one-time pad**).

Le déchiffrement se fait avec la fonction E_K.

1. **Avantages :** simple, parallélisable, possibilité d'accès aléatoire aux données déchiffrées, pas de propagation des erreurs.

2. **Inconvénients :** il faut prendre garde à ne pas réutiliser la clé : chaque message doit être chiffré avec une clé différente, ou bien (plus courant) les bits supérieurs du compteur doivent être initialisés avec un numéro unique (=nonce) pour chaque message, dont le rôle est semblable au vecteur d'initialisation.

2.3 Chiffrement par flux (par flot)

- **On cherche normalement à aller très vite !**
- **Typiquement** : génération d'un flot de bits ("**keystream**") à partir de la clé (+ un éventuel **nonce**), qui est ensuite utilisé comme **masque à usage unique** (i.e., XOR avec le clair).
- Le mode compteur des chiffrements par blocs peut donc être considéré comme un chiffrement par flot.
- **Possible problème de désynchronisation entre source et destination.**
- Il existe des mécanismes d'**auto-synchronisation** permettant d'éviter ce problème. Se conférer le mode **CFB** des chiffrements par blocs.
- Élément fréquent de conception : les **LFSR** (**Linear Feedback Shift Register**) = file de bits de longueur fixée, dont on extrait les bits à un bout en complétant à l'autre bout par le **XOR** entre des bits à emplacements fixes dans la file.

Conception des chiffrements par blocs (suite)

Chiffrement souvent en deux étapes

1. cadencement de clé (= "**key schedule**") : générer les sous-clés (ou clés de tour) $K_1, \cdots, K_r$ à partir de la clé K ;
2. chiffrement proprement dit, en r étapes (r = nombre de tours), typiquement, $x_0 = x$ et $x_{i+1} = F(K_{i+1}; x_i)$, donnant la sortie $y = x_r$ après r étapes, où F est la fonction de tour (doit être inversible).

▶ Le cadencement de clé est parfois très long (exemple : **Blowfish**).

▶ Cela peut être un avantage.

▶ Nombre de tours très variable selon le chiffrement et la complexité de F.

Réseaux de Feistel

Méthode simple pour éviter d'avoir à concevoir une fonction de tour automatiquement inversible.

- Diviser le bloc x_i de taille N en deux blocs de taille $\frac{N}{2}$, soit $x_i = x_i^L \| x_i^R$, et appliquer

$$x_{i+1}^L = x_i^R$$

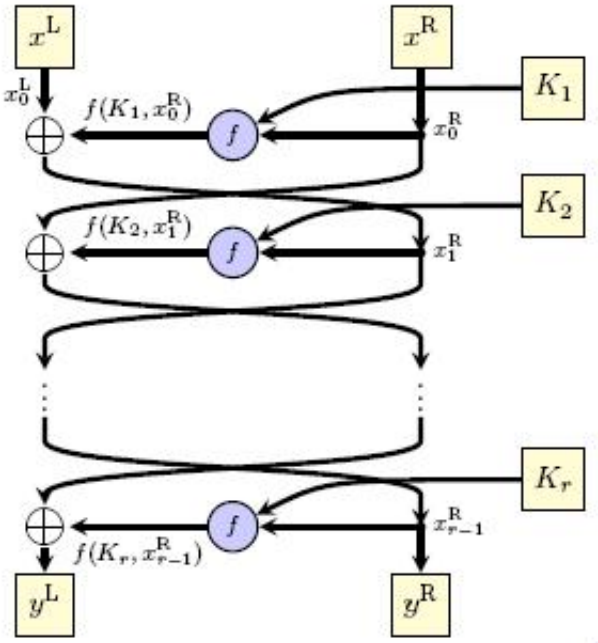
$$x_{i+1}^R = x_i^L \oplus f(K_{i+1}, x_i^R)$$

– Après la dernière étape, par convention, on refait l'échange des deux parties :

$$y^L = x_r^R = x_{r-1}^L \oplus f(K_r, x_{r-1}^R)$$

$$y^R = x_r^L = x_{r-1}^R$$

Avec cette convention, le chiffrement s'inverse en inversant simplement l'ordre des sous-clés $K_1, \cdots, K_r$.



2.4 Fonctions de hachage

- Une fonction de hachage est une fonction publique $H : \{0,1\}^* \to \{0,1\}^n$ telle que :
  - H transforme un message (binaire) de longueur quelconque en un message de longueur fixe ; (**Fonction de Compression**)
  - pour tout x, $H(x)$ est facile à calculer ; (**Facilement calculable**)
- Les images $H(x)$ sont appelées ***hache*** ou ***empreinte***.
- Pour résumer on retiendra que :
  - **Une fonction de hachage en cryptographie est une fonction publique à sens unique sans trappe** (donc facile à calculer et difficile à inverser pour tout le monde) $h : \mathcal{P} \to \mathcal{H}$, **qui transforme un message de longueur quelconque en un message de longueur fixe**.

- ♦ **De plus la probabilité, pour qu'il y ait une collusion doit être faible** [c'est-à-dire il doit être difficle de trouver $x \neq x'$ tels que $h(x) = h(x')$] ;

▶ On les utilise en cryptographie pour :

- ♦ fournir un condensé de taille fixe ;
- ♦ représenter précisément les données : la détection des changements dans le message est simplifiée.

▶ L'intérêt est que cela permet l'usage de la cryptographie asymétrique sans engendrer trop de ralentissement, mais également d'assurer la provenance d'un fichier ainsi que son intégrité.

▶ Dans la majorité des cas, elles seront utilisées pour créer une signature numérique.

▶ Enfin, la qualité principale de ces fonctions est que les algorithmes sont publics.

▶ On parle de "**haché**", de "**résumé**", ou de "**condensé**" pour nommer la caractéristique d'un texte ou de données uniques.

▶ La probabilité d'avoir deux messages avec le même haché doit être extrêmement faible.

▶ Le haché ne contient pas assez d'informations en lui-même pour permettre la reconstitution du texte original.

▶ L'objectif est d'être représentatif d'une donnée particulière et bien définie (en l'occurrence le message).

