

Objectifs	Critères
Expliquer les termes de bases liés à la cryptographie (codage, chiffrement, transposition)	> Peut définir chiffrement, codage et cryptographie > Peut donner 2 exemples de codage simple : transposition, substitution
Expliquer une fonction de hachage	> Peut citer au moins 3 algos de hachage (MD4, MD5, SHA, NIST) > Peut expliquer comment fonctionne le MD5
Appliquer un système de codage simple	> Peut expliquer un codage par substitution et par transposition (ROT13, César, Hill)

Objectifs	Critères
Décomposer un algorithme de cryptographie symétrique utilisant la notion de clé secrète	> Explique le principe d'un chiffrement symétrique (RC4, AES, DES) > Expliciter les différentes briques d'un algorithme de chiffrement symétrique
Expliquer la différence entre le principe de Kerckhoffs et celui de l'obscurité)	> « B » + colliger les 2 principes et choisir celui qui s'adapte le mieux au contexte

**Cryptologie** : Cela signifie la "science du secret". La Cryptologie se partage entre la cryptographie et la Cryptanalyse ;

**Cryptographie** : E` étude des mécanismes destinés a` Assurer - entre autres - la confidentialité `e des Communications ;

**Cryptanalyse** : Son but est de d'déjouer les protections Cryptographiques mises en place.

- 1 **Cryptographie** : Étude et conception des procédés de chiffrement des informations ;
- 2 **Cryptanalyse** : Analyse des textes chiffrés pour retrouver des informations dissimulées, Analyse des procédés de chiffrement afin d'en découvrir les failles de sécurité.
- 1 **Texte (ou message) clair** : Information qu'Alice souhaite transmettre à Bob. Par exemple, un texte en français ou des données numériques ;
- 2 **Chiffrement** : Processus de transformation d'un message clair  $M$  de façon à le rendre incompréhensible (sauf aux interlocuteurs légitimes). Il est basé sur une **fonction de chiffrement**  $E$  qui permet de générer un **message chiffré**  $C := E(M)$  ;
- 3 **Déchiffrement** : Processus de reconstruction du message clair à partir du message chiffré. Il est basé sur une **fonction de déchiffrement**  $D$  telle que si  $C$  est le message chiffré correspondant au message clair  $M$ , alors  $D(C) = M$ .

Un **système cryptographique** ou **procédé** (ou **algorithme**) de chiffrement est la donnée

- 1 d'un espace des clefs  $\mathcal{K}$  ;
- 2 de fonctions de chiffrement et de déchiffrement paramétrées par des éléments de  $\mathcal{K}$  et qui vérifient la **propriété de déchiffrement**.

¶ La **cryptographie** est une des disciplines de la cryptologie s'attachant à protéger des messages (assurant confidentialité et/ou authenticité) en s'aidant souvent de *secrets* ou *clés*. Le mot cryptographie découle des mots grecs "krypto" (je cache) et "graphe" (le document)

¶ La **cryptologie**, étymologiquement la science du secret, ne peut être vraiment considérée comme une science que depuis peu de temps. Cette science englobe la cryptographie, l'écriture secrète et la cryptanalyse, l'analyse de cette dernière. On peut dire que la cryptologie est un art ancien et une science nouvelle

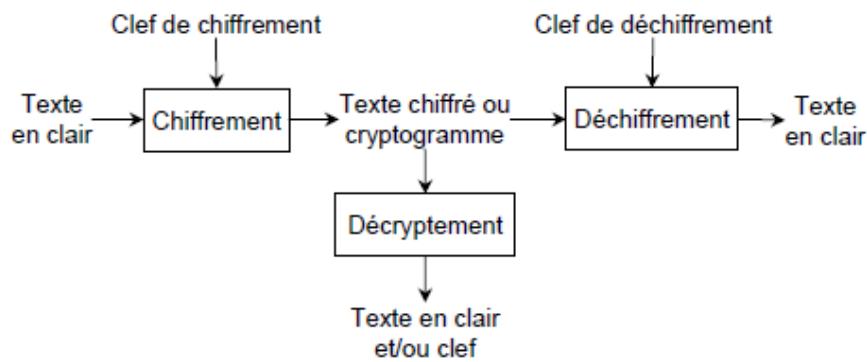
¶ L'**objectif fondamental** de la cryptographie est de permettre à deux personnes de communiquer au travers d'un canal peu sûr (téléphone, réseau informatique ou autre) sans qu'un opposant puisse comprendre ce qui est échangé.

La *cryptographie* est la science qui utilise les mathématiques pour le cryptage et le décryptage de données.

Elle vous permet ainsi de stocker des informations confidentielles ou de les transmettre sur des réseaux non sécurisés (tels que l'Internet), afin qu'aucune personne autre que le destinataire ne puisse les lire.

Alors que la cryptographie consiste à sécuriser les données, la *cryptanalyse* est l'étude des informations cryptées, afin d'en découvrir le secret. La cryptanalyse classique implique une combinaison intéressante de raisonnement analytique, d'application d'outils mathématiques, de recherche de modèle, de patience, de détermination et de chance. Ces cryptanalystes sont également appelés des *pirates*.

La *cryptologie* englobe la cryptographie et la cryptanalyse.



La **cryptologie** est une science mathématique qui comporte deux branches : la **cryptographie** et la **cryptanalyse**.

La **cryptographie** traditionnelle est l'étude des méthodes permettant de transmettre des données de manière confidentielle. Afin de protéger un message, on lui applique une transformation qui le rend incompréhensible ; c'est ce qu'on appelle le **chiffrement**, qui, à partir d'un **texte en clair**, donne un **texte chiffré ou cryptogramme**. Inversement, le **déchiffrement** est l'action qui permet de reconstruire le **texte en clair** à partir du **texte chiffré**. Dans la cryptographie moderne, les transformations en question sont des fonctions mathématiques, appelées **algorithmes cryptographiques**, qui dépendent d'un paramètre appelé **clef**.

La **cryptanalyse**, à l'inverse, est l'étude des procédés cryptographiques dans le but de trouver des faiblesses et, en particulier, de pouvoir décrypter des textes chiffrés. Le **décryptement** est l'action consistant à retrouver le **texte en clair** sans connaître la **clef de déchiffrement**.

**Note :** Les termes "cryptage" et "crypter" sont des anglicismes, dérivés de l'anglais *to encrypt*, souvent employés incorrectement à la place de **chiffrement** et **chiffrer**. En toute rigueur, ces termes n'existent pas dans la langue française. Si le "cryptage" existait, il pourrait être défini comme l'inverse du décryptage, c'est-à-dire comme l'action consistant à obtenir un **texte chiffré** à partir d'un **texte en clair** sans connaître la **clef**. Un exemple concret pourrait être de signer un **texte choisi** en reproduisant un **chiffrement** avec la **clef privée** de la victime. Mais on préfère parler dans ce cas de **contrefaçon** et de l'action de **forger une signature**.

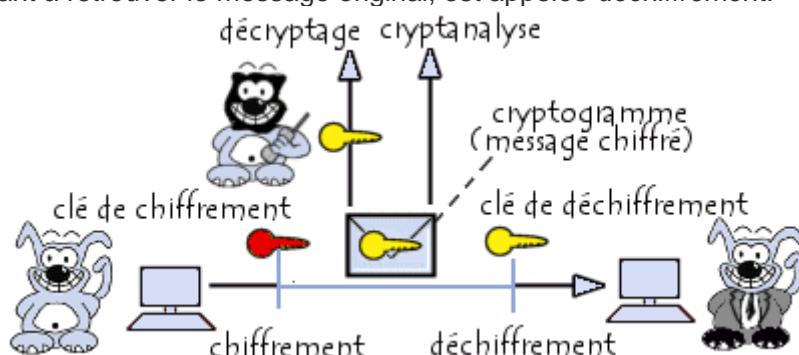
- **Décrypter ou casser un code** c'est parvenir au **texte en clair** sans posséder au départ les règles ou documents nécessaires au **chiffrement**.

Le mot **cryptographie** est un terme générique désignant l'ensemble des techniques permettant de **chiffrer** des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Le verbe **crypter** est parfois utilisé mais on lui préférera le verbe **chiffrer**.

La cryptologie est essentiellement basée sur l'arithmétique : Il s'agit dans le cas d'un texte de transformer les lettres qui composent le message en une succession de chiffres (sous forme de bits dans le cas de l'informatique car le fonctionnement des ordinateurs est basé sur le [binaire](#)), puis ensuite de faire des calculs sur ces chiffres pour :

- d'une part les modifier de telle façon à les rendre incompréhensibles. Le résultat de cette modification (le message chiffré) est appelé **cryptogramme** (en anglais *ciphertext*) par opposition au message initial, appelé **message en clair** (en anglais *plaintext*) ;
- faire en sorte que le destinataire saura les déchiffrer.

Le fait de coder un message de telle façon à le rendre secret s'appelle **chiffrement**. La méthode inverse, consistant à retrouver le message original, est appelée **déchiffrement**.



Le chiffrement se fait généralement à l'aide d'une **clef de chiffrement**, le déchiffrement nécessite quant à lui une **clef de déchiffrement**. On distingue généralement deux types de clefs :

- *Les clés symétriques*: il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.
- *Les clés asymétriques*: il s'agit de clés utilisées dans le cas du chiffrement asymétrique (aussi appelé *chiffrement à clé publique*). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

On appelle *décryptement* (le terme de *décryptage* peut éventuellement être utilisé également) le fait d'essayer de déchiffrer *illégitimement* le message (que la clé de déchiffrement soit connue ou non de l'*attaquant*).

On appelle **cryptanalyse** la reconstruction d'un message chiffré en clair à l'aide de méthodes mathématiques. Ainsi, tout cryptosystème doit nécessairement être résistant aux méthodes de cryptanalyse. Lorsqu'une méthode de cryptanalyse permet de déchiffrer un message chiffré à l'aide d'un cryptosystème, on dit alors que l'algorithme de chiffrement a été « cassé ».

Les postulats de sécurité associés à la cryptographie sont :

- **L'intégrité des données** : Le contrôle d'intégrité d'une donnée consiste à vérifier que cette donnée n'a pas été altérée, frauduleusement ou accidentellement. Garantir que les données sont toujours tels qu'elles ont été créées, sans altération d'aucune sorte.
- **Le contrôle d'accès** : Il s'agit d'authentifier les utilisateurs de façon à limiter l'accès aux données, serveurs et ressources aux seules personnes autorisées (un mot de passe pour un disque dur, par exemple). Protéger l'accès à une ressource via divers moyens, souvent basé sur un secret possédé par les seuls personnes autorisées.
- **La confidentialité** : Il s'agit de rendre l'information inintelligible à tous les opposants tant lors de sa conservation qu'au cours de son transfert par un canal de communication. Protection d'une donnée contre son écoute par des individus non autorisés.
- **L'identification**: Le contrôle d'identification consiste à s'assurer que le destinataire est bien celui qui prétant être (authentification des partenaires) et d'obtenir une garantie que l'expéditeur a bien signé l'acte (authentification de l'origine des informations). Pouvoir vérifier que la donnée provient bien de la bonne source
- **La non répudiation**: Il s'agit de garantir l'authenticité de l'acte. L'expéditeur ne peut nier le dépôt d'information, le réceptionneur ne peut nier la remise d'information, ni l'un ni l'autre ne peut nier le contenu de cette information. Ne pas pouvoir nier être la source d'une donnée

## CAIN (Confidentialité - Authentification - Intégrité - Non-répudiation)

- **Confidentialité** des informations stockées/manipulées
  - utilisation d'un algorithme de chiffrement.
  - empêcher l'accès aux infos pour ceux qui ne sont pas autorisés.
- **Authentification** d'utilisateurs/de ressources
  - utilisation d'algorithmes d'authentification.
  - Alice s'identifie à Bob en prouvant qu'elle connaît un secret  $S$ , (ex : un mot de passe).
- **Intégrité** des informations stockées/manipulées
  - vérifier que les infos transmises n'ont pas subi d'*altérations*
- **Non-répudiation** des informations
  - utilisation d'algorithmes de signatures
  - empêcher un utilisateur de se dédire

- **Le chiffrement** est le procédé grâce auquel on peut rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (dé)chiffrement.
- Une **clé** est un paramètre utilisé en entrée d'une opération cryptographique.
- **Déchiffrer** consiste à retrouver le texte original (aussi appelé clair) d'un message chiffré dont on possède la clé de (dé)chiffrement.
- **Décrypter** consiste à retrouver le texte original à partir d'un message chiffré sans posséder la clé de (dé)chiffrement.
- **Texte clair** [Clear text ou Plain text] : Caractères ou bits sous une forme lisible par un humain ou une machine.
- **Texte chiffré** [Cipher text] : Résultat de la manipulation de caractères ou de bits via des substitutions, transpositions, ou les deux.
- La différence entre un **algorithme** et un **protocole** est une question d'interactivité : pour un algorithme, une seule personne est impliquée, celle qui fait les calculs ; pour un protocole, plusieurs entités interviennent, il y a échange d'informations.

**Cryptage** : transformation d'un message clair en un message crypté.

**Décryptage** : transformation inverse du cryptage qui permet de retrouver à partir d'un message crypté, le message clair correspondant.

<b>Le chiffrement</b>	<b>Le déchiffrement</b>
-----------------------	-------------------------

Transformation d'un message pour en cacher le sens

- L'émetteur doit transmettre un message  $M$  (en clair)  $M \in \text{Messages\_à\_envoyer}$

- Il construit un texte chiffré  $C$  au moyen d'une fonction  $C = E_k(M)$  qui dépend clé  $k$   
 $C \in \text{Messages\_chiffrés}$

Opération inverse du chiffrement  
Récupération d'un message en clair

- La fonction de déchiffrement  $D_{k'}(C)$   
 $\text{Messages\_chiffrés} \rightarrow \text{Messages\_à\_envoyer}$

- Possède la propriété d'être l'inverse à gauche de  $E_k$  :  
 $D_{k'}(C) = D_{k'}(E_k(M)) = M$

L'analyse cryptographique ou Cryptanalyse a pour objet de percer l'écran logique derrière lequel sont cachés les informations chiffrées

- Le chiffrement place l'information dans un coffre fort virtuel dont les personnes non autorisées ignorent la combinaison
- Ceux qui souhaitent accéder aux informations tentent de forcer le coffre soit en recherchant la combinaison, soit en essayant de découvrir une faiblesse insoupçonnée du coffre ou de la serrure Et ce idéalement sans laisser de trace
- ☒ La cryptologie est donc un jeu à deux joueurs
- Le cryptologue-concepteur conçoit des moyens de chiffrement offrant la meilleure protection possible
- Le cryptologue-décodeur utilise tous les moyens imaginables pour percer à jour les messages chiffrés interceptés.

**Message clair**  $m$  : chaîne de caractères composée de lettres de l'alphabet  $\mathcal{A}$  et dont on veut en général conserver la confidentialité. On note  $\mathcal{M}$  l'ensemble de tous les messages clairs possibles.

**Message crypté**  $c$  : chaîne de caractères composée de lettres de l'alphabet  $\mathcal{A}$ , correspondant à un message clair, et dont la diffusion à des entités non autorisées ne doit pas dévoiler pas d'information sur ce message clair. On note  $\mathcal{C}$  l'ensemble de tous les messages cryptés.

**Coder** : Transformer un texte, une information en remplaçant les **mots** dans une écriture faite de signes prédefinis.

**Chiffrer** : Transformer un texte, une information en remplaçant les **lettres** dans une écriture faite de signes prédefinis.<sup>1</sup>

**Chiffrement** : Transformation à l'aide d'une clé de chiffrement d'un message en clair en un message incompréhensible si on ne dispose pas d'une clé de déchiffrement (en anglais encryption)

**Cryptogramme** : Message chiffré

**Clé** : Une clé est un paramètre utilisé en entrée d'une opération cryptographique (chiffrement, déchiffrement, ...)

**Déchiffrer** : Transformation du texte chiffré en texte clair à l'aide de la clé de déchiffrement

**Décrypter** : Retrouver le message clair correspondant à un message chiffré sans posséder la clé de

dechiffrement (terme que ne possèdent pas les anglophones, qui eux « cassent » des codes secrets) Vu le sens des mots chiffrer; dechiffrer et decrypter, le terme « crypter » n'a pas de raison d'être (l'Academie française précise que le mot est à bannir et celui-ci ne figure pas dans son dictionnaire), en tout cas pas dans le sens où on le trouve en général utilisé. Pas plus que le terme « decoder »<sup>2</sup>.

1 Toute la différence entre coder et chiffrer : Le nombre de combinaisons possibles.

## Un système cryptographique ou procédé (ou algorithme) de chiffrement est la donnée

- 1 d'un espace des clefs  $\mathcal{K}$  ;
- 2 de fonctions de chiffrement et de déchiffrement paramétrées par des éléments de  $\mathcal{K}$  et qui vérifient la propriété de déchiffrement.

- Intuitivement, un **code** est une méthode de transformation qui convertit la représentation d'une information en une autre (cette définition nécessitera d'être précisée selon le contexte)
- Selon le but recherché, les codes ont différents usages :
  - **efficacité** de la transmission : **compression des données**
  - **sécurité** de l'information : **cryptage, authentification**
  - **intégrité** du message : **détection, correction des erreurs**
- A l'action de coder (le **codage**), on couple le **décodage** qui a pour but de restituer tout ou partie des messages émis par la source.
- On distingue le **codage avec perte** d'information du **codage sans perte**
- On dissocie le **codage de source** du **codage de canal** : le premier est une représentation (binnaire) concise d'un message, le second vise à adapter la sortie du premier au canal où elle va transiter.
  - la communication fait référence à la circulation d'**information** entre une **source** et un **récepteur**
  - la source émet un **message** qui, avant d'être lu par le récepteur, va transiter dans un **canal**
  - toute perturbation du message due au canal s'appelle le **bruit**
  - la nature de la source peut être variée : on ne considère ici que des **sources discrètes** ( $\neq$  continues) et **sans mémoire** (non-markoviennes)
  - on peut ajouter de la **redondance** au message en cas de bruit
  - on peut aussi vouloir garder à tout prix le message **secret**



Les opérations de chiffrement et de **codage** font partie de la **théorie de l'information** et de la **théorie des codes**. La différence essentielle réside dans la volonté de protéger les informations et d'empêcher des tierces personnes d'accéder aux données dans le cas du chiffrement. Le codage consiste à transformer de l'information (des données) vers un ensemble de mots. Chacun de ces mots est constitué de symboles. La **compression** est un codage : on transforme les données vers un ensemble de mots adéquats destinés à réduire la taille mais il n'y a pas de volonté de dissimuler (bien que cela se fasse implicitement en rendant plus difficile d'accès le contenu).

Le "code" dans le sens cryptographique du terme travaille au niveau de la sémantique (les mots ou les phrases). Par exemple, un code pourra remplacer le mot "avion" par un numéro. Le chiffrement travaille sur des composantes plus élémentaires du message, les lettres ou les bits, sans s'intéresser à la signification du contenu. Un code nécessite une table de conversion, aussi appelée "dictionnaire" (*codebook* en anglais). Ceci étant, "code" et "chiffrement" sont souvent employés de manière synonyme malgré cette différence.

L'exemple de codage le plus connu est le [Code Morse](#) International. Le Morse a été mis au point afin de permettre la transmission d'un texte en impulsions de type tout-ou-rien sans aucun équipement spécialisé, mis à part une clef de Morse et en étant facilement décodable à l'oreille humaine par un télégraphiste spécialement formé à cet effet. La transmission en tout-ou-rien simplifie grandement les équipements utilisés pour la transmission (en filaire, signaux lumineux ou hertziens) tout en augmentant significativement la facilité de décodage dans des conditions de signal faible (rapport signal/bruit faible). L'alphabet Morse étant du domaine public et encore enseigné aujourd'hui, particulièrement dans la communauté [radioamateur](#), il ne peut donc pas servir à cacher un message d'un adversaire. Son usage est lié principalement à ses qualités propres en transmission. On parle donc ici d'un codage dans le sens le plus pur du terme.

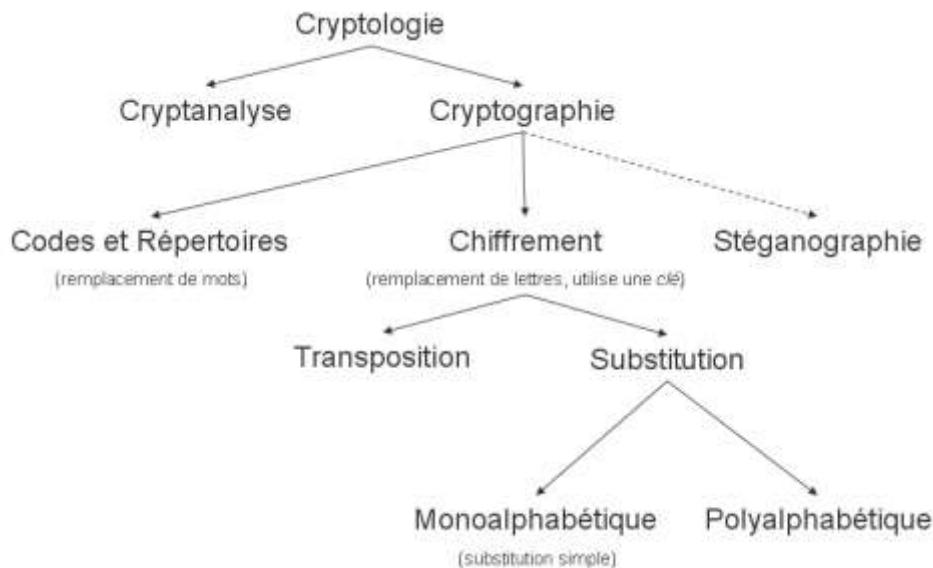
On peut aussi considérer que le chiffrement doit résister à un [adversaire](#) « intelligent » qui peut attaquer de plusieurs manières alors que le codage est destiné à une transmission sur un [canal](#) qui peut être potentiellement bruité. Ce [bruit](#) est un phénomène [aléatoire](#) qui n'a pas d'« intelligence » intrinsèque mais peut toutefois être décrit [mathématiquement](#).

cryptologie = cryptographie + cryptanalyse

À côté de la fonction de chiffrement, qui permet de préserver le secret des données lors d'une transmission, et qui a été utilisée depuis très longtemps, la cryptologie moderne a développé de nouveaux buts à atteindre et qu'on peut énumérer de manière non exhaustive

- confidentialité
- intégrité des données
- authentification des divers acteurs
- non-répudiation d'un contrat numérique
- signature numérique
- certification
- contrôle d'accès
- gestion des clés
- preuve de connaissance

Dans le schéma ci-dessous figurent les différentes branches de la cryptographie classique.



Le chiffrement par substitution consiste à remplacer dans un message une ou plusieurs entités (généralement des lettres) par une ou plusieurs autres entités.

On distingue généralement plusieurs types de cryptosystèmes par substitution :

- La **substitution monoalphabétique** consiste à remplacer chaque lettre du message par une autre lettre de l'alphabet

Chaque lettre est remplacée par une autre lettre ou symbole. Parmi les plus connus, on citera le chiffre de César, le chiffre affine, ou encore les chiffres désordonnés. Tous ces chiffres sont sensibles à l'analyse de fréquence d'apparition des lettres (nombre de fois qu'apparaît une même lettre dans un texte). De nos jours, ces chiffres sont utilisés pour le grand public, pour les énigmes de revues ou de journaux.

Chiffrement de César :

Il s'agit d'un des plus simples et des chiffres classiques les plus populaires. Son principe est un décalage des lettres de l'alphabet. Dans les formules ci-dessous, p est l'indice de la lettre de l'alphabet, k est le décalage.

Pour le chiffrement, on aura la formule  $C = E(p) = (p + k) \text{ mod } 26$

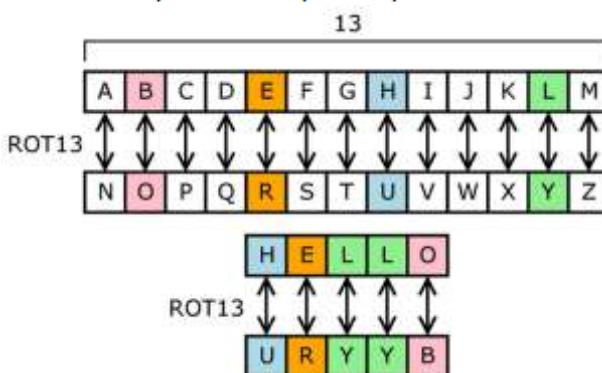
Pour le déchiffrement, il viendra  $p = D(C) = (C - k) \text{ mod } 26$

Si on connaît l'algorithme utilisé (ici César), la cryptanalyse par force brute est très facile. En effet, dans le cas du chiffre de César, seules 25 (!) clés sont possibles.

Le **ROT13** ("rotate by 13 places") (une variante du chiffrier de César) est un algorithme très simple de chiffrement de texte. Comme son nom l'indique, il s'agit d'un décalage de 13 caractères de chaque lettre du texte à chiffrer.

L'avantage de ROT13, c'est le fait que le décalage soit de 13 : Comme l'alphabet comporte 26 lettres le même « algorithme » sert à la fois pour chiffrer et pour déchiffrer.

Il est encore utilisé dans les logiciels de messagerie comme Outlook express ou, dans un forum, on souhaite cacher une réponse à une question posé.



Le **ROT13** (une variante de la **méthode César**) est un algorithme très simple de chiffrement de texte.

Comme son nom l'indique, il s'agit d'un décalage de 13 caractères de chaque lettre du texte à chiffrer. Le défaut de ce chiffrement est que s'il s'occupe des lettres, il ne s'occupe pas des chiffres, des symboles et de la ponctuation.

C'est pourquoi on supprime du texte à chiffrer toute accentuation, et si on veut conserver un texte correctement chiffré, il est nécessaire d'écrire les nombres en toutes lettres.

L'avantage de ROT13, c'est le fait que le décalage soit de 13!

L'alphabet comporte 26 lettres, et si on applique deux fois de suite le chiffrement, on obtient comme résultat le texte en clair. Pour cela on doit considérer l'alphabet comme circulaire, c'est-à-dire qu'après la lettre Z on a la lettre A, ce qui permet de grandement simplifier son usage et sa programmation puisque c'est la même procédure qui est utilisée pour le chiffrement et le déchiffrement.

- La **substitution polyalphabétique** consiste à utiliser une suite de chiffres monoalphabétiques réutilisée périodiquement

C'est une amélioration décisive du chiffre de César. Sa force réside dans l'utilisation non pas d'un, mais de 26 alphabets décalés pour chiffrer un message. On parle du *carré de Vigenère*. Ce chiffre utilise une clef qui définit le décalage pour chaque lettre du message (A : décalage de 0 cran, B : 1 cran, C : 2 crans, ..., Z : 25 crans).

## Chiffre de Vigenere :

C'est une amélioration décisive du chiffre de César. Sa force réside dans l'utilisation non pas d'un, mais de 26 alphabets décalés pour chiffrer un message. On parle du *carré de Vigenère*. Ce chiffre utilise une clef qui définit le décalage pour chaque lettre du message (A : décalage de 0 cran, B : 1 cran, C : 2 crans, ..., Z : 25 crans).

**Exemple :** chiffrer le texte "**CHIFFRE DE VIGENERE**" avec la clef "**BACHELIER**" (cette clef est éventuellement répétée plusieurs fois pour être aussi longue que le texte clair)

Clair	C	H	I	F	R	E	D	E	V	I	G	E	N	E	R	E
Clef	B	A	C	H	E	L	I	E	R	B	A	C	H	E	L	I
Décalage	1	0	2	7	4	11	8	4	17	1	0	2	7	4	11	8
Chiffré	D	H	K	M	J	C	M	H	V	W	T	I	L	R	P	Z

FIG. 3.8 – Application du carré de Vigenère

La lettre de la clef est dans la colonne la plus à gauche, la lettre du message clair est dans la ligne tout en haut. La lettre chiffrée est à l'intersection des deux.

L'emploi du carré de Vigenère est souvent sujet à erreurs : la lecture en est pénible et, à la longue fatigante. Beaucoup de cryptologues préfèrent se servir d'une "régllette", facile à construire, et d'un maniement plus rapide.

- **La substitution homophonique** permet de faire correspondre à chaque lettre du message en clair un ensemble possible d'autres caractères

Il s'agit de remplacer une lettre non pas par un symbole unique, mais par un symbole choisi au hasard parmi plusieurs. Dans sa version la plus sophistiquée, on choisira un nombre des symboles proportionnel à la fréquence d'apparition de la lettre. Ainsi, on obtient un renversement des fréquences ce qui permet de faire disparaître complètement les indications fournies par la fréquence. On parle de Substitution Homophonique ou de Substitution à représentation multiple.

- **La substitution de polygrammes** consiste à substituer un groupe de caractères (polygramme) dans le message par un autre groupe de caractères

Il s'agit ici de chiffrer un groupe de n lettres par un autre groupe de n symboles. On citera notamment le chiffre de Playfair et le chiffre de Hill. Ce type de chiffrement porte également le nom de *substitutions polygrammiques*.

## Chiffrement de HILL :

### 3.2.2 Chiffre de Hill (1929)

Les lettres sont d'abord remplacées par leur rang dans l'alphabet. Les lettres  $P_k$  et  $P_{k+1}$  deviennent  $C_k$  et  $C_{k+1}$

$$\begin{pmatrix} C_k \\ C_{k+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} P_k \\ P_{k+1} \end{pmatrix} \pmod{26}$$

Les composantes de cette matrice doivent être des entiers positifs. De plus la matrice doit être inversible dans  $Z_{26}$ . Cependant, sa taille n'est pas fixée à 2. Elle grandira selon le nombre de lettres à chiffrer simultanément.

Chaque digramme clair ( $P_1$  et  $P_2$ ) sera chiffré ( $C_1$  et  $C_2$ ) selon :

$$C_1 \equiv aP_1 + bP_2 \pmod{26}$$

$$C_2 \equiv cP_1 + dP_2 \pmod{26}$$

**Exemple de chiffrement :** Alice prend comme clef de cryptage la matrice

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$$

pour chiffrer le message "je vous aime" qu'elle enverra à Bob. Après avoir remplacé les lettres par leur rang dans l'alphabet (a=1, b=2, etc.), elle obtiendra

$$C_1 \equiv 9 * 10 + 4 * 5 \pmod{26} = 110 \pmod{26} = 6$$

$$C_2 \equiv 5 * 10 + 7 * 5 \pmod{26} = 85 \pmod{26} = 7$$

Elle fera de même avec les 3e et 4e lettres, 5e et 6e, etc. Elle obtiendra finalement le résultat de la figure 3.6.

Lettres	j	e	v	o	u	s	a	i	m	e
Rangs ( $P_k$ )	10	5	22	15	21	19	1	9	13	5
Rangs chiffrés ( $C_k$ )	6	7	24	7	5	4	19	16	7	22
Lettres chiffrées	F	G	X	G	E	D	S	P	G	V

FIG. 3.6 – Exemple de chiffrement de Hill

Pour déchiffrer, le principe est le même que pour le chiffrement : on prend les lettres deux par deux, puis on les multiplie par une matrice

$$\begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \pmod{26}$$

Cette matrice doit être l'inverse de matrice de chiffrement (modulo 26). Ordinairement, cet inverse est

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

**Exemple de déchiffrement :** Pour déchiffrer le message d'Alice, Bob doit calculer :

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}^{-1} = \frac{1}{43} \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} \pmod{26} = (43)^{-1} \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} \pmod{26}$$

Comme  $\gcd(43, 26) = 1$ ,  $(43)^{-1}$  existe dans  $Z_{26}$  et  $(43)^{-1} = 23$ . Bob a la matrice de déchiffrement :

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}^{-1} = 23 \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} \pmod{26} = \begin{pmatrix} 161 & -92 \\ -115 & 207 \end{pmatrix} \pmod{26} = \begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix} \pmod{26}$$

Bob prend donc cette matrice pour déchiffrer le message "FGXGE DSPGV". Après avoir remplacé les lettres par leur rang dans l'alphabet (A=1, B=2, etc.), il obtiendra :

$$P_1 \equiv 5 * 6 + 12 * 7 \pmod{26} = 114 \pmod{26} = 10$$

$$P_2 \equiv 5 * 6 + 25 * 7 \pmod{26} = 265 \pmod{26} = 5$$

Il fera de même avec les 3e et 4e lettres, 5e et 6e, etc. Il obtiendra finalement le résultat de la figure 3.7

Lettres chiffrées	F	G	X	G	E	D	S	P	G	V
Rangs chiffrés ( $C_k$ )	6	7	24	7	5	4	19	16	7	22
Rangs ( $P_k$ )	10	5	22	15	21	19	1	9	13	5
Lettres	j	e	v	o	u	s	a	i	m	e

FIG. 3.7 – Exemple de déchiffrement de Hill

Un chiffrement par substitution remplace chaque lettres par une autre. Dans ce cas, on peut effectuer une analyse statistique du chiffré pour obtenir la "clé" de la substitution, en connaissant la distribution de chaque lettres. Par exemple, en français, les lettres les plus fréquentes sont:

- E (14,72%)
- S (7,95%)
- A (7,64%)
- I (7,53%)
- T (7,24%)

En faisant correspondre une table de ce genre avec un message chiffré, on peut rapidement le décoder complètement.

Une **substitution** est une bijection qui consiste à remplacer chaque lettre d'un message par une autre lettre. Attention : partout où une lettre donnée apparaît, elle est toujours remplacée par la **même** lettre. Ainsi, si dans le texte original, "a" devient "u", alors dès qu'on verra un "a", on le remplacera par un "u".

Par exemple,

$$M = m \ e \ s \ s \ a \ g \ e \ s \ e \ c \ r \ e \ t \\ C = b \ x \ v \ v \ g \ o \ x \ v \ x \ u \ / \ x \ e$$

Dans les substitutions on peut transformer des lettres en d'autres symboles et non nécessairement d'autres lettres du **même alphabet**.

Une substitution ne change pas l'**ordre** des lettres dans un message mais seulement les lettres elle-mêmes figurant dans le message. En d'autres termes, à partir d'un message, on remplace chaque lettre par un autre symbole mais en conservant l'ordre d'apparition des lettres. Par exemple, si une lettre "e" apparaît en positions 3, 8, 25 d'un message "...  $\underbrace{e}_{3}$  ...  $\underbrace{e}_{8}$  ...  $\underbrace{e}_{25}$  ...", alors on la remplace,

disons, par le symbole "@" exactement aux positions 3, 8 et 25 de telle sorte que l'on obtienne :

"...  $\underbrace{@}_{3}$  ...  $\underbrace{@}_{8}$  ...  $\underbrace{@}_{25}$  ...". Pour changer l'ordre, on utilise les **transpositions**.

Dans une transposition, on ne modifie pas l'écriture des lettres mais leur ordre dans un message. Par exemple,

$$M = m \ e \ s \ s \ a \ g \ e \ s \ e \ c \ r \ e \ t \\ C = e \ m \ e \ a \ s \ g \ s \ e \ c \ s \ r \ t \ e$$

Une transposition opère de façon identique quel que soit le message. Cela signifie que la transposition ne dépend que de l'ordre des lettres dans un message et non des lettres elles-mêmes.

Elles consistent, par définition, à changer l'ordre des lettres. C'est un système simple, mais peu sûr pour de très brefs messages car il y a peu de variantes. Ainsi, un mot de trois lettres ne pourra être transposé que dans  $6 (=3!)$  positions différentes. Par exemple, "col" ne peut se transformer qu'en "col", "clo", "ocl", "olc", "lco" et "loc".

Lorsque le nombre de lettres croît, il devient de plus en plus difficile de retrouver le texte original sans connaître le procédé de brouillage. Ainsi, une phrase de 35 lettres peut être disposée de  $35! = 10^{40}$  manières différentes. Ce chiffrement nécessite un procédé rigoureux convenant auparavant entre les parties.

Une transposition rectangulaire consiste à écrire le message dans une grille rectangulaire, puis à arranger les colonnes de cette grille selon un mot de passe donné (le rang des lettres dans l'alphabet donne l'agencement des colonnes).

**Exemple :**

A la figure 3.15, on a choisi comme clef GRAIN pour chiffrer le message SALUT LES PETITS POTS. En remplissant la grille, on constate qu'il reste deux cases vides, que l'on peut remplir avec des nulles (ou pas, selon les désirs des correspondants).

G	R	A	I	N	
2	5	1	3	4	
S	A	L	U	T	
L	E	S	P	E	
T	I	T	S	P	
O	T	S	(A)	(B)	

A	G	I	N	R
1	2	3	4	5
L	S	U	T	A
S	L	P	E	E
T	T	S	P	I
S	O	(A)	(B)	T

FIG. 3.15 – Application d'une transposition

Les techniques cryptographiques de base sont les techniques qui permettent de répondre aux fonctionnalités que nous avons décrites précédemment. On peut citer essentiellement :

- Les techniques de chiffrement
- Les techniques de signature
- Les techniques d'authentification

¶ Ces techniques font appels à des **primitives cryptographiques** qui elles mêmes sont basées sur des **objets et problèmes mathématiques**.

La cryptographie à clé secrète fait plutôt usage de :

- Fonctions booléennes
  - Générateurs de suites pseudo-aléatoires
  - Corps finis
  - Codes correcteurs d'erreurs
- ¶ La cryptographie à clé publique fait plutôt usage de :
- Problème de la factorisation des entiers
  - Problème du logarithme discret dans des groupes arithmétiques
  - Problèmes liés à la résiduosité quadratique
  - Fractions continues, réseaux arithmétiques
  - Codes correcteurs d'erreurs

Il existe deux grandes catégories de systèmes cryptographiques :

1 **systèmes à clef secrète** (ou **symétriques**) :

$K_e = K_d = K$  et la clef  $K$  est gardée secrète par Alice et Bob. On dit que  $K$  est la **clef secrète** ;

2 **systèmes à clef publique** (ou **asymétriques**) :  $K_e \neq K_d$ ,

$K_e$  est connue de tout le monde : c'est la **clef publique**,  
 $K_d$  n'est connue que du seul Bob : c'est la **clef privée**.

¶ Algorithmes de chiffrement faibles (facilement cassables)

– Les premiers algorithmes utilisés pour le chiffrement d'une information étaient assez rudimentaires dans leur ensemble. Ils consistaient notamment au remplacement de caractères par d'autres. La confidentialité de l'algorithme de chiffrement était donc la pierre angulaire de ce système pour éviter un déchiffrement rapide. Exemples d'algorithmes de chiffrement faibles :

- -ROT13 (rotation de 13 caractères, sans clé) ;
- -Chiffre de César (décalage de trois lettres dans l'alphabet).

¶ Algorithmes de cryptographie symétrique (à clé secrète)

– Les algorithmes de chiffrement symétrique se fondent sur une même clé pour chiffrer et déchiffrer un message. Le problème de cette technique est que la clé, qui doit rester totalement confidentielle, doit être transmise au correspondant de façon sûre. En outre lorsqu'un grand nombre de personnes désirent communiquer ensemble, le nombre de clés augmente de façon importante (une pour chaque couple de communicants). Ceci pose des problèmes de gestions des clés.

- ¶ Chiffre de Vernam
- ¶ DES / 3DES
- ¶ AES
- ¶ RC4 / RC5

## Algorithmes de cryptographie asymétrique (à clé publique et privée)

- Pour résoudre en partie le problème de la gestion des clés, la cryptographie asymétrique a été mise au point dans les années 1970. Elle se base sur le principe de deux clés :
  - une publique, permettant le chiffrement ;
  - une privée, permettant le déchiffrement.
- Comme son nom l'indique, la clé publique est mise à la disposition de quiconque désire chiffrer un message. Ce dernier ne pourra être déchiffré qu'avec la clé privée, qui doit être confidentielle.
- Quelques algorithmes de cryptographie asymétrique très utilisés :
  - RSA ;
  - DSA ;
  - Protocole d'échange de clés Diffie-Hellman ;

## Fonctions de hachage

- Une fonction de hachage est une fonction qui convertit un grand ensemble en un plus petit ensemble, l'empreinte. Il est impossible de la déchiffrer pour revenir à l'ensemble d'origine, ce n'est donc pas une technique de chiffrement.
- Quelques fonctions de hachage très utilisées :
  - MD5 ;
  - SHA-1 ;

La cryptographie symétrique, également dite à clé secrète est la plus ancienne forme de chiffrement

L'un des concepts fondamentaux de la cryptographie symétrique est la clé, qui est une information devant permettre de chiffrer et de déchiffrer un message et sur laquelle peut reposer toute la sécurité de la communication.