■ **Propriétés**

■ **Les fonctions de hachage possèdent de nombreuses propriétés :**

▶ Elles peuvent s'appliquer à n'importe quelle longueur de message M.

▶ Elles produisent un résultat de **longueur constante**.

▶ Il doit être facile de calculer $h = H(M)$ pour n'importe quel message M.

▶ Pour un h donné, il est impossible de trouver x tel que $H(x) = h$. On parle de **propriété à sens unique**.

▶ Pour un x donné, il est impossible de trouver y tel que $H(y) = H(x) \Longrightarrow$ **résistance faible de collision**.

▶ Il est impossible de trouver x, y tels que $H(y) = H(x) \Longrightarrow$ **résistance forte de collision**.

■ **Fonctions de hachage : Intégrité**

▶ Si M est un message alors pour garantir l'intégrité de M, on envoie ou stocke le couple $(M, H(M))$ où $H(M)$ est l'empreinte de M via une fonction de hachage H.

- ▶ Le message est considéré intègre s'il est bien accompagné par son empreinte qu'on ne peut falsifier.
- ▶ Les fonctions de hachage sont utilisées pour :
  - ♦ l'intégrité
  - ♦ construire des générateurs aléatoires cryptographiquement sûrs ;
  - ♦ pour la modélisation théorique des fonctions à sens unique tel que le modèle de l'oracle aléatoire.

■ **Fonction de hachage : Algorithmes**

■ Les fonctions de hachages comptent deux familles

♦ celles utilisant des clés :
  ▶▶ **MAC** (Message Authencation Code)

♦ celles n'utilisant pas de clés
  - **MD** (Message Digest) :
    - ▶▶ MD4, Rivest, 1990 ;
    - ▶▶ MD5, Rivest, 1992, empreinte sur 128 bits, RFC 1321
  - **SHA** (Secure Hash Algorithm) :
    - ▶▶ SHA (NIST-1993), FIPS 180, SHA1, empreinte sur 160 bits, FIPS 180-1
  - **SHS** (Secure Hash Standard) :
    - ▶▶ SHS (2001) FIPS 180-2 inclut SHA1 et SHA2 (SHA-256, SHA-384, SHA-512)
  - RIPEMD, 160

2.5 Schémas symétriques : DES et AES

2.5.1 Data Encryption Standard (DES)

Le chiffrement DES : Data Encryption Standard

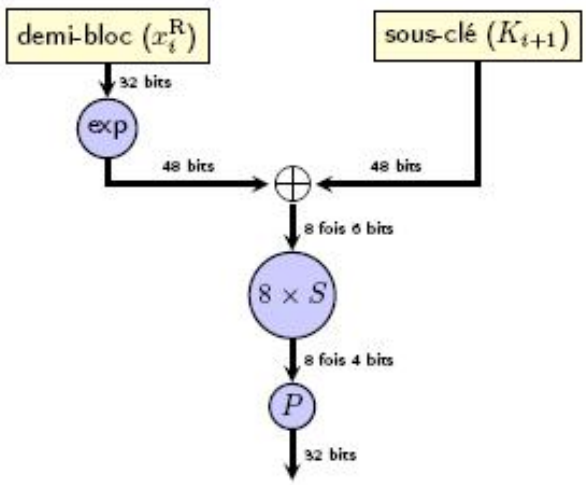
Standard NIST/FIPS publié en 1977, résultat d'un travail par **IBM**, avec des conseils de la NSA (**agence des services secrets américains**).

- **Réseau de Feistel :** prévu pour une implémentation en matériel ;
- **Taille de clé** = 56 bits (" +8 bits de parité ", ce qui ne veut rien dire) ;
- **Sous-clés** de 48 bits (cadencement trivial : ce sont des sous-ensembles des 56 bits de la clé) ;
- **Taille de bloc** = 64 bits (= deux demi-blocs de 32 bits) ;
- Utilise 8 "**boites** S" ("$S-boxes$") $6 \to 4$ fixées une fois pour toutes, c'est-à-dire 8 fonctions

prenant 6 bits en entrée (64 valeurs d'entrée possible) et renvoyant un nombre de 4 bits (16 valeurs de sortie possible). Données par un tableau 8×64. La conception de ces boîtes S n'était pas expliquée à l'époque ;

– Intérêt historique + valeur de référence. A ne pas utiliser actuellement !

Le chiffrement DES : fonction de Feistel



exp = expansion (duplique certains bits)

$8 \times S$ = les 8 boîtes S (apportent la confusion)

P = permutation des bits (assure la diffusion)

Le chiffrement DES : sécurité

– **Cassé par force brute en** 1998 (machine dédiée à l'énumération exhaustive des 256 **clés possibles**) ;

– **Résistant à la cryptanalyse différentielle**, (**Biham** et **Shamir**, 1990) : connue en fait d'**IBM** et de la **NSA**, mais gardée secrète ;

– **Non résistant à la cryptanalyse linéaire**, (**Matsui**, 1993) : attaque légèrement plus efficace qu'une attaque par force brute, mais réalisée seulement plus tard ;

– **De nos jours, DES ne peut plus du tout être considéré comme sûr**.

– On peut le rendre sûr en utilisant **triple-DES** (**3DES**) avec deux clés de 56 bits : $y = E_{K_1}(E_{K_2}^{-1}(E_{K_1}(x)))$ où E_K désigne **DES**.

– Ceci fournit 112 bits de clé $(K_1 \| K_2)$, mais très lent !

2.5.2 Advancing Encryption Standard (AES)

Le remplaçant de DES : AES

– **Concours pour remplacer DES comme standard, lancé en** 1997. Processus de sélection très ouvert. **Cinq finalistes** (**Rijndael, Serpent, Twofish, RC6 et MARS**).

– En 2000, **Rijndael** est choisi pour devenir **AES**. Standard publié en 2001 ;

– **Taille de clé** = 128, 192 ou 256 bits (des variantes sous le nom de Rijndael autorisent 160 ou 224 bits de clé) ;

- **Taille de bloc** = 128 bits (des variantes sous le nom de Rijndael autorisent 160, 192, 224 *ou* 256 bits) ;
- **Nombre de tour** = 10 à 14, **selon la taille de clé** (et **la taille de bloc**) ;
- **Travaille sur des octets** (**pas de permutation de bits**) ;
- **Unique boîte** S, **dont la conception est justifiée** ;
- **Plus rapide que DES** ;
- **Pas d'attaque connue plus rapide que par force brute**, jusqu'à 2011 : gain d'un facteur ~ 4.