Jusqu'aux communications numériques, les systèmes utilisaient l'alphabet et combinaient les substitutions — les symboles sont changés mais restent à leur place — et les transpositions — les symboles ne sont pas modifiés mais changent de place. Lorsque la substitution appliquée dépend de la place de la lettre dans le texte, on parle de substitution polyalphabétique.

Depuis l'avènement du numérique, les paradigmes du chiffrement symétrique ont bien changé. D'une part, la discipline s'est formalisée, même si la conception de système de chiffrement garde inévitablement un aspect artisanal. En effet dans ce domaine, la seule chose que l'on sait prouver est la résistance face à des types d'attaques connues, pour les autres... D'autre part, la forme du texte chiffré ayant changé, les méthodes ont suivi. Les algorithmes modernes chiffrent des suites de bits

On distingue deux types d'algorithmes, les algorithmes en blocs, qui prennent  $n$  bits en entrée et en ressortent  $n$ , et les algorithmes à flots, qui chiffrent bit par bit sur le modèle du chiffre de Vernam

36

## Applications avec une cryptographie symétrique

### Chiffrement de données

La **cryptographie asymétrique**, ou *cryptographie à clé publique* est fondée sur l'existence de fonctions à sens unique, c'est-à-dire qu'il est simple d'appliquer cette fonction à un message, mais extrêmement difficile de retrouver ce message à partir du moment où on l'a transformé.

- $F$  est à sens unique  $\Leftrightarrow$  Quelque soit  $x, y = f(x)$  est calculable rapidement.
- $X = f^{-1}(y)$  se calcule en un temps très long

En réalité, on utilise en cryptographie asymétrique des fonctions à sens unique et à brèche secrète. Une telle fonction est difficile à inverser, à moins de posséder une information particulière, tenue secrète, nommée clé privée.

- Une fonction est dite trappe ou à brèche secrète si elle est à sens unique sauf pour toute personne connaissant un secret ou une brèche, permettent de calculer un algorithme d'inversion rapide.
- On appelle exponentiation modulaire de la variable  $a$  la fonction  $a^p \bmod n$  ( $p$  fixe). Si l'existence d'un algorithme permettant de calculer des racines  $p$ -èmes modulo  $n$  est démontré, on ne connaît néanmoins pas cet algorithme. Par contre, si on connaît la factorisation de  $n$  (la brèche), on peut très facilement inverser l'exponentiation modulaire.

Une clé est utilisée pour coder le message et une autre pour décoder le message crypté. Dans un système à clé publique, chaque personne dispose de deux clés: une publique et une privée. Les messages chiffrés avec l'une des clés peuvent seulement être déchiffrés par l'autre clé de la paire.

## Applications avec une cryptographie asymétrique

- ☒ Transmission sécurisée pour une cryptographie symétrique
- ☒ Mécanismes d'authentification
- ☒ Certificats numériques
- ☒ Infrastructure à clé publique (IGC)
- ☒ Signatures numériques

## Fonction de hachage

- Une fonction de hash (anglicisme) ou fonction de hachage est une fonction qui associe à un grand ensemble de données un ensemble beaucoup plus petit (de l'ordre de quelques centaines de bits) qui est caractéristique de l'ensemble de départ
- Le résultat de cette fonction est par ailleurs aussi appelé **somme de contrôle, empreinte, résumé de message, condensé** ou encore **empreinte cryptographique** lorsque l'on utilise une fonction de hachage *cryptographique*
- Une fonction de hachage cryptographique est utilisée entre autres pour la signature électronique, et rend également possibles des mécanismes d'authentification par mot de passe sans stockage de ce dernier. Elle doit être résistante aux collisions, c'est-à-dire que deux messages distincts doivent avoir très peu de chances de produire la même signature. De par sa nature, tout algorithme de hachage possède des collisions mais on considère le hachage comme cryptographique si les conditions suivantes sont remplies :
  - Il est très difficile de trouver le contenu du message à partir de la signature (attaque sur la première pré image)
  - à partir d'un message donné et de sa signature, il est très difficile de générer un autre message qui donne la même signature (attaque sur la seconde pré image)
  - Il est très difficile de trouver deux messages aléatoires qui donnent la même signature (résistance aux collisions)
- Liste non exhaustive :
  - SHA-1 / SHA-256 / SHA-512
  - MD5

— Une fonction de hachage est une fonction qui convertit un grand ensemble en un plus petit ensemble, l'empreinte. Il est impossible de la déchiffrer pour revenir à l'ensemble d'origine, ce n'est donc pas une technique de chiffrement.

— Quelques fonctions de hachage très utilisées :

- ☒ MD5 ;
- ☒ SHA-1 ;

Cette propriété fait qu'elles sont très utilisées en informatique, en particulier pour accéder rapidement à des données grâce à des "**Tables de hachage**". En effet, une fonction de hachage permet d'associer à une chaîne de caractères un entier particulier. Ainsi, si nous connaissons l'empreinte des chaînes de caractères stockées, nous pouvons rapidement vérifier si une chaîne se trouve ou non dans cette table (en  $O(1)$  si la fonction de hachage est suffisamment bonne). Les fonctions de hachage sont aussi extrêmement utiles en cryptographie pour accélérer le cryptage. Les 2 algorithmes de condensation les plus utilisés sont le "**SHA**" (Secure Hash Algorithm) qui calcule un résumé de 160 bits, et le MD5 (Message Digest 5 – Run Rivest 1992), qui calcule un résumé de 128 bits nommé "**Message Digest**".

Une fonction de hachage est une fonction qui fait subir une succession de traitements à une donnée quelconque fournie en entrée pour en produire une « empreinte » servant à identifier la donnée initiale sans que l'opération inverse de décryptage soit possible.

Le terme hachage évite l'emploi de l'anglicisme hash. Le résultat de cette fonction est par ailleurs aussi appellé somme de contrôle, empreinte, résumé de message, condensé, condensat ou encore empreinte cryptographique.

**Les fonctions de hachage sont conçues pour effectuer un traitement de données rapide : calculer l'empreinte d'une donnée ne doit coûter qu'un temps négligeable. Une fonction de hachage doit aussi éviter le plus possible les collisions (deux empreintes identiques alors que les données diffèrent)**

Selon l'emploi de la fonction de hachage, il peut être souhaitable qu'un infime changement de la donnée en entrée (un seul bit, par exemple) entraîne une perturbation conséquente de l'empreinte correspondante, rendant une recherche inverse par approximations successives impossible : on parlera d'*effet avalanche*.

## Principe d'une fonction de hachage

Calcul d'empreinte

Message de taille quelconque, empreinte de taille fixe

Identification (pseudo-)unique d'un message

## Applications

Contrôle d'intégrité

Signature cryptographique

Génération de nombres aléatoires

Stockage de hachés de mots de passe

## Résistance à l'attaque sur la première pré-image

Il est très difficile de trouver le contenu du message à partir du haché

## Résistance à l'attaque sur la seconde pré-image

À partir d'un message donné, de son haché et du code source de la fonction de hachage, il est très difficile de générer un autre message qui donne le même haché

## Résistance aux collisions

Il est très difficile de trouver deux messages aléatoires qui donnent le même haché

## Fonctions utilisées

MD5 : ne plus utiliser !

SHA-1 : non recommandé

SHA-2 (SHA-256 à SHA-512) : sûr, mais faiblesses proches

SHA-3 : tout juste standardisée, vue comme une alternative

## Situation actuelle

Les fonctions de hachage sont critiques en crypto

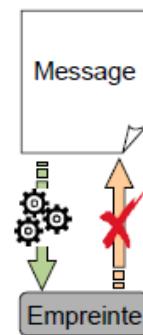
SHA-2 reste sûre mais proche de SHA-1 qui a des faiblesses connues

SHA-3 considérée "aussi sûre" mais totalement différente (fonction éponge)

) Diversification entre SHA-2 et SHA-3

**Mécanismes fournissant les services d'intégrité, d'authentification de l'origine des données et de non-répudiation de la source**

- Fonction de hachage à sens unique
  - ◆ Convertit une chaîne de longueur quelconque en une chaîne de taille inférieure et généralement fixe = empreinte ou condensé
  - ◆ A sens unique :
    - Facile à calculer mais difficile à inverser
    - Il est difficile de trouver deux messages ayant la même empreinte
- MD5 (*Message Digest 5*)
  - ◆ Empreinte de 128 bits
- SHA (*Secure Hash Algorithm*)
  - ◆ Norme NIST
  - ◆ Empreinte de 160 bits
  - ◆ SHA-1 révision publiée en 1994 (corrige une faiblesse non publique)
  - ◆ SHA-1 considéré comme plus sûr que MD5
  - ◆ SHA-2 (octobre 2000) agrandit la taille de l'empreinte



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

Une **fonction à sens unique** est une fonction facile à calculer mais difficile à inverser. La cryptographie à clef publique repose sur l'utilisation de fonctions à sens unique à brèche secrète : pour qui connaît le secret (i.e. la clef privée), la fonction devient facile à inverser.

Une **fonction de hachage à sens unique** est une fonction de hachage qui est en plus une fonction à sens unique : il est aisément de calculer l'empreinte d'une chaîne donnée, mais il est difficile d'engendrer des chaînes qui ont une empreinte donnée, et donc de déduire la chaîne initiale à partir de l'empreinte. On demande généralement en plus à une telle fonction d'être **sans collision**, c'est-à-dire qu'il soit impossible de trouver deux messages ayant la même empreinte. On utilise souvent le terme fonction de hachage pour désigner, en fait, une fonction de hachage à sens unique sans collision.

La plupart des fonctions de hachage à sens unique sans collision sont construites par itération d'une fonction de compression : le message  $M$  est décomposé en  $n$  blocs  $m_1, \dots, m_n$ , puis une fonction de compression  $f$  est appliquée à chaque bloc et au résultat de la compression du bloc précédent ; l'empreinte  $h(M)$  est le résultat de la dernière compression.

- ▶ en plus du **secret**, de nos jours, la cryptographie doit assurer l'**authentification** des messages
- ▶ elle doit également en assurer l'**intégrité** ou même la **non-répudiation**
- ▶ les **fonctions de hachage** (*hash-function* en anglais) remplissent bien ces rôles en produisant des **empreintes** de messages
- ▶ le calcul d'une empreinte est finalement le moyen le plus simple de détecter la présence d'**erreurs de transmission** ou de **corruption**
- ▶ les fonctions de hachage permettent aussi de **diminuer la quantité d'information** à chiffrer
- ▶ elles sont utilisées dans les protocoles de **signatures électroniques** quand les données à signer sont trop volumineuses pour l'être par d'autres procédés
- ▶ certains **mots de passe jetables** sont une application directe des fonctions de hachage
- ▶ finalement, les fonctions de hachage sont un **mécanisme de base** de nombreux protocoles cryptographiques

- ▶ l'empreinte calculée par une fonction de hachage doit être évoluée, dans le sens où elle est censée **identifier** le message
- ▶ on fait l'analogie avec les **empreintes digitales**, qui à défaut de servir à reconstituer les caractéristiques de quelqu'un, permettent de l'identifier
- ▶ d'un point de vue technique, les fonctions de hachage, en tant que codes non injectifs, sont des codes ambigus, compresseurs avec perte (cf. Cours 2)
- ▶ **fonction de hachage** :

$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

$$x \mapsto y = H(x)$$

- ▶ une fonction de hachage est donc une fonction qui transforme une chaîne de taille quelconque en une chaîne de taille fixe : son **empreinte**
- ▶  $H(x)$  doit être **très rapide à calculer** à partir de  $x$

# UTILISATION DES FONCTIONS DE HACHAGE

## Intégrité des données

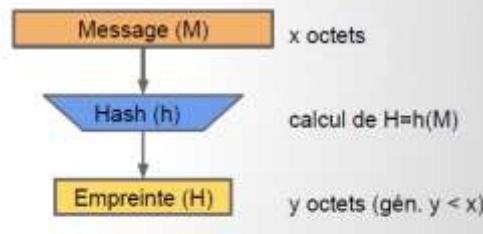
- ▶ on calcule l'empreinte d'une donnée
- ▶ on transmet la donnée et son empreinte
- ▶ à réception, on recalcule l'empreinte de la donnée que l'on compare à l'empreinte reçue
- ▶ cette technique est utilisée contre les virus informatiques ou dans la distribution de logiciels
- ▶ on nomme cette utilisation **MDC** (*Modification Detection Code* ou *Manipulation Detection Code*)

## Authentification des données

- ▶ si Alice et Bob partagent une **clef secrète** et une **fonction de hachage  $H$** , ils peuvent se servir de leur clef comme **valeur initiale** de  $H$
- ▶ si Alice envoie à Bob un message  $m$  et son empreinte  $H(m)$ , à réception, Bob peut vérifier que le message provient bien d'Alice
- ▶ les **MAC** (*Message Authentication Code*) gèrent ainsi l'intégrité des données et l'authentification de leur source

# Principe général

- $M$  peut avoir "n'importe quelle taille":  $x$  (ex. SHA256: ~2M To)
- $H$  est d'une taille "fixée" à l'avance:  $y$
- Il n'y a pas de lien entre  $x$  et  $y$ : la taille de l'empreinte est indépendante de celle du message
- Le calcul de  $H=h(M)$  doit être "simple", l'inverse doit être complexe ("impossible")



Une fonction de hachage est une fonction:

- **Unidirectionnelle**

On peut facilement calculer une empreinte, mais l'inverse doit être difficile

- **Déterministe**

Une entrée produit toujours la même sortie

- **Uniforme**

Les sorties possibles sont distribuées de façon uniforme pour chaque entrée

On peut aussi parler de la propriété suivante (en dehors d'un contexte cryptographique):

- **Continuité**

Deux messages proches doivent avoir une empreinte proche

## Applications avec une cryptographie de hachage

- ❑ Une fonction de hachage cryptographique est utilisée entre autres pour la signature électronique.
- ❑ Une fonction de hachage rend également possibles des mécanismes d'authentification par mot de passe sans stockage de ce dernier.
- ❑ Intégrité des données

- MD5 (Message Digest 5) est une fonction de hachage cryptographique qui permet d'obtenir pour chaque message une empreinte numérique (en l'occurrence une séquence de 128 bits ou 32 caractères en notation hexadécimale) avec une probabilité très forte que, pour deux messages différents, leurs empreintes soient différentes.
- MD5 travaille avec un message de taille variable et produit une empreinte de 128 bits. Le message est divisé en blocs de 512 bits, on applique un remplissage de manière à avoir un message dont la longueur est un multiple de 512. Le remplissage se présente comme suit :
  - on ajoute un '1' à la fin du message
  - on ajoute une séquence de '0' (le nombre de zéros dépend de la longueur du remplissage nécessaire)
  - on écrit la taille du message, un entier codé sur 64 bits
  - Ce remplissage est toujours appliqué, même si la longueur du message peut être divisée par 512. Cette méthode de padding est semblable à celle utilisée dans la plupart des algorithmes de Message Digest des familles MD

## FONCTION DE CONDENSATION MESSAGE DIGEST MD5

Cet algorithme est (était) surtout utilisé pour les signatures numériques (notion utilisée, lors de la validation de certificats d'authenticité comme nous le verrons plus loin).

Voici les différentes étapes de son fonctionnement:

### Etape 1 : Complétion

Le message est constitué de  $b$  bits. On complète le message par un 1, et suffisamment de 0 pour que le message étendu ait une longueur multiple de 512 bits. Après ce traitement initial, on manipule le texte d'entrée par blocs de 512 bits divisés en 16 sous-blocs  $M[i]$  de 32 bits.

### Etape 2 : Initialisation

On définit les variables de chaînage de 32 bits A, B, C et D initialisées ainsi (les chiffres sont hexadécimaux):

$$A=01234567, B=89ABCDEF, C=FEDCBA98, D=76543210$$

On définit aussi 4 fonctions non linéaires F, G, H et I, qui prennent des arguments codés sur 32 bits, et renvoie une valeur sur 32 bits, les opérations se faisant bit à bit.

$$F(X,Y,Z) = (X \text{ AND } Y) \text{ OR } (\text{NOT}(X) \text{ AND } Z)$$

$$G(X,Y,Z) = (X \text{ AND } Z) \text{ OR } (Y \text{ AND } \text{NOT}(Z))$$

$$H(X,Y,Z) = X \text{ XOR } Y \text{ XOR } Z$$

$$I(X,Y,Z) = Y \text{ XOR } (X \text{ OR } \text{NOT}(Z))$$

Ce qu'il y a d'important avec ces 4 fonctions est que si les bits de leurs arguments X, Y et Z sont indépendants, les bits du résultat le sont aussi.

### Etape 3 : Calcul itératif

La boucle principale a 4 rondes qui utilise chaque fois une fonction non linéaire différente (d'où le fait qu'il y en ait 4...). Chaque ronde consiste donc en 16 exécutions d'une opération.

Chaque opération calcule une fonction non linéaire de trois des variables A, B, C et D, y ajoute un sous bloc  $M[i]$  du texte à chiffrer, une constante  $s$  prédéfinie (codée sur 32 bits) et effectue un décalage circulaire vers la gauche, d'un nombre variable  $n$  de bits. Voici l'exemple pour A:

$$- A = B + A + F(B,C,D) + M[i] + s \mid \text{décalé circulairement de } n \text{ vers la gauche}$$

$$- A = B + A + G(B,C,D) + M[i] + s \mid \text{décalé circulairement de } n \text{ vers la gauche}$$

$$- A = B + A + H(B,C,D) + M[i] + s \mid \text{décalé circulairement de } n \text{ vers la gauche}$$

-  $A = B + A + I(B,C,D) + M[i] + s$  | décalé circulairement de  $n$  vers la gauche

Cette nouvelle valeur de  $A$  est ensuite sommée avec l'ancienne.

#### Etape 4 : Ecriture du résumé

Le résumé sur 128 bits est obtenu en mettant bout à bout les 4 variables de chaînage  $A$ ,  $B$ ,  $C$ ,  $D$  de 32 bits obtenues à la fin de l'itération.

Le MD5 n'est pas sûre et pas unique (deux entrées différentes peuvent donner la même signature: nous parlons alors de collision). Cependant, la fonction MD5 reste encore largement utilisée comme outil de vérification lors des téléchargements et l'utilisateur peut valider l'intégrité de la version téléchargée grâce à l'empreinte. Ceci peut se faire avec un programme comme *md5sum* pour MD5 et *sha1sum* pour SHA-1. (cf. chapitre de Codes Correcteurs).

Voici l'empreinte (appelée abusivement "signature") obtenue sur une phrase :

MD5("Wikipedia, l'encyclopédie libre et gratuite") = d6aa97d33d459ea3670056e737c99a3d

En modifiant un caractère, cette empreinte change radicalement :

MD5("Wikipedia, l'encyclopédie libre et gratuitE") = 5da8aa7126701c9840f99f8e9fa54976

Très concrètement, la vérification de l'empreinte ou somme de contrôle MD5 peut-être réalisée de la façon suivante: lors du téléchargement d'un programme, nous notons la série de caractères indiquée sur la page de téléchargement. Quand ce téléchargement est terminé, nous lançons un des utilitaires susmentionné.

Une fonction de hachage calcule l'empreinte  $y$  (ou digest) d'un message  $x$ . Cette fonction  $F$  doit être une fonction à sens unique c'est-à-dire qu'il doit être facile de trouver  $y$  à partir de  $x$ , mais très difficile de trouver  $x$  à partir de  $y$ . Elle doit aussi être très sensible pour qu'une petite modification du message entraîne une grande modification de l'empreinte. En envoyant le message accompagné de son empreinte, le destinataire peut ainsi s'assurer de l'intégrité du message en recalculant le résumé à l'arrivée et en le comparant à celui reçu. Si les deux résumés sont différents, cela signifie que le fichier n'est plus le même que l'original : il a été altéré ou modifié par une tierce personne. Les fonctions de hachage les plus répandus sont MD5 et SHA-1 qui sont basés tous les deux sur MD4, MD5 générant des empreintes de 128 bits et SHA-1 de 160 bits(seul MD5 sera décrit, ces deux fonctions ayant un fonctionnement similaire). Nous verrons plus en avant qu'une fonction de hachage joue un rôle dans la signature électronique, méthode qui permet d'authentifier l'expéditeur.

#### Transposition :

Avec le principe de la transposition toutes les lettres du message sont présentes, mais dans un ordre différent. Il utilise le principe mathématique des **permutations**. Plusieurs types différents de transpositions existent :

- **Transposition simple par colonnes (par bloc)** : on écrit le message horizontalement dans une matrice prédéfinie, et on trouve le texte à chiffrer en lisant la grille verticalement (cf. la figure ci-dessous). Le destinataire légal pour décrypter le message réalise le procédé inverse.

L'algorithme allemand ADFGFX est fondé sur ce principe et fut utilisé pendant la première guerre mondiale. Il fut cassé par une jeune étudiante française.

*Exemple : texte à chiffrer = « I LOVE MY ENGLISH TEACHER »  
utilise une matrice [6;4].*

I	L	O	V
E	M	Y	E
N	G	L	I
S	H	T	E
A	C	H	E
R			

fig 1.2 TRANSPOSITION SIMPLE PAR COLONNES

texte chiffré = « IENSA RLMGH COYLT HVEIE E »

- **Transposition complexe par colonnes** : un mot clé secret (avec uniquement des caractères différents) est utilisé pour dériver une séquence de chiffres commençant à 1 et finissant au nombre de lettres composant le mot clé. Cette séquence est obtenue en numérotant les lettres du mot clé en partant de la gauche vers la droite et en donnant l'ordre d'apparition dans l'alphabet. Une fois que la séquence de transposition est obtenue, on chiffre en écrivant d'abord le message par lignes dans un rectangle (comme le dessin ci-dessous le montre), puis on lit le texte par colonnes en suivant l'ordre déterminé par la séquence.

*Exemple : texte en clair = « I LOVE MY ENGLISH TEACHER »  
utilise le mot clé SERGIO.*

Cle:    S E R G I O  
              6 1 5 2 3 4

I	L	O	V	E	M
Y	E	N	G	L	I
S	H	T	E	A	C
H	E	R			

fig 1.3 TRANSPOSITION COMPLEXE PAR COLONNES

texte chiffré = « LEHEV GEELA MICON TRIYS H »

- **Transposition par carré polybique (par clé)** : un mot clé secret est utilisé pour construire un alphabet dans un tableau. Les coordonnées des lignes et des colonnes correspondant au lettres du texte à chiffrer sont utilisés pour transcrire le message en chiffres. Avec ce procédé chaque lettre du texte en clair est représenté par deux chiffres écrit verticalement. Ces deux coordonnées sont ensuite transposées en les recombinant par deux sur la ligne ainsi obtenue.

*Exemple : texte en clair = « CRYPTOLOGY IS A PASSIONATE TOPIC »  
utilise le mot clé : SERGIO.*

Cle:    1 2 3 4 5 6

1	S	E	R	G	I	O
2	A	F	T	P	K	M
3	L	N	Z	Y	U	X
4	W	Q	B	V	C	H
5	J	D	&	'	#	}
6	%	S	E	*	p	\$

fig 1.4 TRANSPOSITION PAR CARRE POLYBIQUE

texte en clair (coordonnées) : « 413221311311222111132121214 »  
« 534436164451242115623236455 »

texte fractionné groupé par 2 et recombiné en coordonnées :

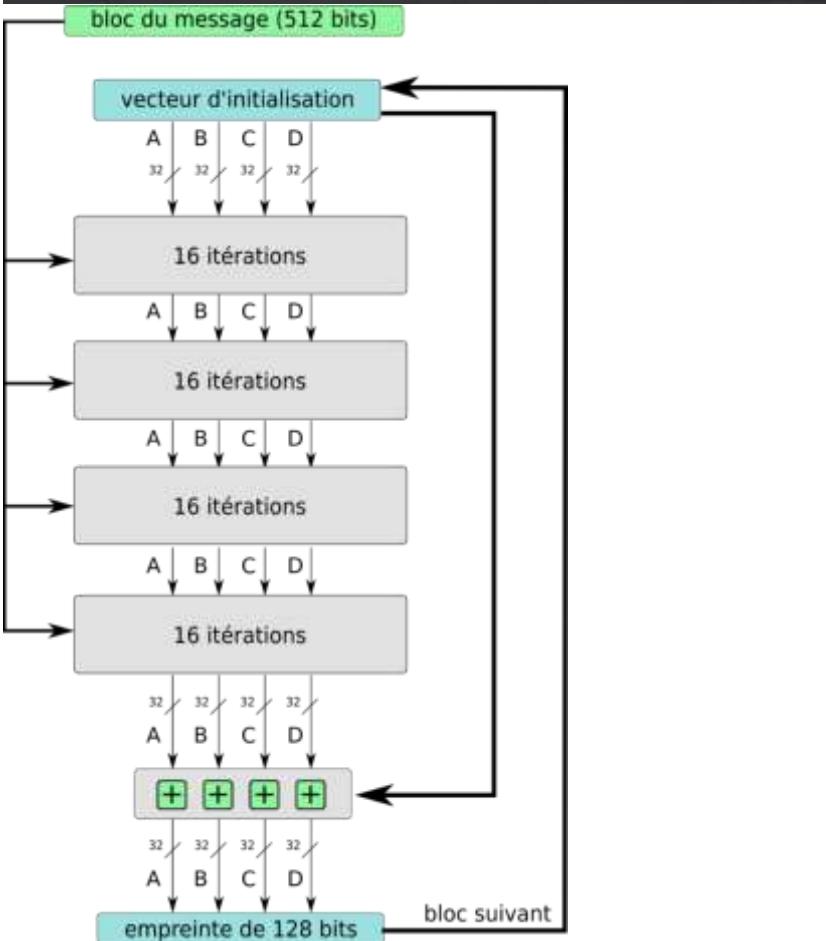
« 413221311311222111132121214534436164451242115623236455 »  
« G T E R L S F E S S T A A W U V £ % V I Q E J M T £ ' 5 »

texte chiffré avec divisions des mots :

« GTERL SFESS TAAWU V£%VI QEJMT £'5 »

Il est important de faire remarquer que les transpositions sont plus contraignantes que les substitutions, car elles ont besoin de plus de mémoire et ne fonctionnent que sur des messages à chiffrer d'une longueur limitée ; c'est pourquoi elles sont moins utilisées dans les algorithmes bien que pourtant un peu plus sûres que les substitutions.

- ▶ MD5 (*Message Digest 5*) a été introduit en 1991 par R. Rivest
- ▶ c'est une amélioration de MD4 datant de 1990, cassé puis abandonné
- ▶ en 1996, une faille est trouvée dans la mesure où l'on peut construire systématiquement 2 antécédents ayant la même empreinte
- ▶ considéré dès lors comme non sûr, il a été remplacé par SHA-1
- ▶ la taille de l'empreinte MD5 est de 128 bits donc la recherche de collisions par force brute est de  $2^{64}$  calculs d'empreinte
- ▶ or on y arrive de nos jours en  $2^{42}$  voire  $2^{30}$  calculs
- ▶ toujours utilisé pour vérifier l'intégrité d'un logiciel à télécharger (*HashCalc, md5sums*)
- ▶ avec un sel, toujours utilisé pour coder des mots de passe (GNU-LINUX, CISCO)



## 9.1 MD5

Conçu par Ronald Rivest (le R dans RSA), c'est le dernier d'une série (MD2, MD4). Cet algorithme produit un condensé de 128 bits. Il était il y a encore quelques temps l'algorithme de hachage le plus largement répandu. La cryptanalyse et l'attaque par force brute (2004) l'ont affaibli. Ses spécifications sont disponibles sur Internet dans le RFC 1321.

### 9.1.1 Vue d'ensemble

Le déroulement général de l'algorithme est illustré à la figure 9.1.

1. Complétion : ajout de padding si nécessaire afin que le message ait une longueur de  $448 \bmod 512$ . Cet ajout a toujours lieu.
2. Ajout de la longueur : on ajoute la longueur réelle du message (sur 64 bits) après les 448 bits. En conséquence, la taille totale du bloc atteint 512 bits. Si la longueur nécessite plus de 64 bits, on ne note que les 64 bits de poids faible.
3. Initialisation : initialiser 4 buffers de 32 bits chacun (A,B,C,D), qui constituent l'IV.
4. Calcul itératif : traiter le message par blocs de 512 bits. Il y a 4 rondes de 16 opérations qui sont réalisées en fonction du bloc (512), du contenu des buffers et de fonctions primitives, tel qu'illustré à la figure 9.3.
5. Le résultat final est obtenu en concaténant les résultats des additions des registres A,B,C,D avec la valeur de CV<sub>q</sub> (voir 9.2).

### 9.1.3 Fonction de compression

Chaque ronde comprend 16 itérations de la forme :

(A,B,C,D) = (D,B + ((A + g(B,C,D) + X[k] + T[i]) <<< s),B,C)

Les lettres a, b, c, d se rapportent aux 4 buffers, mais sont utilisés selon des permutations variables. Il est à remarquer que cela ne met à jour qu'un seul des buffers (de 32 bits). Après 16 étapes, chaque buffer a été mis à jour 4 fois. La fonction  $g(b, c, d)$  est une fonction non-linéaire différente dans chaque ronde (notée F,G, H et I dans la figure 9.2).  $T[i]$  est une valeur constante dérivée de la fonction sinus. (ns) représente un décalage circulaire à gauche (pour chaque buffer séparément) de s bits.

## 8.2 MD5

En 1991, Ronald Rivest améliore l'architecture de MD4 et crée MD5 (Message Digest 5).

C'est une fonction de hachage cryptographique qui permet d'obtenir pour chaque message une chaîne de 32 caractères (soit 128 bits) hexadécimaux avec une probabilité très forte que, pour deux messages différents, leurs empreintes soient différentes.

Ce quelle que soit la taille de l'information en entrée (de 0 octets à plusieurs gigas).

Cette transformation est donc irréversible (Dans le sens où on ne peut pas trouver l'information en entrée à partir d'une somme MD5).

En 1996, une faille grave (possibilité de créer des collisions à la demande) est découverte. En 2004, une équipe chinoise découvre des collisions complètes. MD5 n'est donc plus considéré comme sûr au sens cryptographique. MD5 reste encore utilisé comme outil de vérification lors des téléchargements (par exemple, en FTP). Les sites affichent encore souvent la signature en MD5 de leurs fichiers.

Le programme John the ripper permet de casser les MD5 triviaux par force brute. Des serveurs de "tables inverses" (à accès direct, et qui font parfois plusieurs gigaoctets) permettent de les casser souvent en moins d'une seconde.

Aujourd'hui, il est par exemple possible de créer des pages HTML aux contenus très différents et ayant pourtant le même MD5. La présence de codes de "bourrage" placés en commentaires, visibles seulement dans la source de la page web, trahit toutefois les pages modifiées pour usurper le MD5 d'une autre.

MD5 (« Message Digest ») est un des plus connus algorithmes de hachage. C'est une version améliorée de MD4 tous deux conçus par Ron Rivest, un des créateurs de RSA. MD5 fabrique une empreinte d'une taille de 128 bits.

### Padding

Soit un message  $m$  d'une longueur de  $n$  bits. MD5 manipule des blocs de 512 bits, l'algorithme complète le message avec un 1 suivi d'autant de 0 que nécessaires jusqu'à ce que la longueur de message soit congrue à 448 modulo 512. L'opération de padding a toujours lieu même si la longueur du message est déjà congrue à 448 modulo 512.

### Ajout de la taille

On ajoute à ce message la valeur de  $n$ , codée en binaire sur 64 bits. On obtient donc un message dont la longueur est un multiple de 512 bits.

Chaque bloc de 512 bits est décomposé en 16 blocs de 32 bits.

### Initialisation

MD5 prend 4 tampons de 32 bits en entrée initialisés de la manière suivante (en hexadécimal) :

A=01234567

B=89abcdef

C=fedcba98

D=76543210

### Rondes

MD5 est composé de quatre rondes qui exécutent chacune 16 opérations. Pour chaque ronde, une seule fonction prenant 3 arguments codés sur 32 bits et renvoyant une valeur sur 32 bits est utilisée pour les 16 opérations. Les 4 fonctions sont les suivantes :

F(X,Y,Z) = (X and Y) or (not(X) and Z)

G(X,Y,Z) = (X and Z) or (Y and not(Z))

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X or not(Z))

Pour chaque bloc de 512 bits, on effectue les opérations suivantes :

\_ on sauvegarde les valeurs des tampons (A, B, C et D) dans des registres AA, BB, CC et DD.

\_ On calcule les nouvelles valeurs pour A, B, C et D à partir de leurs anciennes valeurs, des bits du bloc qu'on étudie et une des quatre fonctions F, G, H ou I selon la ronde.

\_ On effectue  $A=AA+A$ ,  $B=BB+B$ ,  $C=CC+C$ ,  $D=DD+D$

### Écriture de l'empreinte

L'empreinte sur 128 bits est obtenue en mettant bout à bout les quatre tampons finaux A, B, C et D.

#### 9.1.4 MD4

Précurseur du MD5, il produit également des condensés de 128 bits. Il ne possède que 3 rondes de 16 étapes contre 4 dans MD5. Les buts à réaliser lors de sa conception étaient de le faire résistant aux collisions, et d'apporter une sécurité directe (c'est-à-dire qu'il n'a pas de dépendance envers des problèmes mathématiques). Il devait de plus être aussi rapide, simple et compact que possible. « MD4 » utilise des opérations « bit-à-bit » très rapides. Seule, l'addition modulo 232 est arithmétique. Elle fut ainsi construite pour être très rapide d'exécution. La sécurité de « MD4 » ne repose pas sur un problème difficile à résoudre. Comme pour le « DES », pendant longtemps, toute la sécurité était basée sur la durée pendant laquelle aucune attaque n'avait été proposée. Toutefois, très vite des chercheurs ont cryptanalysé avec succès les premières rondes de l'algorithme. En 1995, Dobbertin a même mené une attaque mettant en évidence une collision forte. Ron Rivest renforça son algorithme et le résultat de cette riposte aux attaques fut... « MD5 ».

D'après Rivest, les objectifs de conception de « MD4 » étaient les suivants : Sécurité : il est impossible de trouver par le calcul deux textes qui aient la même empreinte ; aucune attaque n'est plus efficace que l'attaque exhaustive. Sécurité directe : la sécurité de « MD4 » n'est basée sur aucune hypothèse telle que la difficulté de factorisation. Vitesse : « MD4 » est adapté pour des réalisations logicielles rapides. Simplicité : « MD4 » est aussi simple que possible sans grandes structures de données ni programme compliqué. Architecture matérielle

Amélioration de « MD5 » par rapport à « MD4 » :  
ü Une quatrième ronde a donc été ajoutée.  
ü A chaque étape « i » (il y en a  $4 \times 16 = 64$ ) est ajoutée une constante unique égale à la partie entière de  $232 \times \text{valeur absolue}(\sinus(i))$ , i étant exprimé en radian.  
ü La nouvelle fonction g est moins symétrique que celle de « MD4 ».  
ü Des effets d'avalanche ont été accélérés, au sein de chaque étape par l'ajout du résultat de l'étape précédente, par des nombres plus optimisés de décalage circulaire qui plus est différents d'une ronde à l'autre. Les résultats d'attaques, partiellement réussies, menées contre « MD5 » n'ont pas mis en cause l'utilisation de « MD5 » dans les algorithmes et produits de chiffrements.

- ▶ SHA-1 (*Secure Hash Algorithm*) a été publié en 1995 par le NIST
- ▶ a posteriori, il s'est avéré être un bon **générateur de nombre pseudo-aléatoires**
- ▶ sa fonction de compression utilise des blocs de  $b = 512$  bits pour produire un **empreinte** de 160 bits
- ▶ la recherche de collisions par force brute a une complexité en  $\mathcal{O}(2^{80})$
- ▶ des collisions ont été obtenues en  $\mathcal{O}(2^{63})$  mais l'attaque reste difficile
- ▶ SHA-1 est considéré comme théoriquement cassé, mais du fait de sa popularité et de la difficulté de l'attaque, il est encore largement répandu
- ▶ SHA-256, publié en 2000, est une variante plus robuste, avec une taille d'**empreinte** de 256 bits
- ▶ il a inspiré le chiffrement à clef secrète SHACAL-2, sans attaque efficace connue à ce jour

### 8.3 SHA-1

SHA-1 (Secure Hash Algorithm) est une fonction de hachage cryptographique concue par la NSA (1995), et publiée par le gouvernement des Etats-Unis comme un standard fédéral de traitement de l'information. Elle produit un résultat de 160 bits (40 caractères).

Même si on arrive à générer des collisions avec SHA-1. C'est-à-dire que l'on peut trouver deux messages au contenu aléatoire qui produisent la même signature. On ne sait toujours pas, à partir d'une signature donnée, forger un second message qui génère la même valeur. Or, c'est ce type d'attaque qui pourrait mettre en péril les applications comme PGP et l'authenticité des données. Des versions offrant plus de sécurité sont également disponibles : SHA-256, SHA-384 et SHA-512. Comme leur nom l'indique, ces versions fournissent des signatures de 256, 384 et 512 bits.

Exemple :

SHA-1 (Les poules se sont échappées dès qu'on avait ouvert la porte) :  
a187da360890f111566557aa6c197aa238dc533a

SHA-1 (Les poules se sont échappées des con avait ouvert la porte) :  
15166056114addee4bd717a1c45ae51389920a61 Comme vous le constatez, les deux phrases n'ont rien à voir entre elles

SHA (Secure Hash Algorithm) a été conçu par NIST et NSA en 1993, et révisé 1995 pour étendre ses

capacités en matière de sécurité. Ses spécifications sont publiées dans le RFC 3174. L'algorithme est le SHA, la norme est SHS (Secure Hash Standard). Contrairement au MD5 qui produit des condensés de 128 bits, le SHA produit des valeurs condensées de 160 bits. Jusqu'à 2005, il était l'algorithme généralement préféré pour le hachage, mais des rumeurs de cassage le font peu à peu évoluer vers des versions plus sophistiquées. Il convient aujourd'hui (2009) d'utiliser le SHA-2.

Le SHA-1 est (était) utilisé en concurrence du MD5 pour les signatures numériques (Digital Signature Algorithm) comme spécifié par le Digital Signature Standard (DSS). Pour un message de longueur inférieure à  $2^{64}$ , le SHA-1 génère un condensé de 160 bits du message appelé "hash". A nouveau, à l'identique du MD5, une modification infime du message d'origine doit avoir une grosse répercussion sur le message condensé et il ne doit pas exister de Message Digest identiques pour deux messages d'origine différents.

Comme pour le MD5, on travaille sur des messages dont la longueur est un multiple de 512 bits.

#### Etape 1 : Complétion

Si le message n'a pas une longueur de 512 bits, on rajoute autant de 0 que nécessaire à la fin de ce dernier. Les derniers 64 bits du bloc de 512 bits sont utilisés pour définir la longueur d'origine du message. On transforme ensuite le bloc de 512 bits en sous-blocs  $M[i]$  de 32 bits chacun exprimés en hexadécimal ( $0 \leq i \leq 15$ ).

#### Etape 2 : Initialisation

Comme pour le MD5, on définit cette fois 80 variables de chaînage de 32 bits  $K[i]$  initialisées ainsi (les chiffres sont hexadécimaux):

$$\begin{aligned} K[t] &= 01234567 \quad | \quad 0 \leq t \leq 19 \\ K[t] &= 89ABCDEF \quad | \quad 20 \leq t \leq 39 \\ K[t] &= FEDCBA98 \quad | \quad 40 \leq t \leq 59 \\ K[t] &= 76543210 \quad | \quad 60 \leq t \leq 79 \end{aligned}$$

On définit aussi 80 fonctions non linéaires  $F[1], F[2], \dots, F[79]$  qui prennent des arguments codés sur 32 bits, et renvoie une valeur sur 32 bits, les opérations se faisant bit à bit.

$$\begin{aligned} F[t](X,Y,Z) &= (X \text{ AND } Y) \text{ OR } (\text{NOT}(X) \text{ AND } Z) \quad | \quad 0 \leq t \leq 19 \\ F[t](X,Y,Z) &= (X \text{ XOR } Y) \text{ XOR } D \quad | \quad 20 \leq t \leq 39 \\ F[t](X,Y,Z) &= (X \text{ AND } Y) \text{ OR } (X \text{ AND } Z) \text{ OR } (Y \text{ AND } Z) \quad | \quad 40 \leq t \leq 59 \\ F[t](X,Y,Z) &= X \text{ XOR } Y \text{ XOR } Z \quad | \quad 60 \leq t \leq 79 \end{aligned}$$

Ce qu'il y a d'important avec ces 80 fonctions est que si les bits de leurs arguments X, Y et Z sont indépendants, les bits du résultat le sont aussi.