Le corps d'AES

AES utilise deux opérations sur les octets : $\oplus$ est le **ou exclusif** (**bit-à-bit**), et $\circledast$ définie comme suit :

Définition mathématique

Si

$$A = a_7 \cdot 2^7 + \cdots + a_1 \cdot 2 + a_0$$

et

$$B = b_7 \cdot 2^7 + \cdots + b_1 \cdot 2 + b_0$$

sont deux octets, alors :

$$A \circledast B = c_7 \cdot 2^7 + \cdots + c_1 \cdot 2 + c_0$$

où le polynôme $c_7X^7 + \cdots + c_1X + c_0$ est le reste de la division euclidienne du polynôme $(a_7X^7 + \cdots + a_0) \times (b_7X^7 + \cdots + b_0)$ par le polynôme $X^8 + X^4 + X^3 + X + 1$, sur le corps à deux éléments $\mathbb{F}_2 = \{0, 1\}$ (où l'addition est modulo 2).

En plus condensé : $\mathbb{F}_{256} = \mathbb{F}_2[\mathrm{X}] = (\mathrm{X}^8 + \mathrm{X}^4 + \mathrm{X}^3 + \mathrm{X} + 1)$.

Le nombre 0x1b est $2^4 + 2^3 + 2 + 1$.

Muni de ces **deux opérations** de "**ou exclusif**" ($\oplus$) et "**multiplication de polynômes modulo** $X^8 + X^4 + X^3 + X + 1$" ($\circledast$) :

- L'ensemble $\mathbb{F}_{256}$ des octets est un corps à 256 éléments de caractéristique 2, c'est-à-dire que : $\oplus$ est associative et commutative, $x \oplus 0 = x$, $x \oplus x = 0$, $\circledast$ est distributive sur $\oplus$ et est associative et commutative, $x \circledast 1 = x$, et pour tout $x \neq 0$ il existe x' tel que $x \circledast x' = 1$;
- $2 \circledast x$ vaut soit $2x$ (si $x < 128$) soit $2x \oplus 0x1b$ ($x \geq 128$) ;
- $3 \circledast x = (2 \circledast x) \oplus x$ (par distributivité) ;
- $2^i \circledast x = 2 \circledast (2 \circledast (\cdots (2 \circledast x) \cdots))$;
- $2 \circledast 0x8d = 1$ (i.e., $0x8d = 141$ est l'inverse de 2).

Il existait d'autres choix possibles de polynômes à la place de $X^8 + X^4 + X^3 + X + 1$ (précisément 30 choix possibles).

Tous auraient donné un objet $\mathbb{F}_{256}$ abstraitement identique ("**isomorphe**"), mais codé différemment sur les octets.

"États" d'AES

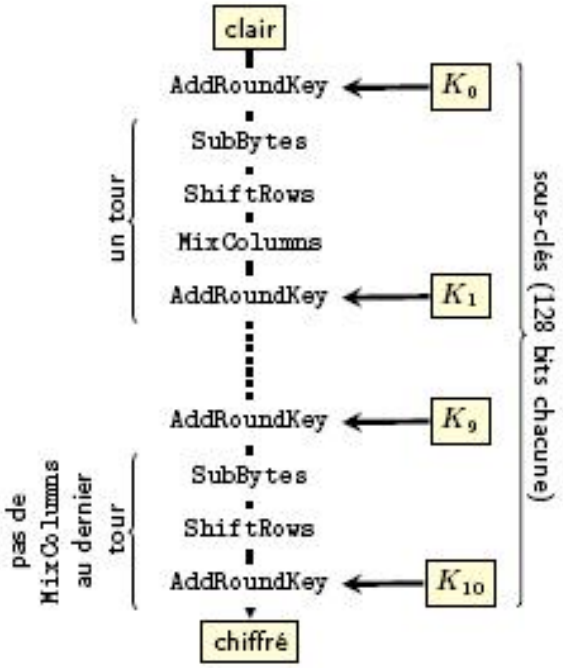
Un bloc **AES** est vu comme un tableau de 4 (**lignes**) $\times 4$ (**colonnes**) d'octets, soit 16 octets $=$ 128 bits, **appelé état** :

X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8
X_9	X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}	X_{16}

On définit 4 opérations sur cet "état" :

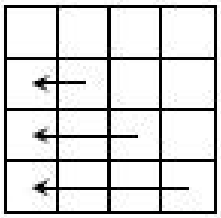
1. **AddRoundKey** (**ajout d'une sous-clé par** $\oplus$) ;
2. **SubBytes** (application boîte S) ;
3. **ShiftRows** (décalage des lignes) ;
4. **MixColumns** (transformation linéaire des colonnes).

Schéma général d'AES



La diffusion dans AES

– **ShiftRows** : applique un déclage cyclique des lignes de l'état de 0, 1, 2 *et* 3 cases vers la gauche : $x_{i,j} \leftarrow x_{i,(j+i)}\%4$;



– **MixColumns** : applique une transformation linéaire sur $\mathbb{F}_{256}$ de chaque colonne séparément :

$$\begin{cases} x_{i,0} \leftarrow 2 \circledast x_{i,0} \oplus 3 \circledast x_{i,1} \oplus x_{i,2} \oplus x_{i,3}, & ; \\ x_{i,1} \leftarrow x_{i,0} \oplus 2 \circledast x_{i,1} \oplus 3 \circledast x_{i,2} \oplus x_{i,3}, & ; \\ x_{i,1} \leftarrow x_{i,0} \oplus x_{i,1} \oplus 2 \circledast x_{i,2} \oplus 3 \circledast x_{i,3}, & ; \\ x_{i,3} \leftarrow 3 \circledast x_{i,0} \oplus x_{i,1} \oplus x_{i,2} \oplus 2 \circledast x_{i,3}, & . \end{cases}$$

Ensemble, ces deux opérations apportent la diffusion.