#### Etape 3 : Calcul Itératif

L'itération, utilise deux buffers, chacun consistant en l'utilisation de 5 variables de chaînage. Les variables de chaînage du premier buffer sont notées A, B, C, D, E. Le second paquet de 5 contient les variables de chaînage notées H[0], H[1], H[2], H[3], H[4].

Par ailleurs, notons  $S^n$  le décalage circulaire de n bits vers la gauche

## Processus

Le texte est complété de telle façon que sa longueur en bits soit un multiple de 512 par ajout d'un « 1 » suivi d'autant de « 0 » que nécessaire pour que seuls 64 bits manquent par rapport à un multiple de 512. Puis, on ajoute à ce texte complété 64 bits contenant la longueur du message avant remplissage.

« MD5 », comme « MD4 », utilise 4 registres (mais produit des empreintes sur 128 bits). « SHA » utilise 5 variables initialisées par les valeurs hexadécimales suivantes : A = 67452301, B = EFCDAB89, C = 98BADC9E, D = 10325476 et E = C3D2E1F0.

L'algorithme traite le texte par bloc de 512 bits et utilise des fonctions non-linéaires différentes en fonctions de l'opération :

$$f_t(X, Y, Z) = (X \wedge Y) \vee ((\neg X) \wedge Z) \text{ pour } t \text{ entre } 0 \text{ et } 19$$

$$f_t(X, Y, Z) = X \oplus Y \oplus Z \text{ pour } t \text{ entre } 20 \text{ et } 39$$

$$f_t(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \text{ pour } t \text{ entre } 40 \text{ et } 59$$

$$f_t(X, Y, Z) = X \oplus Y \oplus Z \text{ pour } t \text{ entre } 60 \text{ et } 79.$$

La boucle principale comprend 4 rondes de 20 opérations (« MD5 » compte, elle, 4 rondes de 16 étapes) et utilise 4 constantes hexadécimales  $K_t$  : 5A827999, 6ED9EBA1, 8F1BBCDC et CA62C1D6 égalant respectivement à  $\sqrt{2}/4$ ,  $\sqrt{3}/4$ ,  $\sqrt{5}/4$  et  $\sqrt{10}/4$ ...

- ▶ KECCAK est une fonction de hachage inventée par G. Bertoni, J. Daemen, M. Peeters, et G. Van Assche
- ▶ proposé à la compétition SHA-3, KECCAK a été choisi par le NIST en octobre 2012
- ▶ ce n'est pas forcément pour remplacer SHA-2 sans attaque significative ni SHA-1, le but était de faire émerger une fonction de hachage différente dans sa conception
- ▶ SHA-3 utilise une construction en éponge nouvelle !
- ▶ la taille des empreintes y est variable, de 224 à 512 bits.

- SHA-1 (Secure Hash Algorithm 1), comme MD5, est basé sur MD4. Il fonctionne également à partir de blocs de 512 bits de données et produit par contre des condensés de 160 bits en sortie. Il nécessite donc plus de ressources que MD5.
- SHA-2 (Secure Hash Algorithm 2) a été publié récemment et est destiné à remplacer SHA-1. Les différences principales résident dans les tailles de hachés possibles : 256, 384 ou 512 bits. Il sera bientôt la nouvelle référence en termes de fonction de hachage

Ron Rivest a rendu public les choix de conception de « MD5 ». Le « NIST » et la « NSA » ne l'ont pas fait... « SHA », similaire à « MD4 », met en œuvre toutefois une étape d'expansion, la sommation de l'étape précédente à l'entrée de l'étape suivante afin d'obtenir un « effet d'avalanche ». Une quatrième ronde a également été ajoutée mais dans le « SHA » la fonction utilisée est la même ce qui n'est pas le cas de « MD5 ».

MD5      mélange de bits amélioré ; une ronde supplémentaire ; meilleur effet d'avalanche ; empreinte sur 128 bits

MD4

SHA      addition d'une transformation d'expansion ; une ronde supplémentaire ; meilleur effet d'avalanche même par rapport à « MD5 » ; empreinte sur 160 bits  $\Rightarrow$  meilleure résistance à une attaque massive

Le National Institute of Standards and Technology, ou NIST (qu'on pourrait traduire par « Institut national des normes et de la technologie »), est une agence du département du Commerce des États-Unis. Son but est de promouvoir l'économie en développant des technologies, la métrieologie et des standards de concert avec l'industrie. Cette agence a pris la suite en 1988 du National Bureau of Standards fondé en 1901 avec substantiellement les mêmes missions.

Les algorithmes de la famille SHA-2 sont très semblables, il y a essentiellement deux fonctions différentes, SHA-256 et SHA-512, les autres étant des variantes de l'une ou l'autre. Les fonctions SHA-256 et SHA-512 ont la même structure mais diffèrent par la taille des mots et des blocs utilisés. Cette structure est assez proche de celle de SHA-1, mais un peu plus complexe et en évite certaines faiblesses connues. Elle se rattache plus généralement à une famille de fonctions de hachage inspirées de [MD4](#) et [MD5](#) de [Ron Rivest](#). On retrouve comme primitives l'addition pour des entiers de taille fixe  $n$  soit une addition modulo  $2^n$ , opération non linéaire (au sens de l'[algèbre linéaire](#)) sur le corps des booléens  $\mathbf{F}_2$ , ainsi que des [opérations bit à bit](#) (xor et autres).

Comme toutes les fonctions de cette famille (sauf SHA-3, reposant sur une [fonction éponge](#)), elles suivent un schéma itératif qui suit la [construction de Merkle-Damgård](#) (sans opération de finalisation). La [fonction de compression](#) itérée possède deux entrées de taille fixe, la seconde entrée étant de même taille que la sortie de la fonction :

- une donnée obtenue par découpage du message à traiter, la taille est de 512 bits pour SHA-256 et SHA-224 et de 1024 bits pour SHA-512 et SHA-384,
- le résultat de la fonction de compression à l'itération précédente (256 bits pour SHA-256 et SHA-224, 512 pour SHA-512 et SHA-384).

Les entrées de la fonction de compression sont découpées

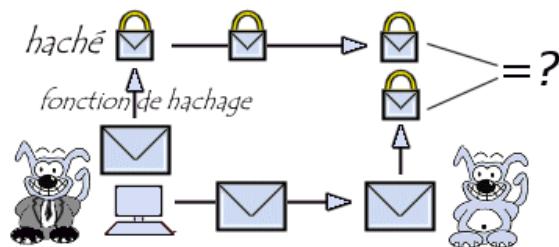
- en mots de 32 bits pour SHA-256 et SHA-224,
- en mots de 64 bits pour SHA-512 et SHA-384.

La fonction de compression répète les mêmes opérations un nombre de fois déterminé, on parle de *tour* ou de *ronde*, 64 tours pour SHA-256, 80 tours pour SHA-512. Chaque tour fait intervenir comme primitives l'addition entière pour des entiers de taille fixe, soit une addition modulo  $2^{32}$  ou modulo  $2^{64}$ , des opérations bit à bit : opérations logiques, décalages avec perte d'une partie des bits et [décalages circulaires](#), et des constantes prédéfinies, utilisées également pour l'initialisation.

Avant traitement, le message est complété par [bourrage](#) de façon que sa longueur soit un multiple de la taille du bloc traité par la fonction de compression. Le bourrage incorpore la longueur (en binaire) du mot à traiter : c'est le renforcement de Merkle-Damgård ([\(en\)](#) *Merkle-Damgård strengthening*), ce qui permet de réduire la [résistance aux collisions](#) de la fonction de hachage à celle de la fonction de compression. Cette longueur est stockée en fin de bourrage sur 64 bits dans le cas de SHA-256 (comme pour SHA-1), sur 128 bits dans le cas de SHA-512, ce qui « limite » la taille des messages à traiter à  $2^{64}$  bits pour SHA-256 (et SHA-224) et à  $2^{128}$  bits pour SHA-512 (et SHA-384).

## Vérification d'intégrité

En expédiant un message accompagné de son haché, il est possible de garantir l'intégrité d'un message, c'est-à-dire que le destinataire peut vérifier que le message n'a pas été altéré (intentionnellement ou de manière fortuite) durant la communication.

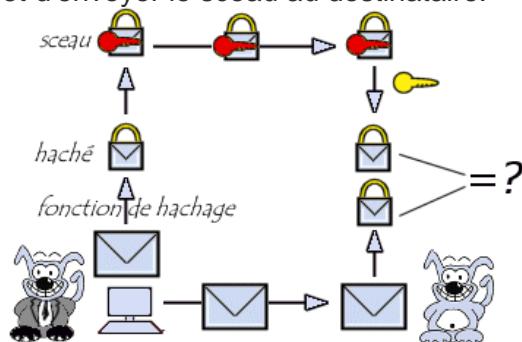


Lors de la réception du message, il suffit au destinataire de calculer le haché du message reçu et de le comparer avec le haché accompagnant le document. Si le message (ou le haché) a été falsifié durant la communication, les deux empreintes ne correspondront pas.

### Le scellement des données

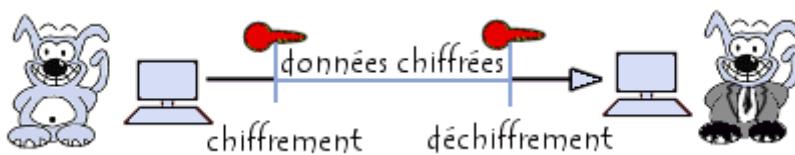
L'utilisation d'une fonction de hachage permet de vérifier que l'empreinte correspond bien au message reçu, mais rien ne prouve que le message a bien été envoyé par celui que l'on croit être l'expéditeur.

Ainsi, pour garantir l'authentification du message, il suffit à l'expéditeur de chiffrer (on dit généralement *signer*) le condensé à l'aide de sa clé privée (le *haché signé* est appelé **sceau**) et d'envoyer le sceau au destinataire.



À réception du message, il suffit au destinataire de déchiffrer le sceau avec la clé publique de l'expéditeur, puis de comparer le haché obtenu avec la fonction de hachage au haché reçu en pièce jointe. Ce mécanisme de création de sceau est appelé *scellement*.

**Le chiffrement symétrique** (aussi appelé *chiffrement à clé privée* ou *chiffrement à clé secrète*) consiste à utiliser la même clé pour le chiffrement et le déchiffrement.



Le chiffrement consiste à appliquer une opération (algorithme) sur les données à chiffrer à l'aide de la clé privée, afin de les rendre inintelligibles. Ainsi, le moindre algorithme (tel qu'un OU exclusif) peut rendre le système quasiment inviolable (la sécurité absolue n'existant pas).

Toutefois, dans les années 40, *Claude Shannon* démontra que pour être totalement sûr, les systèmes à clefs privées doivent utiliser des clefs d'une longueur au moins égale à celle du message à chiffrer. De plus le chiffrement symétrique impose d'avoir un canal sécurisé pour l'échange de la clé, ce qui dégrade sérieusement l'intérêt d'un tel système de chiffrement.

Le principal inconvénient d'un cryptosystème à clefs secrètes provient de l'échange des clés. En effet, le chiffrement symétrique repose sur l'échange d'un secret (les clés). Ainsi, se pose le problème de la distribution des clés :

D'autre part, un utilisateur souhaitant communiquer avec plusieurs personnes en assurant de niveaux de confidentialité distincts doit utiliser autant de clés privées qu'il a d'interlocuteurs. Pour un groupe de  $N$  personnes utilisant un cryptosystème à clés secrètes, il est nécessaire de distribuer un nombre de clés égal à  $N * (N-1) / 2$ .

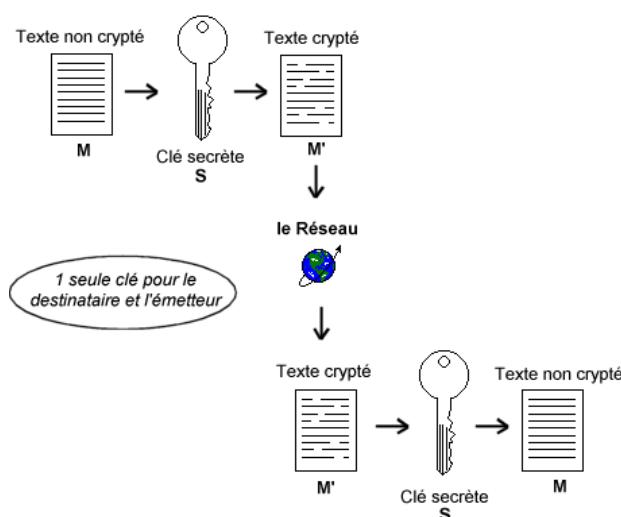
Ainsi, dans les années 20, *Gilbert Vernam* et *Joseph Mauborgne* mirent au point la méthode du *One Time Pad* (traduisez *méthode du masque jetable*, parfois appelé « One Time Password » et noté *OTP*), basée sur une clé privée, générée aléatoirement, utilisée une et une seule fois, puis détruite. À la même époque, le Kremlin et la Maison Blanche étaient reliés par le fameux **téléphone rouge**, c'est-à-dire un téléphone dont les communications étaient cryptées grâce à une clé privée selon la méthode du *masque jetable*. La clé privée était alors échangée grâce à la valise diplomatique (jouant le rôle de canal sécurisé).

**La même clé est utilisée pour coder et décoder et doit bien évidemment rester secrète.**

Un algorithme répandu consiste à effectuer des substitutions et des transpositions en cascade sur les lettres du message. Par exemple : MICROAPPLICATION devient IRMCAPOIALCINTO si l'on fixe comme clé le principe suivant : "la première lettre sera la troisième, la seconde sera la première, la troisième sera la quatrième et la quatrième sera la seconde".

La figure suivante illustre le principe du chiffrement symétrique.

**Figure 23.1. Chiffrement symétrique**



Quelques algorithmes couramment utilisés (liste non exhaustive) :

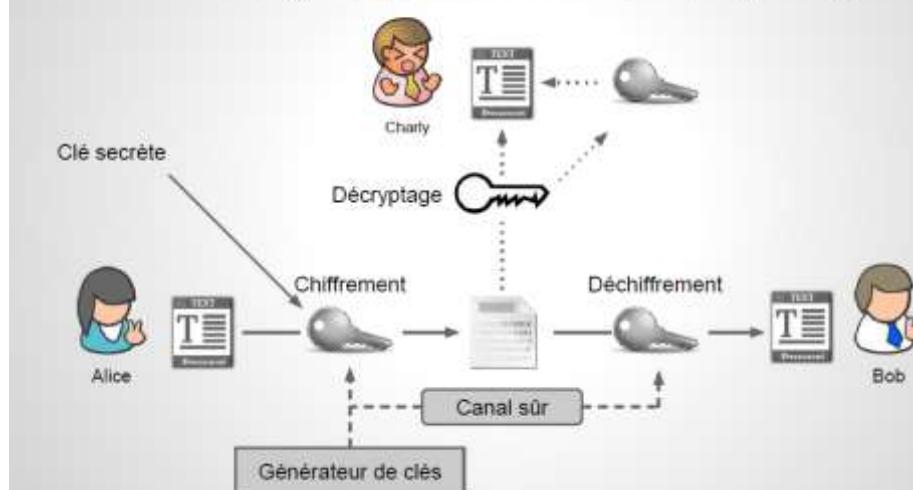
- Data Encryption Standard (DES) est une invention du National Bureau of Standards (NSB) américain, qui date de 1977 : c'est le premier algorithme de chiffrement public gratuit. Les données sont découpées en blocs de 64 bits et codées grâce à la clé secrète de 56 bits propre à un couple d'utilisateurs
- RC2, RC4 et RC5 sont des algorithmes créés à partir de 1989 par Ronald Rivest pour la RSA Security. L'acronyme "RC" signifie "Ron's Code" ou "Rivest's Cipher". Ce procédé permet aux utilisateurs la possibilité de choisir la grandeur de la clé (jusqu'à 1024 bits).
- Advanced Encryption Standard (AES) est un standard approuvé par le ministère américain du commerce en 2001 et vise à remplacer le DES ; il est placé dans le domaine public et accepte des clés d'une taille de 128, 192 ou 256 bits.

Les inconvénients de ce système de chiffrement sont les suivants :

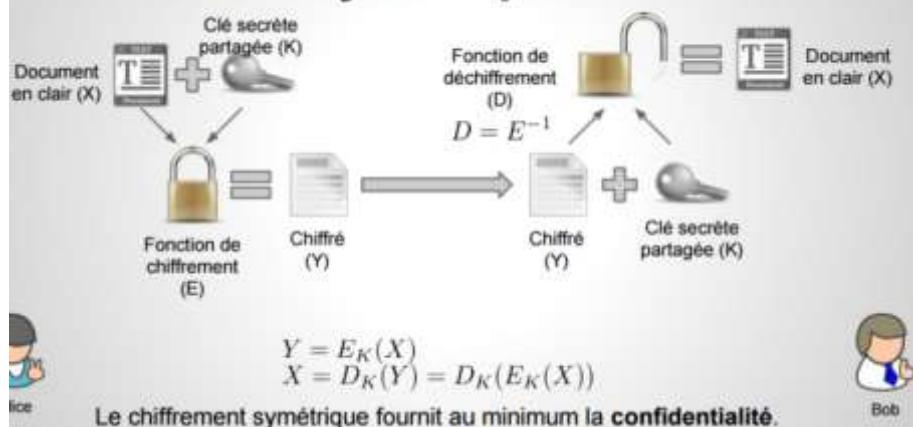
- il faut autant de paires de clés que de couples de correspondants ;
- la non-répudiation n'est pas assurée. Mon correspondant possédant la même clé que moi, il peut fabriquer un message en usurpant mon identité ;
- il faut pouvoir se transmettre la clé au départ sans avoir à utiliser le média à sécuriser !

La cryptographie à clefs privées, appelée aussi cryptographie symétrique est utilisée depuis déjà plusieurs siècles. C'est l'approche la plus authentique du chiffrement de données et mathématiquement la moins problématique. La clef servant à chiffrer les données peut être facilement déterminée si l'on connaît la clef servant à déchiffrer et vice-versa. Dans la plupart des systèmes symétriques, la clef de cryptage et la clef de décryptage sont une seule et même clef. Les principaux types de cryptosystèmes à clefs privés utilisés aujourd'hui se répartissent en deux grandes catégories : les cryptosystèmes par flots et les cryptosystèmes par blocs.

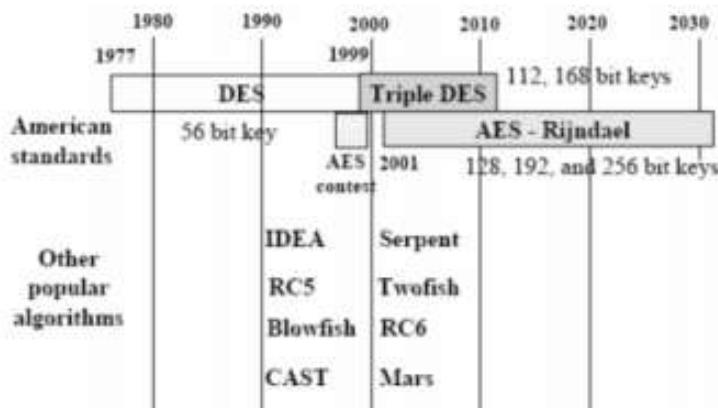
## Terminologie (chiffrement sym.)



## Chiffrement symétrique



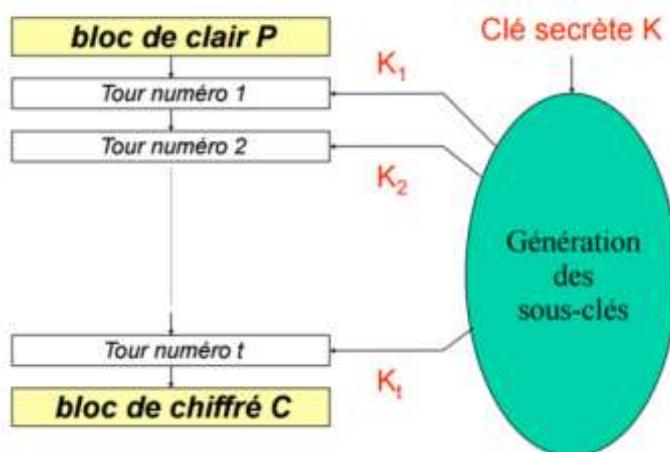
## Chiffrement symétriques les plus fréquents



Deux principaux paramètres de sécurité :— La taille du bloc (e.g. n = 64 ou 128 bits). Les modes opératoires permettent généralement des attaques quand plus de  $2n/2$  blocs sont chiffrés avec une même clé – La taille de clé (e.g. k = 128 bits). Pour un bon algorithme, la meilleure attaque doit coûter  $2^k$  opérations (recherche exhaustive)

CONSTRUCTION • Algorithmes itératifs : une fonction de tour est itérée t fois • Génération de clés de tour (ou sous-clés) à partir de la clé secrète K • Utilisation d'opérations simples et efficaces (+, XOR, \*, tableaux)

## Construction



Effet d'avalanche : Propriété des chiffrements par blocs composés de couches (layers) ou "rounds" avec un petit changement à l'entrée. Le changement d'un simple bit d'entrée produit généralement de multiples changements de bits après un round, plusieurs autres changements de bits après un autre round jusqu'au changement éventuel de la moitié du bloc.

Dans un cryptosystème par flots, le cryptage des messages se fait caractère par caractère ou bit à bit, au moyen de substitutions de type César générées aléatoirement : la taille de la clef est donc égale à la taille du message. L'exemple le plus illustratif de ce principe est le chiffre de Vernam. Cet algorithme est aussi appelé « One Time Pad » (masque jetable), c'est à dire que la clef n'est utilisée qu'une seule fois.

Voici un exemple simple de l'application du chiffre de Vernam :

Exemple:

Message en clair: "**SALUT**"

=> (conversion en binaire)

01010011 01000001 01001100 01010101 01010100

XOR

Clef (générée aléatoirement)

01110111 01110111 00100100 00011111 00011010

=

00100100 00110110 01101000 01001010 01001110

## => (conversion en caractère) "Message chiffré: \$6jJM"

La deuxième classe de cryptosystèmes utilisée aujourd’hui est celle des cryptosystèmes par blocs. Dans ce mode de cryptage, le texte clair est fractionné en blocs de même longueur à l'aide d'une clef unique. Les algorithmes de chiffrement par blocs sont en général construits sur un modèle itératif. Ce modèle emploie une fonction F qui prend en paramètres une clef k et un message de n bits. F est répétée un certain nombre de fois, on parle de ronde. A chaque ronde, la clef k utilisée est changée et le message que l'on chiffre est le résultat de l'itération précédente.

$$C_1 = F(k_1, M)$$

$$C_2 = F(k_2, C_1)$$

$$\dots C_r = F(k_r, C_{r-1})$$

Emetteur et destinataire se partagent une clé K secrète. L'algorithme qui engendre les clefs  $k_i$  à partir de K se nomme l'algorithme de cadencement des clefs.

La fonction F doit être inversible, ce qui veut dire qu'il faut pour toute clef k et message M pouvoir recalculer M à partir de  $F(k', M)$ , sinon le déchiffrement est impossible et on ne dispose pas d'un algorithme utilisable. C'est-à-dire qu'il existe une fonction G vérifiant  $G(k, F(k, M)) = M$  et que F est une permutation.

La sécurité d'un algorithme de chiffrement par blocs réside principalement dans la conception de l'algorithme de cadencement des clefs et la robustesse de la fonction F. Si l'algorithme de cadencement est mal élaboré, les  $k_i$  peuvent être déductibles les unes des autres. La fonction F doit donc être difficile à inverser sans connaître la clef k ayant servi dans le calcul de  $C = F(k, M)$ . En d'autres termes, connaissant seulement C, F et G, on ne doit pouvoir retrouver le message M seulement en effectuant une recherche exhaustive de la clef.

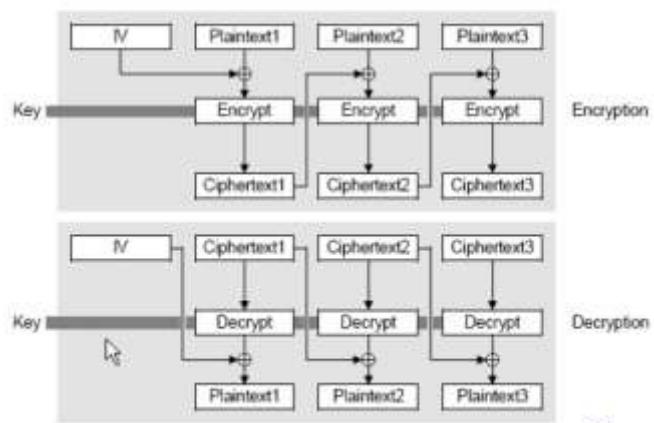
Les caractéristiques de ces systèmes sont en général liées à leur très forte sensibilité à la dépendance inter-symboles, ainsi qu'à leur mécanisme de propagation d'erreurs. Toute erreur commise sur un bloc de texte clair ou chiffré peut perturber gravement le chiffrement/déchiffrement de ses voisins.

**Le chiffrement par bloc (block cipher) est une des deux grandes catégories de chiffrements modernes en cryptographie symétrique**

Le principe consiste à un découpage des données en blocs de taille généralement fixe (souvent une puissance de deux comprise entre 32 et 512 bits). Les blocs sont ensuite chiffrés les uns après les autres. Il est possible de transformer un chiffrement de bloc en un chiffrement par flot en utilisant un mode d'opération comme ECB (chaque bloc chiffré indépendamment des autres) ou CFB (on chaîne le chiffrement en effectuant un XOR entre les résultats successifs).

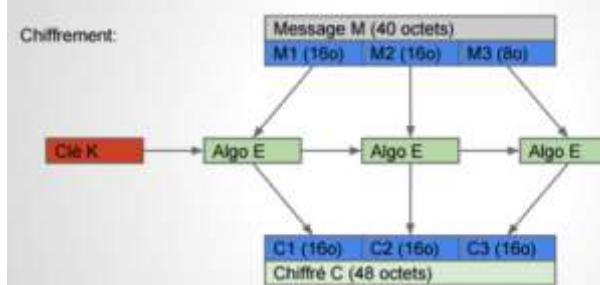
Une liste non-exhaustive :

- DES, l'ancêtre
- AES, le remplaçant de DES
- Blowfish et Twofish, des alternatives à AES

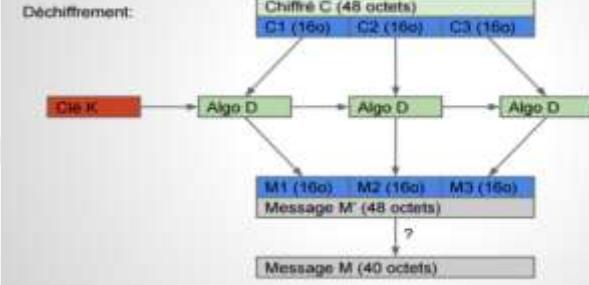


Le chiffrement par bloc permet de travailler sur des blocs de taille fixée (le plus souvent 16 octets). On traite un message long comme une succession de blocs. Ces algorithmes nécessitent en général d'ajouter du bourrage lorsque le message initial n'est pas un multiple de la taille de blocs

## Par blocs / par flot



## Par blocs / par flot



L'idée générale du chiffrement par blocs est la suivante:

Remplacer les caractères par un code binaire

Découper cette chaîne en blocs de longueur donnée

Chiffrer un bloc en l'"additionnant" bit par bit à une clé.

Déplacer certains bits du bloc. Recommencer éventuellement un certain nombre de fois l'opération

3. On appelle cela une ronde.

Passer au bloc suivant et retourner au point 3 jusqu'à ce que tout le message soit chiffré.

### Chiffrement par substitution

Les substitutions consistent à remplacer des symboles ou des groupes de symboles par d'autres symboles ou groupes de symboles dans le but de créer de la confusion.

### Chiffrement par transposition

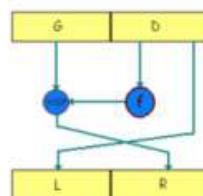
Les transpositions consistent à mélanger les symboles ou les groupes de symboles d'un message clair suivant des règles prédéfinies pour créer de la diffusion. Ces règles sont déterminées par la clé de chiffrement. Une suite de transpositions forment une permutation.

**Chiffrement par produit :** la combinaison des deux Le chiffrement par substitution ou par transposition ne fournit pas un haut niveau de sécurité, mais en combinant ces deux transformations, on peut obtenir un chiffrement robuste. La plupart des algorithmes de cryptage par clés symétriques utilisent le chiffrement par produit. Un « round » est complété lorsque les deux transformations ont été faites une fois (substitution et transposition).

- Structure décrite en 1973 (Feistel – IBM)
- Adapte la structure de Shannon afin de rendre la structure inversible ce qui permet de réutiliser le matériel de chiffrement pour déchiffrer un message. La seule modification s'effectue sur la manière dont la clé est utilisée
- Une couche de S-Box et de P-Box est utilisée par Round

### Feistel - principe

- Le texte est chiffré à partir de la même transformation sur le texte clair dans chaque round.



Chiffrement symétrique - DES - 8

Description du schéma :

Dans une construction de Feistel, le bloc d'entrée d'un round est séparé en deux parties.

- La fonction de chiffrement est appliquée sur la première partie du bloc
- et l'opération binaire OU-Exclusif est appliquée sur la partie sortante de la fonction et la deuxième partie.
- Ensuite les deux parties sont permutees et le prochain round commence.

Une des techniques utilisé dans les algorithmes de chiffrement par bloc est le réseau de Feistel. Il s'agit d'une succession d'étapes semblables ("rondes" ou "tours"). On effectue une opération sur la moitié des données, dont le résultat est combiné à l'autre moitié, et on inverse les deux parties. Un réseau de Feistel est constitué de plusieurs tours successifs. au final on obtient une bijection partant des N bits d'entrée vers N bits de sortie.

L'avantage est que la fonction de chiffrement et la fonction de déchiffrement sont identiques. Ainsi la fonction n'a pas à être inversible, c'est la structure qui l'est.

## Feistel – Choix des paramètres

- La réalisation d'un tel réseau dépend des choix effectués pour les paramètres suivants :
  - Taille du bloc :
    - ↗ taille → ↗ sécurité
  - Taille de clé :
    - ↗ taille → ↗ sécurité
  - Nombre de cycle :
    - ↗ nombre → ↗ sécurité
  - Algorithme de génération des sous-clés :
    - ↗ complexité → ↗ difficultés de cryptanalyse
  - Vitesse de chiffrement/déchiffrement logiciel

Quatre modes de chiffrement par bloc (ou code d'identification de messages) sont utilisés :

Electronic CodeBook (ECB), Cipher Block Chaining (CBC), Cipher FeedBack (CFB) ou Output FeedBack (OFB).

Les quatres modes cités précédemment sont plus ou moins indépendants de l'algorithme choisi.

Toutefois, tout les algorithmes ne permettent pas d'utiliser tout les modes possibles.

Le mode Electronic CodeBook (ECB) est le plus simple des modes et s'applique aux block ciphers. Il revient à crypter un bloc indépendamment des autres; cela permet entre autre de crypter suivant un ordre aléatoire (bases de données, etc...) mais en contre-partie, ce mode est très vulnérable aux attaques. Il est par exemple possible de recenser tous les cryptés possibles (code books) puis par recoupements et analyses statistiques recomposer une partie du message original sans avoir tenté de casser la clé de chiffrement. Il demeure que si la clé fait 128 bits ou plus, cette attaque n'es pas exploitable en pratique de nos jours.

Le mode Cipher Block Chaining (CBC) peut-être utilisé par les algorithmes en bloc. C'est d'ailleurs le mode le plus courant. Il permet d'introduire une complexité supplémentaire dans le processus de cryptage en créant une dépendance entre les blocs successifs; autrement dit, le cryptage d'un bloc va être -d'une manière ou d'une autre- lié à ou aux blocs/chiffrés précédents.

Le mode Cipher FeedBack (CFB) est un mode destiné aux block ciphers dans le but d'en autoriser une utilisation plus souple, qui s'apparente plus à celle des algorithmes en continu. On peut le considérer comme un intermédiaire entre les deux. En effet, en partant d'un algorithme en bloc utilisant une longueur standard de n bits/blocs, le mode CFB va permettre de crypter des blocs dont la longueur pourra varier de n à 1 bits/blocs.

Le mode Output FeedBack (OFB) est une variante de mode CFB précédemment abordé. Il est d'ailleurs parfois appelé internal feedback. Il présente beaucoup de problèmes de sécurité et il est peu conseillé sauf dans le cas où sa longueur est égale à celle de l'algorithme utilisé.

Le **chiffrement de flux** ou **chiffrement par flot** (stream cipher) est une des deux grandes catégories de chiffrements modernes en cryptographie symétrique. Un chiffrement par flot arrive à traiter les données de longueur quelconque et n'a pas besoin de les découper.

Une liste non-exhaustive de chiffrements par flot :

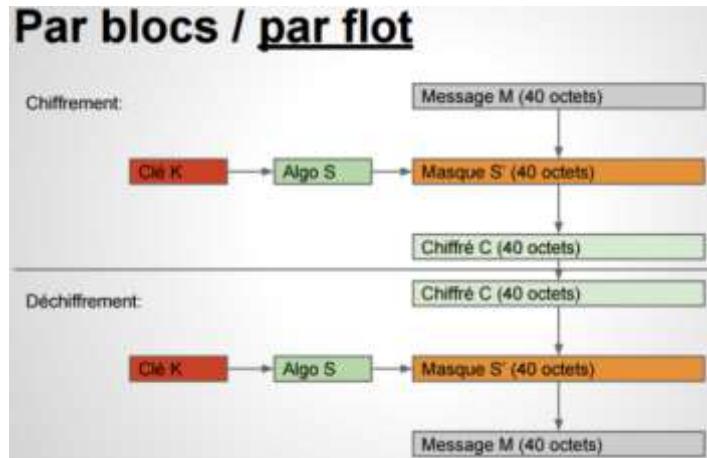
- A5, utilisé dans les téléphones mobiles de type GSM pour chiffrer la communication par radio entre le mobile et l'antenne-relais la plus proche,
- RC4, le plus répandu, conçu par Ronald Rivest, utilisé notamment par le protocole WEP du WiFi
- Py, un algorithme récent de Eli Biham
- E0 utilisé par le protocole Bluetooth

Un chiffrement par flot se présente souvent sous la forme d'un générateur de nombres pseudo-aléatoires avec lequel on opère un XOR entre un bit à la sortie du générateur et un bit provenant des données. Toutefois, le XOR n'est pas la seule opération possible. L'opération d'addition dans un groupe est également envisageable (par exemple, addition entre deux octets, modulo 256). Un chiffrement par bloc peut être converti en un chiffrement par flot grâce à un mode opératoire qui permet de chaîner plusieurs blocs et traiter des données de taille quelconque.

### Chiffrement/déchiffrement avec XOR

- Soit l'opération booléenne XOR :
- Chiffrement du message M avec la clé K :
- Déchiffrement du message C avec la clé K :

## Par blocs / par flot



Chiffrement par bloc (block cipher) • DES (Data Encryption Standard): bloc=64bits, clé=56bits • AES (Advanced Encryption Standard): bloc=128bits, clé=128/192/256bits • Blowfish: bloc=64bits, clé=32..448bits

Chiffrement par flot (ou chiffrement en continu, stream cipher) • A5/1 (chiffrement GSM): clé=64bits (seulement 54 pour le GSM) • RC4 (WEP): clé variable • E0 (Bluetooth): clé=128bits en général

Il existe deux types de chiffrement à clé symétrique :

- Le chiffrement par blocs : l'opération de chiffrement s'effectue sur des blocs de texte clair (ex : le DES avec des blocs de 64 bits).
- Le chiffrement par flots (ou par stream ou de flux) : l'opération de chiffrement s'opère sur chaque élément du texte clair (caractère, bits). On chiffre un bit/caractère à la fois. La structure d'un chiffrement par stream repose sur un générateur de clés qui produit une séquence de clés k<sub>1</sub>, k<sub>2</sub>, ..., k<sub>i</sub>.

## MECANISME CRYPTO

**L'authentification de personne**, aussi appelée identification, permet à une entité de prouver qu'elle est bien celle qu'elle prétend être. La technique du login/mot de passe n'est pas sûre car un adversaire qui écoute le canal peut rejouer cette authentification et se faire passer pour l'utilisateur. Un protocole d'authentification entre un utilisateur et un vérificateur est un ensemble de questions posées par le vérificateur auxquelles l'utilisateur doit être capable de répondre. L'adversaire peut être simplement passif, c'est-à-dire qu'il écoute des communications entre un utilisateur et un vérificateur honnêtes et tente ensuite de se faire passer pour l'utilisateur à partir de ces seules informations.

Mais il peut être aussi actif : dans ce cas, il peut prendre la place du vérificateur et poser des questions à l'utilisateur pour essayer d'extraire de l'information sur son secret. Idéalement, une preuve d'authentification doit permettre à un utilisateur de prouver qu'il connaît un secret associé à une clé publique tout en assurant que le vérificateur n'obtient aucun bit d'information sur le secret et ne peut pas ultérieurement se faire passer pour l'utilisateur. Une preuve d'identité, pour être sûre, ne doit pas être transférable. La cryptographie à clé publique permet de construire de tels schémas en utilisant des preuves zero knowledge. La cryptographie symétrique propose des méthodes pour l'identification mais les garanties de sécurité ne sont pas aussi fortes que celles apportées par les techniques zero knowledge. Dans la suite, on décrit le système IFF et le système OTP.

Le protocole **IFF** (Identification Friends and Foes) permet de construire un schéma d'identification sûr en pratique mais qui n'est pas zero knowledge. L'utilisateur et le vérificateur partagent une clé secrète  $k$  pour l'algorithme DES par exemple. Le vérificateur envoie un challenge aléatoire  $r$  pris dans l'ensemble  $\{0, 1\}^{64}$  et l'utilisateur répond par  $y = E_k(r)$ . En réception, le vérificateur calcule avec sa clé  $y' = E_k(r)$  et contrôle que si  $y = y'$ . Il est évident que si le vérificateur ne choisit pas les challenges de manière aléatoire dans  $\{0, 1\}^{64}$  et s'il redemande une ancienne valeur du challenge déjà envoyé, alors une personne qui aura espionné le canal pourra s'identifier.

Il existe un autre mode d'identification appelé One-Time Password (**OTP**) basé sur des fonctions de hachage supposées se comporter comme des fonctions à sens unique (§ 3). L'idée consiste à utiliser un mot de passe  $pwd$  et à le hacher  $t$  fois. La dernière valeur  $ht = H^t(pwd)$  est conservée par le serveur. Lorsque le client veut s'authentifier, il calcule à l'aide de son mot de passe,  $ht^{-1} = H^{t-1}(pwd)$  et le serveur vérifie si la correspondance est bonne.

En cryptographie, on a souvent besoin **d'aléas pour générer** des clés ou pour randomiser les schémas de signature ou de chiffrement. En général, les générateurs aléatoires mixent diverses sources d'aléas (horloge, copie d'écran, numéro de processus, mouvements de la souris, intervalle entre plusieurs frappes...) dans une première phase. Ensuite, un traitement algorithmique permet de lisser la distribution de sortie du générateur en utilisant par exemple des fonctions de hachage ou des algorithmes de chiffrement par bloc. Il est difficile de générer des bits aléatoires. Un générateur de bits pseudo-aléatoires (pseudorandom bit generator : PRBG) est un algorithme déterministe qui, étant donné une suite binaire de longueur  $k$ , produit une suite binaire de longueur qui apparaît aléatoire. L'entrée du PRBG est appelée le germe, alors que la sortie du PRBG est appelée une suite de bits pseudo-aléatoires.