La boîte S de AES

Chaque octet de l'état indépendamment : $x_{i,j} \leftarrow S[x_{i,j}]$. Cette boîte S est indépendante aussi bien des clés que des entrées, et toujours la même.

La boîte S est définie comme la composée de deux fonctions sur les octets :

– **La fonction "inverse"** $x \mapsto x^{-1}$, $0 \mapsto 0$ dans $\mathbb{F}_{256}$ (il se trouve que c'est aussi $x \mapsto x^{254}$). **Son rôle est de maximiser la confusion** ;

– Une **fonction affine** sur l'octet vu comme $(\mathbb{F}_2)^8$ (chaque bit est remplacé par le ou exclusif de cinq autres bits et d'un bit constant). **Le rôle de cette fonction affine est de "casser" la structure trop algébrique de** $\mathbb{F}_{256}$ tout en préservant les propriétés de **confusion** de la boîte S.

Chapitre 3

La cryptographie à clé publique

3.1 Concept des systèmes non symétriques

- Dans le cas des systèmes symétriques, on utilise une même clé pour le chiffrement et le déchiffrement.
- Le problème repose dans la transmission de la clé : il faut une clé par destinataire.
- Dans le cas des systèmes asymétriques, chaque personne possède 2 clés distinctes (une privée, une publique) avec impossibilité de déduire la clé privée à partir de la clé publique.
- De ce fait, il est possible de distribuer librement cette dernière.

■ On peut classer l'utilisation des algorithmes à clé publique en 3 catégories :

- **Chiffrement/déchiffrement :** cela fournit le secret.
- **Signatures numériques :** cela fournit l'authentification.
- **Échange de clés** (ou des clefs de session).

■ Quelques algorithmes conviennent pour tous les usages, d'autres sont spécifiques à un d'eux.

■ La sécurité de tels systèmes repose sur des problèmes calculatoires :

- **RSA :** factorisation de grands entiers.
- **ElGamal :** logarithme discret.
- **Merkle-Hellman :** problème du sac à dos (knapsacks).
- **NTRU :** problème sur les réseaux arithmétiques (SVP).

■ La recherche des clés par force brute est toujours théoriquement possible mais les clefs utilisées sont trop grandes (> 1024 *bits*).

■ La sécurité se fonde sur une assez grande différence en termes de difficulté entre les problèmes faciles (déchiffrement) et difficiles (décryptement) :

- généralement le problème difficile est connu, mais il est trop complexe à résoudre en pratique.
- La génération des clés exige l'utilisation de très grands nombres.

■ En conséquence, ce type de chiffrement est lent si on le compare aux chiffrements symétriques.

3.2 Limites des systèmes asymétriques

- Pour les systèmes à clé publique, le problème du canal sûr de communication ne se pose pas, mais néanmoins deux autres problèmes sont soulevés.
  - **Confidentialité de la clé privée** (gardée sur un ordinateur ou une clé USB appelée Token) ;
  - **Intégrité de la clé publique** (publiée dans ce qu'on appel un annuaire) ;
- **Pour garantir l'intégrité de la clé publique** de Bob, les différentes parties en communication conviennent de s'en référer à un Tiers de Confiance appelé **Autorité de Confiance** (AC) qui, si elle est sollicitée peut garantir si une clé donnée appartient bien à Bob et si elle n'est pas compromise.
- L'utilisation des systèmes à clé publique :
  - **à grande échelle**, nécessite des systèmes complexes appelés PKI (Public Key Infrastructure).
  - **à petite échelle**, nécessite une chaine de confiance à défaut d'un PKI : c'est le cas de l'Utilisation de Gnu-PG (voir TP) pour le chiffrement des mails sur internet
- Les algorithmes à clé publique utilisent des fonctions à sens unique dont la sécurité repose sur l'existence de problèmes difficiles comme la factorisation.
- Tous les algorithmes qui résolvent ces problèmes difficiles sont au moins **sous-exponentiels** et on montre que les puissances de calculs actuels et les attaques connues imposent de choisir des tailles de clés de plus de 1000 bits (Par exemple le module **RSA** est à 2048 bits actuellement).
- L'utilisation de nombres très grands par les algorithmes à clés publiques ainsi que d'opérations couteuses comme l'exponentiation font ques ces derniers ne sont pas adaptés au chiffrement de données de grande taille.
- Ainsi les algorithmes à clé publique sont essentiellement utilisés :
  - **pour chiffrer des clés symétriques** qui à leur tour seront utilisées pour chiffrer les

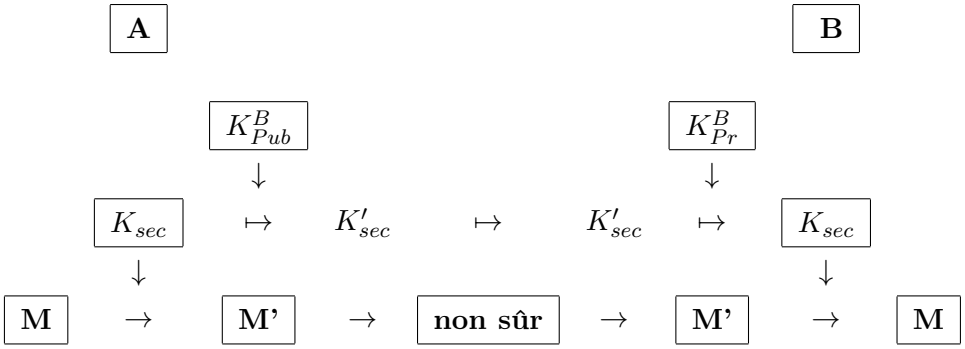
données pour la **confidentialité** (système hybride : voir point suivant) ;

- **pour signer des messages** pour garantir la **non répudiation** ;
- **pour construire des protocoles (authentification, identification, partage de secrets,...)** ;