Le **masque jetable** combine le message en clair avec une clé.

- \_ La clé doit être une suite de caractères aussi longue que le message à chiffrer.
- \_ Les caractères composant la clé doivent être choisis de façon totalement aléatoire.
- \_ Chaque clé, ou "masque", ne doit être utilisée qu'une seule fois (d'où le nom de masque jetable).
- \_ Interêt : **sécurité théorique absolue** (C. Shannon 1949).

\* **Comment générer l'aléa?**

Système de chiffrement à flot

\* **Chiffrement à flot : générateurs aléatoires**

Un candidat naturel (rapide) : registre à décalage. (**LFSR**: Linear Feedback Shift Register )

Dans tous les cas, la sécurité d'un système de chiffrement symétrique doit reposer sur la clé utilisée (appelée **clé secrète**). • Elle est partagée par toutes les entités devant pouvoir chiffrer/déchiffrer • Il n'est pas possible de distinguer les deux rôles • Une clé secrète doit circuler le moins possible (idéalement: pas du tout). o Il existe des algorithmes permettant de calculer une clé secrète sur des systèmes distants (Diffie-Hellman, S2S...) o En cas d'envoi, on utilise généralement un chiffrement asymétrique pour protéger une clé secrète.

Les algorithmes de chiffrement symétriques sont **utilisé** partout où l'on souhaite de la confidentialité, et où les algorithmes asymétriques ne peuvent être utilisés (c'est à dire, dès que l'on dépasse le kilo-octet...). Exemple: • Chiffrement de communication: on utilise une clé secrète "jetable", unique pour chaque échange. On peut alors maintenir un flux chiffré avec de bonnes performances. • Stockage sécurisé: on utilise des clés secrètes (par exemple, une par fichier). Ces clés sont elles-mêmes stockées dans un tressailloir sécurisé par d'autres moyens.

## DES

En mai 1973, le National Bureau of Standards américain demande la création d'un chiffrement utilisable par les entreprises. À cette époque, IBM dispose déjà de Lucifer, l'algorithme d'Horst Feistel. En bonne logique, cet algorithme aurait dû être sélectionné par le NBS. En pratique, ce fut presque le cas : la NSA demanda à ce que Lucifer soit modifié, par ses soins. Ainsi fut créé le DES (Data Encryption Standard), qui fut adopté comme standard en novembre 1976.

Cela suscita des rumeurs selon lesquelles la NSA aurait volontairement affaibli l'algorithme, dans le but de pouvoir le casser. (L'algorithme initialement conçu par IBM utilisait une clé de 112 bits. L'intervention de la NSA a ramené la taille de clé à 56 bits.)

DES a été l'algorithme « officiel » de l'administration américaine jusqu'en 1999. C'est à dire que tout les document confidentiel defense et secret defense étaient crypté DES.

Le système Unix qui date de cette période utilise encore le DES pour crypter les mots de passe.

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

- posséder un haut niveau de sécurité lié à une clé de petite taille servant au chiffrement et au déchiffrement
- être compréhensible
- ne pas dépendre de la confidentialité de l'algorithme
- être adaptable et économique
- être efficace et exportable

Fin 1974, IBM propose « Lucifer », qui, grâce à la NSA (National Security Agency), est modifié le 23 novembre 1976 pour donner le **DES** (*Data Encryption Standard*). Le DES a finalement été approuvé en 1978 par le NBS. Le DES fut normalisé par l'*ANSI* (*American National Standard Institute*) sous le nom de *ANSI X3.92*, plus connu sous la dénomination *DEA* (*Data Encryption Algorithm*).

## Principe du DES

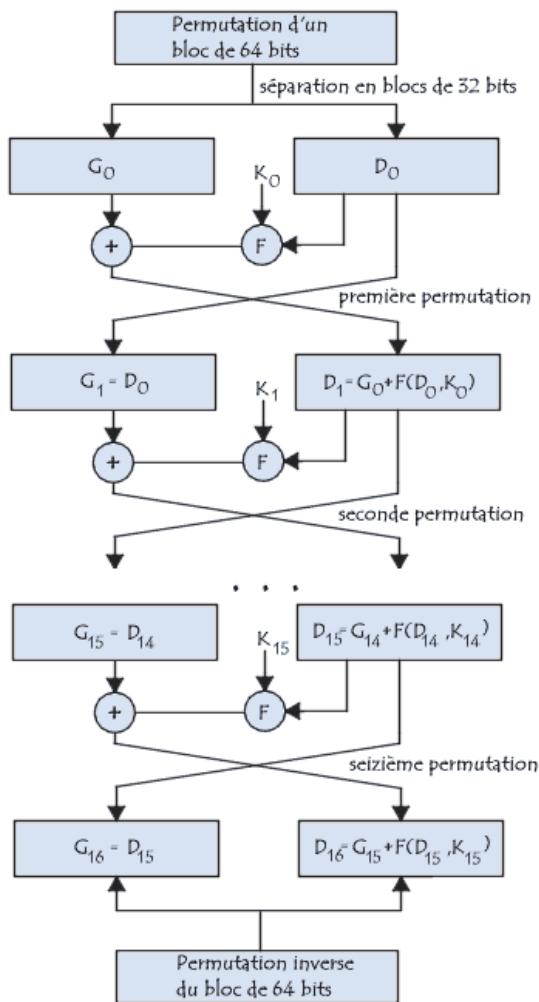
Il s'agit d'un système de chiffrement symétrique par blocs de 64 bits, dont 8 bits (un octet) servent de test de parité (pour vérifier l'intégrité de la clé). Chaque bit de parité de la clé (1 tous les 8 bits) sert à tester un des octets de la clé par parité impaire, c'est-à-dire que chacun des bits de parité est ajusté de façon à avoir un nombre impair de '1' dans l'octet à qui il appartient. La clé possède donc une longueur « utile » de 56 bits, ce qui signifie que seuls 56 bits servent réellement dans l'algorithme.

L'algorithme consiste à effectuer des combinaisons, des substitutions et des permutations entre le texte à chiffrer et la clé, en faisant en sorte que les opérations puissent se faire dans les deux sens (pour le déchiffrement). La combinaison entre substitutions et permutations est appelée **code produit**.

La clé est codée sur 64 bits et formée de 16 blocs de 4 bits, généralement notés  $k$ , à  $k_{16}$ . Étant donné que « seuls » 56 bits servent effectivement à chiffrer, il peut exister  $2^{56}$  (soit  $7.2 \cdot 10^{16}$ ) clés différentes !

Les grandes lignes de l'algorithme sont les suivantes :

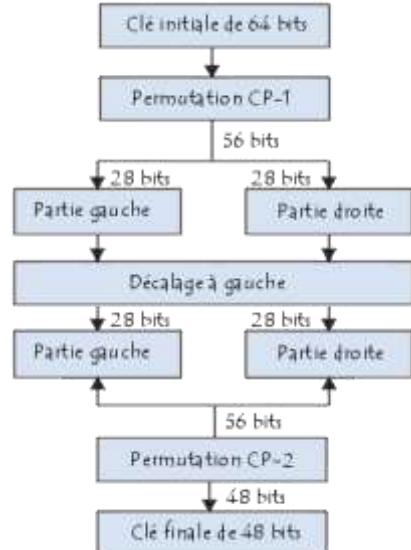
- Fractionnement du texte en blocs de 64 bits (8 octets) ;
- Permutation initiale des blocs ;
- Découpage des blocs en deux parties: gauche et droite, nommées  $G$  et  $D$  ;
- Etapes de permutation et de substitution répétées 16 fois (appelées **rondes**) ;
- Recollement des parties gauche et droite puis permutation initiale inverse.



## Génération des clés

Etant donné que l'algorithme du DES présenté ci-dessus est public, toute la sécurité repose sur la complexité des clés de chiffrement.

L'algorithme ci-dessous montre comment obtenir à partir d'une clé de 64 bits (composé de 64 caractères alphanumériques quelconques) 8 clés diversifiées de 48 bits chacune servant dans l'algorithme du DES :



Méthodes simples de chiffrement et des clés très longues .

Le DES

- Produit de transpositions et substitutions nombreuses et compliquées pour une clé relativement courte

=> facilité de transport.

- Les chiffres à substitution et à transposition sont faciles à réaliser en matériel.

**Les boîtes de transposition "P-Box"**

**Les boîtes de substitution "S-Box"**

**Deux modes**

- Mode cryptage par bloc de 64 bits
- Mode cryptage à la volée ("stream") (octets par octets avec des registres à décalage)

### **Utilisation d'une clé sur 56 bits**

En fait 8 fois 7 bits avec une parité (initialement 128 bits)

### **19 étages de logique combinatoire**

Appliquent des transpositions, substitutions sur des blocs de 2 x 32 bits

- 1 étage amont, 2 en aval sont des transpositions simples fixes
- 16 étages intermédiaires dépendent de la clé de façon complexe.

Le DES comporte plusieurs avantages qui en ont fait l'algorithme de chiffrement symétrique standard pendant longtemps, jusqu'il y a quelques années. En voici quelques-uns :

- il possède un haut niveau de sécurité,
- il est complètement spécifié et facile à comprendre,
- la sécurité est indépendante de l'algorithme lui-même,
- il est rendu disponible à tous, par le fait qu'il est public,
- il est adaptable à diverses applications (logicielles et matérielles),
- il est rapide et exportable,
- il repose sur une clé relativement petite, qui sert à la fois au chiffrement et au déchiffrement,
- il est facile à implémenter.

Le D.E.S. est un cryptosystème agissant par blocs. Cela signifie que D.E.S. ne chiffre pas les données à la volée quand les caractères arrivent, mais il découpe virtuellement le texte clair en blocs de 64 bits qu'il code séparément, puis qu'il concatène. Un bloc de 64 bits du texte clair entre par un côté de l'algorithme et un bloc de 64 bits de texte chiffré sort de l'autre côté. L'algorithme est assez simple puisqu'il ne combine en fait que des permutations et des substitutions.

C'est un algorithme de chiffrement à clef secrète. La clef sert donc à la fois à chiffrer et à déchiffrer le message. Cette clef a ici une longueur de 64 bits, c'est-à-dire 8 caractères, mais dont seulement 56 bits sont utilisés. On peut donc éventuellement imaginer un programme testant l'intégrité de la clef en exploitant ces bits inutilisés comme bits de contrôle de parité.

L'entièr sécurité de l'algorithme repose sur les clefs puisque l'algorithme est parfaitement connu de tous. La clef de 64 bits est utilisée pour générer 16 autres clefs de 48 bits chacune qu'on utilisera lors de chacune des 16 itérations du D.E.S.. Ces clefs sont les mêmes quel que soit le bloc qu'on code dans un message. Cet algorithme est relativement facile à réaliser matériellement et certaines puces chiffrent jusqu'à 1 Go de données par seconde. Pour les industriels, c'est un point important notamment face à des algorithmes asymétriques, plus lents, tels que l'algorithme R.S.A.

Plusieurs attaques existent pour casser le DES :

- La recherche exhaustive par force brute : on teste toutes les clés possibles, l'une après l'autre, afin de déchiffrer un bloc de données. On a besoin, en moyenne, de 2<sup>55</sup> essais.
- Une machine dédiée au calcul des clés : Deep Crack (Des Cracker, élaboré par l'Electronic Frontier Foundation) a coûté moins de 250'000 dollars. Elle disposait de 1850 processeurs en parallèle pour un temps de recherche de 56 heures (1998).
- Le calcul distribué : un regroupement d'ordinateurs par Internet qui partagent leur puissance de calcul. En 1998, Distributed.net a pu décrypter un message en 39 jours<sup>1</sup>. Le 19 janvier 1999, EFF et Distributed.net ont cassé en travaillant conjointement une clé en 22 heures et 15 minutes.
- Depuis 2006, un autre ordinateur dédié, nommé COPACABANA<sup>2</sup>, permet de casser une clé DES en 9 jours. En 2008, des améliorations logicielles ont même permis de diminuer le temps de recherche à 6,4 jours. Son avantage est son coût : 10.000\$. On peut donc facilement en acquérir plusieurs et donc diminuer le temps de recherche en conséquence. Par exemple, pour 4 machines acquises (=40,000\$), il faudra compter 1,6 jour de recherche.
- Cryptanalyse différentielle : il s'agit de la possibilité de produire au moyen de textes clairs choisis les textes chiffrés correspondants avec une clé inconnue. On analyse les différences résultantes sur les textes chiffrés et on donne des probabilités aux clés testées. En affinant, on trouve la clé la plus probable. La meilleure attaque différentielle connue demande 2<sup>47</sup> textes clairs choisis
- Cryptanalyse linéaire : on dispose d'un dictionnaire de (M,C). On ne possède pas la « boite noire » comme dans le cas différentiel. L'objectif est de déterminer une équation linéaire modélisant au mieux les transformations du cryptogramme créé par l'algorithme de chiffrement.
- Il existe d'autres attaques spécifiques au DES (Davie's attack, ou des attaques sur des versions simplifiées du DES) mais qui n'entrent pas dans le cadre de ce cours.

### **AES**

AES est un algorithme de chiffrement symétrique, choisi en octobre 2000 par le NIST pour être le nouveau standard de chiffrement pour les organisations du gouvernement des Etats-Unis. Il est issu d'un appel à candidatures international lancé en janvier 1997 et ayant reçu 15 propositions. Au bout

de cette évaluation, ce fut le candidat Rijndael (prononcer "Rayndal"), du nom de ses deux concepteurs Joan Daemen et Vincent Rijmen (tous les deux de nationalité belge) qui a été choisi.

### Le choix : Rijndael

A la suite de nombreux tests, c'est finalement Rijndael qui a remporté la médaille, et est ainsi devenu le remplaçant officiel du DES.

Il possède les propriétés suivantes :

- Plusieurs longueurs de clé et de bloc sont possibles : 128, 192, ou 256 bits ;
- Le nombre de cycles ("rondes") varie en fonction de la longueur des blocs et des clés (de 10 à 14) ;
- La structure générale ne comprend qu'une série de transformations/permutions/sélections ;
- Il est beaucoup plus performant que le DES ;
- Il est facilement adaptable à des processeurs de 8 ou de 64 bits ;
- Le parallélisme peut être implémenté

À chaque ronde, quatre transformations sont appliquées :

1. substitution d'octets dans le tableau d'état
2. décalage de rangées dans le tableau d'état
3. déplacement de colonnes dans le tableau d'état (sauf à la dernière ronde)
4. addition d'une "clé de ronde" qui varie à chaque ronde

### Principe de fonctionnement

-L'algorithme prend en entrée un bloc de 128 bits (la clé fait 128, 192 ou 256 bits. Les 128 bits en entrée sont « mélangés » selon une table définie au préalable.

-Ces octets sont ensuite placés dans une matrice de 4x4 éléments et ses lignes subissent une rotation vers la droite.

-L'incrément pour la rotation varie selon le numéro de la ligne.

-Une transformation est ensuite appliquée sur la matrice par un XOR avec une matrice clé.

-Finalement, un XOR entre la matrice et une autre matrice permet d'obtenir une matrice intermédiaire.

-Ces différentes opérations sont répétées plusieurs fois et définissent un « tour ».

-Pour une clé de 128, 192 ou 256, AES nécessite respectivement 10, 12 ou 14 tours.

L'algorithme AES n'est pas cassé à la date d'aujourd'hui.

Historiquement, le développement de l'AES a été instigué par le NIST (National Institute of Standards and Technology) le 2 janvier 1997. L'algorithme a été choisi il y a peu de temps : il s'agit de l'algorithme [Rijndael](#)(prononcer "Raindal"). Cet algorithme suit les spécifications suivantes :

- l'AES est un standard, donc libre d'utilisation, sans restriction d'usage ni brevet.
- c'est un algorithme de type symétrique (comme le DES)
- c'est un algorithme de chiffrement par blocs (comme le DES)

Le choix de cet algorithme répond à de nombreux critères plus généraux dont nous pouvons citer les suivants :

- sécurité ou l'effort requis pour une éventuelle cryptanalyse.
- facilité de calcul : cela entraîne une grande rapidité de traitement
- besoins en ressources et mémoire très faibles
- flexibilité d'implémentation: cela inclut une grande variété de plateformes et d'applications ainsi que des tailles de clés et de blocs supplémentaires (c.f. ci-dessus).
- hardware et software : il est possible d'implémenter l'AES aussi bien sous forme logicielle que matérielle (câblé)
- simplicité : le design de l'AES est relativement simple

Si l'on se réfère à ces critères, on voit que l'AES est également un candidat particulièrement approprié pour les implantations embarquées qui suivent des règles beaucoup plus strictes en matière de ressources, puissance de calcul, taille mémoire, etc... C'est sans doute cela qui a poussé le monde de la 3G (3ème génération de mobiles) à adopter l'algorithme pour son schéma d'authentification "Millenage".

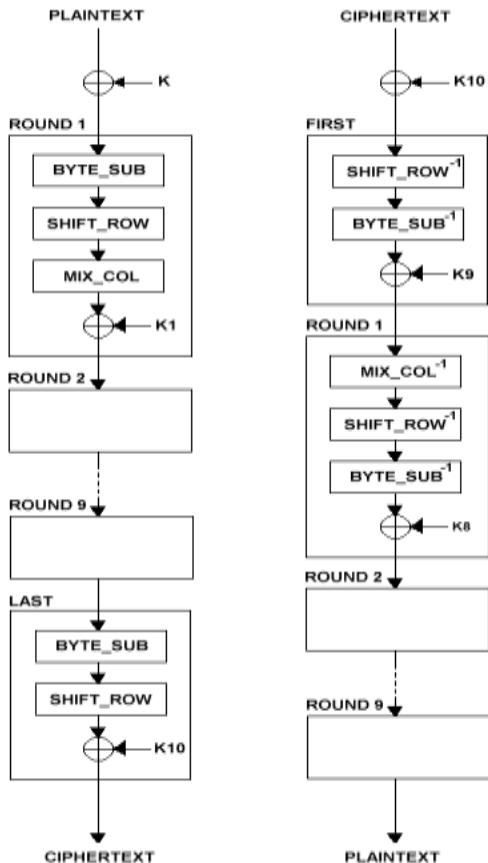
L'AES opère sur des blocs de 128 bits (plaintext P) qu'il transforme en blocs cryptés de 128 bits (C) par une séquence de N opérations ou "rounds", à partir d'une clé de 128, 192 ou 256 bits. Suivant la taille de celle-ci, le nombre de rounds diffère : respectivement 10, 12 et 14 rounds.

Le schéma suivant décrit succinctement le déroulement du chiffrement :

- BYTE\_SUB (Byte Substitution) est une fonction non-linéaire opérant indépendamment sur chaque bloc à partir d'un tableau dit de substitution.

- SHIFT\_ROW est une fonction opérant des décalages (typiquement elle prend l'entrée en 4 morceaux de 4 octets et opère des décalages vers la gauche de 0, 1, 2 et 3 octets pour les morceaux 1, 2, 3 et 4 respectivement).
- MIX\_COL est une fonction qui transforme chaque octet d'entrée en une combinaison linéaire d'octets d'entrée et qui peut être exprimée mathématiquement par un produit matriciel sur le corps de Galois ( $2^8$ ).
- le + entouré d'un cercle désigne l'opération de OU exclusif (XOR). (ADD ROUNDKEY) = LAST
- $K_i$  est la  $i$ ème sous-clé calculée par un algorithme à partir de la clé principale  $K$ . = LAST

Le déchiffrement consiste à appliquer les opérations inverses, dans l'ordre inverse et avec des sous-clés également dans l'ordre inverse.



## Fonctionnement

- L'algorithme prend en entrée un bloc de 128 bits (16 octets), la clé fait 128, 192 ou 256 bits. Les 16 octets en entrée sont permutés selon une table définie au préalable. Ces octets sont ensuite placés dans une matrice de 4x4 éléments et ses lignes subissent une rotation vers la droite. L'incrément pour la rotation varie selon le numéro de la ligne. Une transformation linéaire est ensuite appliquée sur la matrice, elle consiste en la multiplication binaire de chaque élément de la matrice avec des polynômes issus d'une matrice auxiliaire, cette multiplication est soumise à des règles spéciales selon GF(2<sup>8</sup>) (groupe de Galois ou corps fini). La transformation linéaire garantit une meilleure diffusion (propagation des bits dans la structure) sur plusieurs tours. Finalement, un XOR entre la matrice et une autre matrice permet d'obtenir une matrice intermédiaire. Ces différentes opérations sont répétées plusieurs fois et définissent un « tour ». Pour une clé de 128, 192 ou 256, AES nécessite respectivement 10, 12 ou 14 tours.