3.3 Schéma hybride

- Si Alice veut envoyer un message chiffré à Bob, elle génère une clé secrète K_{sec} et l'utilise pour chiffrer le message clair M en M' puis chiffre la clé K_{sec} en K'_{sec} en utilisant la clé publique de Bob K^B_{Pub}. Enfin elle envoie le couple (M', K'_{sec}) à Bob à travers un canal non nécessairement sûr.
- A l'arrivée, Bob déchiffre K'_{sec} en K_{sec} en utilisant sa clé privée K^B_{Pr}, puis déchiffre M' en M en utilisant K_{sec}

Schéma d'un système hybride



3.4 Signature numérique

- Nous avons jusqu'ici regardé l'authentification de messages.
- Mais nous n'avons pas abordé les questions de manque de confiance.
- En effet, l'authentification des messages protège les deux entités communicantes d'un intrus.
- Cependant, cela ne les protège en rien l'un de l'autre.
- En cas de conflit, rien n'empêche la **répudiation** du ou des messages.
- Deux exemples illustrent ce principe :
  - L'entité $\mathcal{A}$ pourrait forger un message donné et dire qu'il vient de $\mathcal{B}$. Il lui suffit pour cela de créer un message et lui appliquer un MAC en utilisant la clé qu'ils ont en commun.
  - Le cas inverse peut également exister. $\mathcal{B}$ peut nier le fait d'avoir envoyé un message, pour la simple raison que la construction expliquée au point précédent est possible.

- ♦ Il est donc impossible de prouver qu'un tel message a été ou non envoyé par $\mathcal{B}$ plutôt que par $\mathcal{A}$.

► C'est ici qu'interviennent les **signatures numériques**.

► Elles permettent notamment :

- ♦ de vérifier l'auteur ;
- ♦ la date et l'heure de la signature ;
- ♦ d'authentifier le contenu d'un message ;
- ♦ et peuvent être vérifiées par des tiers pour résoudre des conflits.

■ Signature numérique : Définition et Propriétés

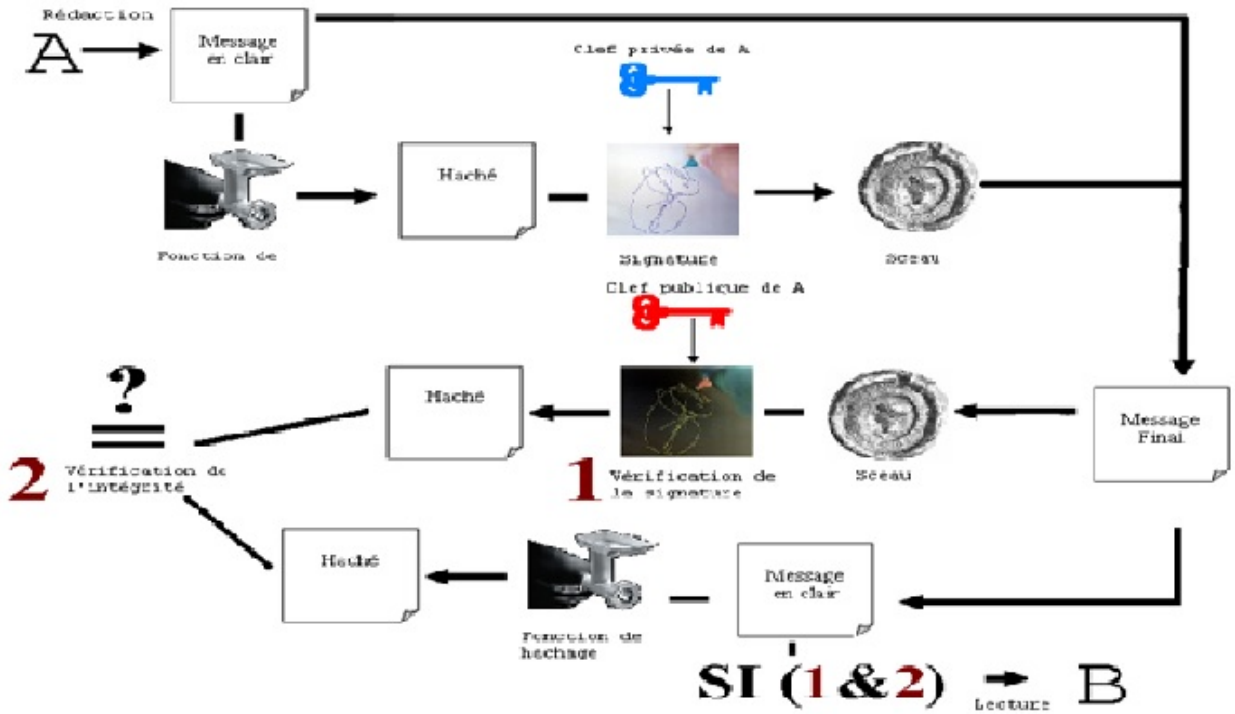
► Une signature (digitale-manuelle ou numérique-cryptographique) est un procédé, qui, appliqué à un message, garantit la **non répudiation** par le **signataire** et donc réalise les deux objectifs suivants :

1. **identification unique du signataire**,
2. **et preuve d'accord sur le contenu du document**.

► Elle doit posséder les propriétés suivantes :

1. unique : dépendre du message signé (employer une information unique propre à l'expéditeur pour empêcher la contrefaçon et le démenti).
2. impossible à usurper : c'est à dire être mathématiquement infaisable à forger (par construction de nouveaux messages pour une signature numérique existante, ou par construction d'une signature numérique frauduleuse pour un message donné).
3. impossible à répudier par son auteur.
4. facile à vérifier (reconnaitre) par un tiers.
5. être relativement facile à générer (produire) et à stocker. .

■ Signature numérique : Schéma



■ **Signature numérique et Signature manuelle**

■ On a vu les 5 points communs entre les signatires manuelle et numérique dans la première définition. Ici on regarde les différences.

■ **Signature manuelle**

1. Associé physiquement au document signé ;
2. Identique pour tous les documents venant d'un même signataire ;
3. Habituellement à la dernière page.

■ **Signature numérique**

1. Peut être stockée et envoyée indépendamment du document signé ;
2. Fonction du document même si le signataire signe avec la même clé privée
3. Couvre l'entièreté du document (dépend de tout le message)

■ **Signature numérique : Modélisation (première version)**

Un système de signature est composé d'un quintuplet $(\mathcal{P}, \mathcal{S}, S_{k'}, V_k, \mathcal{K})$ où :

1. $\mathcal{P}$ est un ensemble appelé espace des textes clairs ;
2. $\mathcal{S}$ est un ensemble appelé espace des signatures ;
3. $S_{k'} : \mathcal{P} \to \mathcal{S}$ est une fonction injective dite fonction de signature (non nécessairement bijective) qui dépend d'un paramètre k' appelé clé privée.

4. $V_k : \mathcal{P}x\mathcal{S} \rightarrow \{vrai, faux\}$ est la fonction de vérification de signature binaire telle que $V_k(m, s) =$ vrai si et seulement si $S_{k'}(m) = s$ (dépendant de la clé publique k) .

5. $\mathcal{K}$ l'ensemble des paramètres utilisés est l'espace des clés.

■ **Signature numérique : Problème d'intégrité**

■ **La signature, telle que présentée jusqu'à présent, ne garantit pas l'intégrité (qu'elle soit manuelle ou numérique).**

- ▶ Dans le cas manuelle c'est l'observation et l'analyse du document qui permet de croire ou de s'assurer que le texte n'a pas été modifié.
- ▶ Dans le cas numérique ; **Pour résoudre ce problème** :
  - ♦ Le massagge est**haché** avant d'être signé pour garantir l'intégrité.
  - ♦ La fonction de hachage ne doit pas être **homomorphique**.

■ **Signature numérique : Définition**

■ Données ajoutées à une unité de données ou transformation cryptographie d'une unité de données, permettant à un destinataire de**vérifier la source** et l'**intégrité** de l'unité de données, garantissant la **non répudiation** et protégeant contre la contrefaçon.

■ **Une signature numérique doit fournir les services :**

- ▶ d'authentification (de l'origine des données) ;
- ▶ de leur intégrité ;
- ▶ et de non répudiation (pour le signataire).

■ **Signature numérique : Modélisation (deuxième version)**

Un système de **signature avec appendice** est composé d'un 6-tuplet $(\mathcal{P}, \mathcal{H}, \mathcal{S}, S_{k'}, V_k, \mathcal{K})$ où :

1. $\mathcal{P}$ est un ensemble appelé espace des textes clairs ;
2. $\mathcal{S}$ est un ensemble appelé espace des signatures ;
3. $h : \mathcal{P} \rightarrow \mathcal{H}$ une fonction de hachage ;
4. $S_{k'} : \mathcal{H} \rightarrow \mathcal{S}$ est une fonction injective dite fonction de signature (non nécessairement bijective) qui dépend d'un paramètre k' appelé clé privée ;
5. $V_k : \mathcal{P}x\mathcal{S} \rightarrow \{vrai, faux\}$ est la fonction de vérification de signature binaire telle que $V_k(m, s) =$ vrai si et seulement si $S_{k'}\big(h(m)\big) = s$ (dépendant de la clé publique k) ;
6. $\mathcal{K}$ l'ensemble des paramétres utilisés est l'espace des clés.

3.5 Diffie-Hellman, RSA et ElGamal

3.5.1 Protocole d'échange de clés Diffie-Hellman

Whitfield Diffie
(1944-...)

Martin Hellman
(1945-...)

- L'inconvénient majeur des systèmes à clés publiques est qu'ils sont beaucoup plus lents que les systèmes à clé secrète.
- Par exemple, RSA utilisé avec un nombre premier de 512 bits (on dit RSA-512) chiffre 600 Ko par seconde, c'est 1500 fois plus lent que l'implémentation la plus rapide de DES qui permet de chiffrer 1 Go par seconde.
- Donc en pratique on utilise souvent les systèmes à clé publique pour se transmettre une clé secrète qu'on utilise ensuite pour chiffrer les messages.
- Historiquement, la naissance de la cryptologie à clé publique remonte à l'invention d'une méthode permettant à deux personnes distantes de se mettre d'accord sur un nombre (appelé à devenir une clé secrète) sans que quiconque écoutant l'intégralité de la discussion ne puisse calculer ce nombre.
- Diffie et Hellmann ont proposé une méthode pour y parvenir, et ceci en 1976.
- **Problème** : **Alice** et **Bob** veulent générer un **secret commun** (***aléatoire***).
- Ils disposent d'un canal de communication sur lequel **Eve** peut écouter mais ne peut rien altérer.

1. **Alice** et **Bob** fixent publiquement un n premier et un entier g primitif modulo n (i.e., g engendre $(\mathbb{Z}/n\mathbb{Z})^\times$).
2. **Alice** choisit a aléatoire entre 0 et $n-2$, le garde secret, et calcule $A = g^a \pmod n$, qu'elle envoie à **Bob**.
3. **Bob** choisit b aléatoire entre 0 et $n-2$, le garde secret, et calcule $B = g^b \pmod n$, qu'il envoie à **Alice**.
4. **Alice** calcule $B^a = g^{ab}$. **Bob** calcule $A^b = g^{ab}$. Ce nombre commun (modulo n) est leur secret commun.

5. **Eve** connaît $A = g^a$ et $B = g^b$ en écoutant la communication et g qui est public, mais ne parvient pas à calculer a, b ou bien g^{ab}.

3.5.2 Rivest Shamir Adleman (RSA)


Ron Rivest
(1947-...)


Adi Shamir
(1952-...)


Leonard Adleman
(1945-...)

RSA : Présentation

- Du nom de ses inventeurs (Rivest, Shamir et Adleman), RSA est le premier algorithme à clé publique inventé en 1978.
- Le système RSA repose sur la difficulté de factoriser de grands nombres premiers.
- Pour assurer la sécurité les nombres premiers choisis doivent être suffisamment grands.

RSA : Génération des clés

- Choisir deux entiers premiers p et q assez grands ;
- Calculer le modulo $n = pq$.
- Calculer l'indicateur d'Euler $\varphi(n) = (p-1)(q-1)$. $\varphi(n)$ est le nombre d'éléments inversibles du groupe $(\frac{\mathbb{Z}}{n\mathbb{Z}})^*$;
- Choisir un entier e tel qu'il soit plus petit que $\varphi(n)$ et $\text{pgcd}(e, \varphi(n)) = 1$;
- Calculer l'entier d tel que $ed = 1 \mod \varphi(n)$ (avec l'algorithme étendu d'Euclide).
- Clé publique (e, n) : (généralement $e = 2^{16} + 1$ est fixe) ;
- Clé privée (d, n).

p, q et $\varphi(n)$ doivent rester secret.

RSA : Chiffrement

- Soit $M \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ le message à chifferer ;
- Prendre la clé publique (e, n) ;
- Calculer le chiffré $C = M^e \mod n$

■ **RSA : Déchiffrement**

- Soit le chiffré $C \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ reçu ;
- Prendre la clé publique d ;
- Calculer le déchiffré $M' = C^d \mod n$

■ **RSA : Signature**

- Soit $M \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ le message à signer ;
- Prendre la clé privée d ;
- Calculer la signature $S = M^d \mod n$

■ **RSA : Vérification**

- Soit $S \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ la signature du message M reçue ;
- Prendre la clé publique (e, n) ;
- Calculer $M' = S^e \mod n$ puis comparer M' et M.
- La signature est valide s'il y a égalité.

■ **RSA : Forces**

On peut citer entre autres

- Chiffrement et signature en même temps ;
- Simplicité et efficience ;
- Capacité d'adaptation : toutes les attaques connues jusqu'ici ont pu être contournées :
  - ♦ soit en utilisant OAEP pour le chiffrement et PSS pour la signature pour les rendre probabilistes ;
  - ♦ soit en modifiant la taille des paramètres : n, p, q, e et d ;

Récemment il a été prouvé qu'un nouveau algorithme **Post-Quantum RSA** peut résister à l'**ordinateur quantique**.

■ **RSA : Faiblesses**

On peut citer entre autres

- Son déterminisme (cause de beaucoup de vulnérabilités) ;
- Son algorithme de génération de clés :$ed = 1 \mod \varphi(n)$;
- Ressemblance des fonctions de déchiffrement et signature ;

- La lenteur des opérations de déchiffrement et de signature ;
- La difficulté qu'on a à traiter RSA sur différents groupes finis ;
- Sa propriété d'homomorphie ;

3.5.3 Le schéma d'El Gamal

Taher Elgamal

ELGAMAL : Présentation

- Le chiffrement d'El Gamal est un système de cryptage à clé publique inventé en 1984 par l'égyptien Taher ElGamal et utilisé pour chiffrer mais aussi pour signer des messages.
- Ce système est assez peu utilisé en tant qu'algorithme de chiffrement pur, mais plutôt dans les systèmes de signatures ;
- Il est d'ailleurs à l'origine du DSS (Digital Signature Standard), devenu une norme fédérale en 1994.
- Ce chiffrement tire son nom, comme RSA, du nom de son créateur.

ELGAMAL : Génération des clés

- Choisir un nombre premier très grand p ;
- Choisir g un générateur de $(\frac{\mathbb{Z}}{p\mathbb{Z}})^*$ d'ordre $n = p - 1$;
- Choisir un entier aléatoire a dans $[2, n]$;
- Calculer $h = g^a \mod p$;
- La clé publique est (g, h, p) ;
- La clé privée est a.

ELGAMAL : Chiffrement

- Soit m le message clair ;

- Choisir un nombre aléatoire k assez grand ;
- Prendre la clé publique est (g, h, p) ;
- Pour chiffrer le message, calculer :
  - $m_1 = g^k \mod p$;
  - $m_2 = m.h^k \mod p$.
- Le chiffré est $c = (m_1, m_2)$.

ELGAMAL : Déchiffrement

- Soit le chiffré $c = (m_1, m_2)$;
- Prendre la clé privée a ;
- Calculer $m' = m_1^{p-1-a} \cdot m_2 = g^{-ka} \cdot g^{ka} \cdot m = m$.

ELGAMAL : Signature

Pour signer un message

- Alice souhaite signer un document M ;
- Elle choisit au hasard un entier $k \in [1, p-2]$; tel que $\text{pgcd}(k, p-1) = 1 \iff k^{-1} \in \mathbb{Z}_{P-1}$ existe ;
- Signature de M : $s(M) = (r, s)$ avec
  - $r = g^k \mod p$;
  - $s = k^{-1}(M - a.r) \mod (p-1)$;
- Le document signé est alors $[M, s(M)]$.

ELGAMAL : Vérification

La signature est dite valide si

- $0 < r < p$;
- $h^r r^s \equiv g^M (\mod p)$

ELGAMAL : Forces

- La force du chiffrement d'El Gamal est son usage d'un problème mathématique connu comme difficile à résoudre, celui du logarithme discret.
- En effet, le calcul d'un logarithme modulo n est possible mais plus n est grand et plus le calcul sera long, pouvant atteindre plusieurs milliers d'années.
- On définit le chiffrement dans le corps fini $\mathbb{Z}_p$, avec p un nombre premier.

- Le groupe $\mathbb{Z}_p^*$ est cyclique et on appelle ses générateurs racines primitives modulo p.

ELGAMAL : Faiblesses

- Du nom de son créateur, c'est un autre algorithme basé sur le protocole Diffie Helmann qu'on retrouve notamment dans les dernières versions de PGP ;
- Algorithme qui n'est pas sous brevet.
- A noter que le message chiffré est deux fois plus long que le message d'origine : ce qui peut engendrer quelques complications en terme de stockage ou de communication.
- Certaines implémentations d'ElGamal génèrent parfois des risques quant à l'exposition possible de la clé privée.
- Le problème du logarithme discret (DLP) est cassé par les ordinateurs quantiques.

3.6 Cryptanalyse de RSA

3.6.1 Cryptanalyse de RSA connaissant $\varphi(N)$

Proposition

Soit N un module **RSA**. Si on connait $\varphi(N)$, alors on peut factoriser N.

Démonstration

Supposons que $\varphi(N)$ est connu. Ainsi, on dispose d'un système de deux équations en p et q :

$$\begin{cases} pp = N \\ p + q = N + 1 - \varphi(N), \end{cases}$$

qui donnent l'équation en p : $p^2 - (N + 1 - \varphi(N))p + N = 0$

On obtient ainsi

$$p = \frac{N + 1 - \varphi(N) + \sqrt{(N + 1 - \varphi(N))^2 - 4N}}{2}$$

$$q = \frac{N + 1 - \varphi(N) - \sqrt{(N + 1 - \varphi(N))^2 - 4N}}{2}$$

3.6.2 Utilisation du même module et deux exposants différents

Proposition

Soit N un module **RSA**. Soient e_1 et e_2 deux exposants premiers entre eux. Si un message clair M est chiffré avec e_1 et e_2, alors on peut calculer M.

■ Démonstration

Soient C_1 et C_2 deux messages chiffrés par

$$C_1 \equiv M^{e_1} \mod N,$$
$$C_2 = M^{e_2} \mod N,$$

et supposons que C_1 et C_2 sont rendus publiques.

Si $pgcd(e_1, e_2) = 1$, alors il existe deux entiers x_1 et x_2 tels que $x_i < \frac{1}{2}e_i$ et vérifiant $e_1x_1 - e_2x_2 = \pm 1$.

Ces deux entiers peuvent être déterminer par l'**algorithme d'Euclide**.

D'autre part, ils vérifient

$$\left|\frac{e_1}{e_2} - \frac{x_2}{x_1}\right| = \frac{|e_1x_1 - e_2x_2|}{x_1e_2} = \frac{1}{x_1e_2} < \frac{1}{2x_1^2}$$

Ainsi x_1 et x_2 peuvent être déterminés comme dénominateur et numérateur de l'une des convergentes de $\frac{e_1}{e_2}$. Si $e_1x_1 - e_2x_2 = 1$, on obtient alors

$$C_1^{x_1}C_2^{-x_2} = M^{e_1x_1}M^{-e_2x_2} = M^{e_1x_1 - e_2x_2} = M \mod N,$$

ce qui donne le message M. Si $e_1x_1 - e_2x_2 = -1$, on calcule $C_1^{-x_1}C_2^{x_2}$ et on obtient le même résultat.

3.6.3 Utilisation de modules différents pour le même message

■ Proposition

Soit $k = 2$ un nombre entier. On considère k modules **RSA** N_i avec $1 \leq i \leq k$. Soient C_i, $1 \leq i \leq k$, des messages chiffrés du même message clair M à l'aide du même exposant e. Si $M^e < \sum_{i=1}^{k} N_i$ alors on peut déterminer le message clair M sans factoriser les modules.

■ Démonstration

Supposons que le même message clair M est chiffré k fois par $C_i = M^e \mod N_i$, pour $i = 1, \cdots, k$. Soit $N = \sum_{i=1}^{k} N_i$.

On peut appliquer le **Théorème des restes chinois** pour résoudre le système formé des k équations $x = C_i \mod N_i$, avec $x < N$.

Si on suppose que $M^e < N$, alors $x = M^e$ en tant que nombres entiers.

Ainsi $M = x^{\frac{1}{e}}$, ce qui donne le message clair M.

3.6.4 Cryptanalyse de RSA si $|p-q| < cN^{1/4}$: Méthode de Fermat

■ Dans cette partie, on suppose que les nombres premiers p et q qui forment le module **RSA** $N = pq$ sont très proches, plus précisément $|p-q| < cN^{1/4}$ où c est une constante fixe, assez petite.

■ **Proposition**

Soit $N = pq$ un module **RSA** où les nombres premiers p et q vérifient $|p-q| < cN^{1/4}$ où c est une constante assez petite. Alors on peut factoriser N en temps polynômial dépendant de c.

■ **Démonstration**

La méthode de **Fermat** consiste en la recherche de deux nombres entiers x et y tels que

$$4N = x^2 - y^2 = (x+y)(x-y).$$

Si en plus $x - y \neq 2$, alors on obtient la factorisation de N en posant

$$p = \frac{x+y}{2},\ q = \frac{x-y}{2}.$$

Pour déterminer x et y, on prend pour x les valeurs $x_0 = \lfloor 2\sqrt{N} \rfloor,\ x_1 = \lfloor 2\sqrt{N} \rfloor + 1,\ x_2 = \lfloor 2\sqrt{N} \rfloor + 2, \cdots$ et on teste si $4N - x^2$ est un carré parfait.

On désigne alors par k le nombre entier pour lequel $x_k = \lfloor 2\sqrt{N} \rfloor + k$ donne la factorisation de N.

Alors $x_k = p + q$ et on a, en supposant que $|p-q| < cN^{1/4}$:

$$\begin{aligned} k = x_k - \lfloor 2\sqrt{N} \rfloor &= p + q - \lfloor 2\sqrt{N} \rfloor \\ &< p + q - 2\sqrt{N} + 1 \\ = \frac{(p+q)^2 - 4N}{p+q+2\sqrt{N}} + 1 &= \frac{(p-q)^2}{p+q+2\sqrt{N}} + 1 \\ &< \frac{c^2\sqrt{N}}{2\sqrt{N}} + 1 \\ &< \frac{c^2}{2} + 1 \end{aligned}$$

Il en résulte que le nombre de tests est assez petits si c est une constante qui ne dépend pas de N.

Chapitre 4

Applications

4.1 Chiffrement/Signature : RSA et El Gamal sous Java/Maple

- Chiffrement/Signature : RSA sous Java/Maple
- Chiffrement/Signature : El Gamal sous Java/Maple

4.2 OpenSSL : Chiffrement/Signature, Certificats

- Chiffrement Symétrique
- Chiffrement Symétrique
- Signature Numérique
- Certificat Numérique

4.3 GnuPG : Chiffrement/Signature des mails

- GnuPG : logiciel de chiffrement des mails
- Thunderbird : client de messagerie
- Enigmail : pluging liant GnuPG et Thunderbird

4.4 KeyTools : Gestion des clés et certificats

- Création d'un KeyStore de type JKS ou JCEKS
- Génération des clés et certificats
- Exportation/Importation de clés et certificats
- Chiffrement / Signature de fichiers