## Avantages et limites

Les principaux avantages sont :

- des performances très élevées,
- la possibilité de réalisation en "Smart Card" avec peu de code,
- la possibilité de parallélisme,
- il ne comprend pas d'opérations arithmétiques : ce sont uniquement des décalages et des XOR,
- il n'utilise pas de composants d'autres cryptosystèmes,
- il n'est pas fondé sur des relations obscures entre opérations,
- le nombre de rondes peut facilement être augmenté si c'est requis,
- il ne possède pas de clés faibles,
- il est résistant à la cryptanalyse différentielle et linéaire.

Il possède pourtant quelques inconvénients et limites :

- le code et les tables sont différents pour le chiffrement et déchiffrement,
- le déchiffrement est plus difficile à implanter en "Smart Card",
- dans une réalisation matérielle, il y a peu de réutilisation des circuits de chiffrement pour effectuer

le déchiffrement.

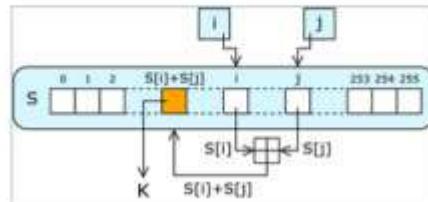
- **RC4** est un algorithme de chiffrement à flot conçu en 1987 par Ronald Rivest, l'un des inventeurs du RSA, pour les Laboratoires RSA. Il est supporté par différentes normes, par exemple dans SSL ou encore WEP.

#### ■ Principe

- RC4 fonctionne de la façon suivante : la clef RC4 permet d'initialiser un tableau de 256 octets en répétant la clef 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 est de mélanger autant que possible le tableau. Au final on obtient une suite de bits qui paraît tout à fait aléatoire. Par la suite on peut extraire des bits par conséquent pseudo-aléatoires.

#### ■ Crypto systèmes basés sur RC4

- WEP
- WPA



RC4 a été conçu par Ronald Rivest (Le 'R' de RSA) en 1987. Officiellement nommé Rivest Cipher 4, l'acronyme RC est aussi surnommé Ron's Code

Il est utilisé dans des protocoles comme WEP, WPA ainsi que TLS. Les raisons de son succès sont liées à sa grande simplicité et à sa vitesse de chiffrement. Les implementations matérielles ou logicielles étant faciles à mettre en œuvre.

La clé RC4 permet d'initialiser un tableau 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 est de mélanger autant que possible le tableau. Au final on obtient une suite de bits pseudo-aléatoires qui peuvent être utilisés pour chiffrer le message via un XOR (Comme dans le cas d'un masque jetable).

**RC4 fonctionne de la façon suivante : la clé RC4 permet d'initialiser un tableau 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 est de mélanger autant que possible le tableau. Finalement on obtient une suite de bits pseudo-aléatoires qui peuvent être utilisés pour chiffrer les données via un XOR.**

RC4 est un algorithme de chiffrement à flot conçu en 1987 par Rivest et dédié aux applications logicielles. Il est largement déployé notamment dans le protocole SSL/TLS et la norme WEP pour les réseaux sans fil, IEEE 802.11. RC4 présente certaines faiblesses dues à son initialisation, qui est relativement faible, et qui ne prévoit pas de valeur initiale pour la re-synchronisation. Employé par exemple avec le protocole de re-synchronisation choisi dans la norme IEEE 802.11, RC4 s'avère extrêmement faible. ◊ RC4 est un algorithme de chiffrement à flot destiné aux applications logicielles. Il a été conçu par R. Rivest en 1987 pour les laboratoires RSA [4]. Malgré un certain nombre de faiblesses, il est encore très utilisé aujourd'hui, notamment du fait de sa vitesse élevée (7 cycles par octet sur un Pentium III par exemple). Il est employé par exemple dans SSL/TLS, protocole permettant d'assurer la confidentialité des transactions Web, dans la norme de chiffrement WEP (Wired Equivalent Privacy) pour les réseaux sans fil, IEEE 802.11b...

#### Paramètres.

- Longueur de clé : variable, entre 40 et 1024 bits.
- Pas de valeur initiale.

L'état interne de RC4 est formé d'un tableau de  $2^n$  éléments de  $n$  bits. À chaque instant, ce tableau correspond à une permutation des mots de  $n$  bits. Généralement, on prendra  $n=8$ , qui correspond à un tableau de 256 octets.

**Description.** RC4 est composé d'une phase d'initialisation, qui consiste à engendrer une permutation des mots de  $n$  bits à partir de la clé secrète.

Puis, à chaque instant, on modifie le tableau initial en permutant deux de ses éléments, et on produit un mot de  $n$  bits de suite chiffrante, qui correspond à un élément du tableau.

**Sécurité.** La plupart des attaques connues sur RC4 exploitent des faiblesses de la phase d'initialisation. Ainsi, plusieurs attaques par distinguo ont été proposées, notamment par Mantin et Shamir [3]. De façon générale, les premiers mots de la suite générée sont particulièrement vulnérables, et il est généralement conseillé de ne pas rendre en compte les 512 premiers octets produits [2].

Il n'existe à ce jour aucune attaque par recouvrement de clef sur RC4 sauf dans des contextes particuliers. Par exemple, Fluhrer, Mantin et Shamir [1] ont montré qu'en l'absence de valeur initiale, le mécanisme de re-synchronisation utilisé dans la norme [WEP IEEE](#) 802.11 conduisait à l'utilisation de la même suite chiffrante pour chiffrer différents messages. Le niveau de sécurité offert par RC4 avec un protocole de re-synchronisation de ce type est donc extrêmement faible. Plus généralement, on considère que RC4 est particulièrement sensible aux attaques liées au protocole de réinitialisation.

**Propriété intellectuelle.** RC4 est un algorithme propriétaire de la société RSA Data Security, Inc.

Il s'agit d'un chiffrement par flux créé en 1987 par Ron Rivest<sup>1</sup>. Diverses analyses ont démontré que la période du générateur est supérieure à  $10^{100}$ .

Il s'agit probablement du chiffrement par flots le plus utilisé actuellement. 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). Ce dernier fut remplacé par le WPA (Wi-Fi Protected Access), mais celui-ci utilise toujours le RC4.

Initialement gardé secret par la RSA, cet algorithme a été publié anonymement en 1994 sur Internet. Il existe une version allégée du chiffrement RC4, portant le nom de ARC4 (Alleged RC4), utilisable légalement. Le RC4 reste la propriété de RSA Labs.

Un chiffrement par RC4 est très rapide, comme le montre le tableau 6.1

Algorithm	Longueur de la clé	Vitesse (en Mbps)
DES	56	9
3DES	168	3
RC4	Variable	45

TAB. 6.1 – Vitesses de quelques chiffrements symétriques.

### 6.3.1 Fonctionnement du RC4

Cet algorithme fonctionne sur les octets. Ainsi, la clé, de longueur variable, peut avoir une taille comprise entre 1 et 256 octets (de 8 à 2048 bits). Elle est utilisée pour initialiser un vecteur S de 256 octets. A tout moment, S contient une permutation de toutes les cellules le composant. La figure 6.8 illustre le principe du RC4.

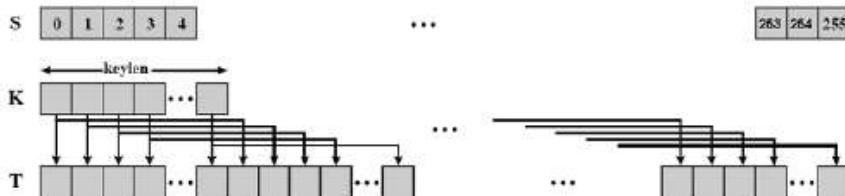


FIG. 6.8 – Fonctionnement du chiffrement RC4

Initialement, les cellules de S reçoivent une valeur égale à leur position (i.e.,  $S[0]=0$ ,  $S[1]=1$ , ...) Un vecteur temporaire de longueur T (de longueur égale à S) est également créé 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. 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. Ces deux portions de code portent le nom de KSA (Key Schedule Algorithm).

A partir de cet instant, 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  $S[0]$ . A nouveau, on peut illustrer algorithmiquement la méthode (on parle de PRGA pour Pseudo-Random Generation Algorithm).

## 6.4 Comparaisons des chiffrements par blocs et par flots

Le tableau 6.2 regroupe les avantages et inconvénients de chaque famille. Il est toutefois à remarquer qu'au niveau des applications, chaque type de chiffrement peut être utilisé dans toutes les applications.

	par Blocs	par Flots
<b>Avantages</b>	- Réutilisation des clés	- Rapidité - Moins de code d'implémentation
<b>Inconvénients</b>		- Deux utilisations d'une même clé facilite la cryptanalyse
<b>Applications</b>	- Transfert de fichiers	- Chiffrement de canal de communication

La première fonction de la cryptographie consiste donc à assurer la confidentialité d'un échange d'informations. Deux parties d'un échange confidentiel s'accordent d'abord sur une convention secrète pour rédiger leurs messages, et si elles l'ont soigneusement choisie, personne d'autre ne devrait pouvoir saisir leur échange.

Si le caractère secret de telles conventions est envisageable entre quelques personnes isolées pour une période limitée, il est inconcevable à grande échelle et pour une durée assez longue.

C'est ce qu'avait compris Auguste Kerchoffs lorsqu'il établit les principes de base de la cryptographie pratique dont un principe fondamental exige un système de chiffrement: "qui n'exige pas le secret, et qui puisse sans inconvénient tomber entre les mains de l'ennemi".

Un autre principe précise que: "la clé doit pouvoir être changée ou modifiée au gré des correspondants".

Le premier de ces deux principes, connu aujourd'hui sous le nom de "[principe de Kerckhoffs](#)", stipule donc que la sécurité d'un système de chiffrement n'est pas fondée sur le secret de la procédure qu'il suit, mais uniquement sur un paramètre utilisé lors de sa mise en oeuvre: la clé. Cette clé est le seul secret de la convention d'échange.

Ce principe a cependant été reformulé par Claude Shannon : "l'adversaire connaît le système".

Cette formulation est connue sous le nom de la "[maxime de Shannon](#)". C'est le principe le plus souvent adopté par les cryptologues, par opposition à la sécurité par l'obscurité.

1. Le système doit être matériellement, sinon mathématiquement indéchiffrable ;
- 2. Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;**
3. La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;
4. Il faut qu'il soit applicable à la correspondance télégraphique ;
5. Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes ;
6. Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

Il faut distinguer les termes "Secret" et "Robustesse" d'un algorithme. Le secret de l'algorithme revient à cacher les concepts de celui-ci, ainsi que les méthodes utilisées (fonctions mathématiques). La robustesse quant à elle désigne la résistance de l'algorithme à diverses attaques qui seront explicitées dans la suite de ces notes.

Le principe de la **sécurité par l'obscurité** (de l'anglais : « *security through/by obscurity* ») repose sur la non-divulgation d'information relative à la structure, au fonctionnement et à l'implémentation de l'objet ou du procédé considéré, pour en assurer la sécurité. Cela s'applique aux domaines sensibles de l'informatique, de la [cryptologie](#), de l'armement, etc.

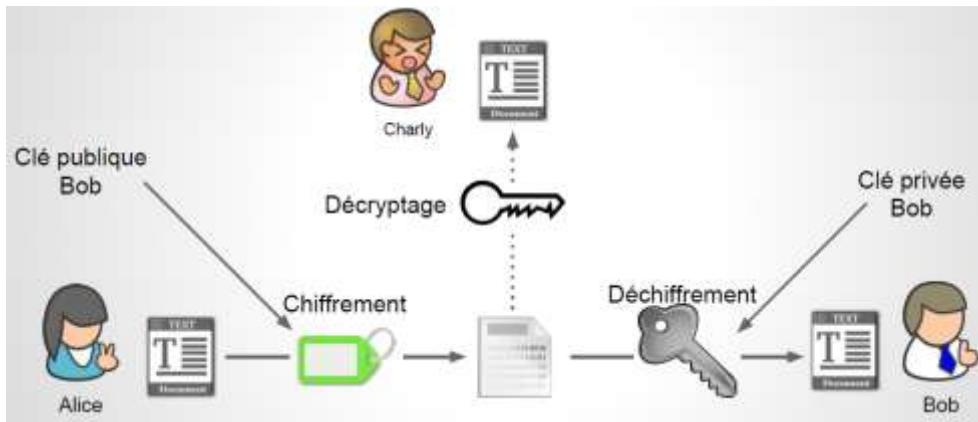
En cryptologie, la sécurité par l'obscurité va à l'encontre du [principe de Kerckhoffs](#) qui veut que la sécurité de tout [cryptosystème](#) ne repose que sur le secret de la clé.

Un message chiffré par exemple avec [ROT13](#) ne peut rester secret que si la méthode utilisée pour chiffrer reste inconnue de l'adversaire. Si l'attaquant « connaît le système » alors il sera capable de déchiffrer le message. Les cryptologues prônent la transparence en matière de processus et de conception de  [primitives cryptographiques](#). Une telle pratique permet de mieux évaluer la sécurité des algorithmes.

Dans tous les cas, les algorithmes sont connus et reposent sur l'utilisation d'un secret plus ou moins long pour assurer la sécurité (principe de **Kerckhoffs**).

Dans le cas où l'on garde l'algorithme secret, on parle de "**sécurité par l'obscurité**". En plus de n'apporter que peu (voir pas) de réel sécurité en pratique, cela empêche l'audit des méthodes de sécurité et pose donc un doute sur leur efficacité.

Violation du principe de Kerckhoffs qui veut qu'un système soit sécurisé par sa conception, et non parce que sa conception est inconnue par son adversaire. Théoriquement, un algorithme de chiffrement n'a pas besoin d'être sécurisé, et inconnu de tous, il suffit que la clé soit « cachée » pour que celle-ci soit considérée sûre et indéchiffrable, même si le processus de chiffrement et de déchiffrement est connu. La sécurité par l'obscurité prend cette théorie à contre-pied en empêchant la divulgation de la totalité de l'opération, qu'il s'agisse de la clé, de l'algorithme utilisé, et ainsi de suite...



#### Algorithmes de cryptographie asymétrique (à clé publique et privée)

- Pour résoudre en partie le problème de la gestion des clés, la cryptographie asymétrique a été mise au point dans les années 1970. Elle se base sur le principe de deux clés :
  - une publique, permettant le chiffrement ;
  - une privée, permettant le déchiffrement.
- Comme son nom l'indique, la clé publique est mise à la disposition de quiconque désire chiffrer un message. Ce dernier ne pourra être déchiffré qu'avec la clé privée, qui doit être confidentielle.
- Quelques algorithmes de cryptographie asymétrique très utilisés :
  - RSA ;
  - DSA ;
  - Protocole d'échange de clés Diffie-Hellman ;

La **cryptographie asymétrique**, ou *cryptographie à clé publique* est fondée sur l'existence de fonctions à sens unique, c'est-à-dire qu'il est simple d'appliquer cette fonction à un message, mais extrêmement difficile de retrouver ce message à partir du moment où on l'a transformé.

- $F$  est à sens unique  $\Leftrightarrow$  Quelque soit  $x$ ,  $y = f(x)$  est calculable rapidement.
- $X = f^{-1}(y)$  se calcule en un temps très long

En réalité, on utilise en cryptographie asymétrique des fonctions à sens unique et à brèche secrète. Une telle fonction est difficile à inverser, à moins de posséder une information particulière, tenue secrète, nommée *clé privée*.

- Une fonction est dite trappe ou à brèche secrète si elle est à sens unique sauf pour toute personne connaissant un secret ou une brèche, permettant de calculer un algorithme d'inversion rapide.
- On appelle **exponentiation modulaire** de la variable  $a$  la fonction  $a^p \bmod n$  ( $p$  fixe). Si l'existence d'un algorithme permettant de calculer des racines  $p$ -èmes modulo  $n$  est démontré, on ne connaît néanmoins pas cet algorithme. Par contre, si on connaît la factorisation de  $n$  (la brèche), on peut très facilement inverser l'exponentiation modulaire.

Une clé est utilisée pour coder le message et une autre pour décoder le message crypté. Dans un système à clé publique, chaque personne dispose de deux clés: une publique et une privée. Les messages chiffrés avec l'une des clés peuvent seulement être déchiffrés par l'autre clé de la paire.

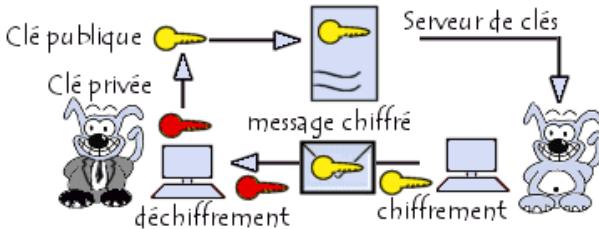
Le principe de **chiffrement asymétrique** (appelé aussi **chiffrement à clés publiques**) est apparu en 1976, avec la publication d'un ouvrage sur la [cryptographie](#) par *Whitfield Diffie et Martin Hellman*.

Dans un cryptosystème asymétrique (ou *cryptosystème à clés publiques*), les clés existent par paires (le terme de *bi-clés* est généralement employé) :

- Une clé publique pour le chiffrement ;
- Une clé secrète pour le déchiffrement.

Ainsi, dans un système de chiffrement à clé publique, les utilisateurs choisissent une clé aléatoire qu'ils sont seuls à connaître (il s'agit de la [clé privée](#)). A partir de cette clé, ils déduisent chacun automatiquement un algorithme (il s'agit de la clé publique). Les utilisateurs s'échangent cette clé publique au travers d'un canal non sécurisé.

Lorsqu'un utilisateur désire envoyer un message à un autre utilisateur, il lui suffit de chiffrer le message à envoyer au moyen de la clé publique du destinataire (qu'il trouvera par exemple dans un serveur de clés tel qu'un [annuaire LDAP](#)). Ce dernier sera en mesure de déchiffrer le message à l'aide de sa clé privée (qu'il est seul à connaître).



Ce système est basé sur une fonction facile à calculer dans un sens (appelée *fonction à trappe à sens unique* ou en anglais *one-way trapdoor function*) et mathématiquement très difficile à inverser sans la clé privée (appelée *trappe*).

A titre d'image, il s'agit pour un utilisateur de créer aléatoirement une petite clé en métal (la clé privée), puis de fabriquer un grand nombre de cadenas (clé publique) qu'il dispose dans un casier accessible à tous (le casier joue le rôle de canal non sécurisé). Pour lui faire parvenir un document, chaque utilisateur peut prendre un cadenas (ouvert), fermer une valisette contenant le document grâce à ce cadenas, puis envoyer la valisette au propriétaire de la clé publique (le propriétaire du cadenas). Seul le propriétaire sera alors en mesure d'ouvrir la valisette avec sa clé privée.

#### Avantages et inconvénients

Le problème consistant à se communiquer la clé de déchiffrement n'existe plus, dans la mesure où les clés publiques peuvent être envoyées librement. Le chiffrement par clés publiques permet donc à des personnes d'échanger des messages chiffrés sans pour autant posséder de secret en commun.

En contrepartie, tout le challenge consiste à (s')assurer que la clé publique que l'on récupère est bien celle de la personne à qui l'on souhaite faire parvenir l'information chiffrée !

Ces algorithmes sont aussi synonymes d'algorithmes à clés publiques. Une clé différente est utilisée à la fois pour chiffrer et déchiffrer, et il est impossible de générer une clé à partir de l'autre. Il a été inventé en 1975 par deux ingénieurs en électronique : Whitfield Diffie et Martin Hellman de l'Université de Stanford.

- **Théorie :**

Une des difficultés principales de la méthode ci-dessus est que chaque couple potentiel d'utilisateurs doit posséder sa propre clé secrète, et se l'échanger par un moyen sécurisé avant leur premier échange d'informations, ce qui peut s'avérer difficile à réaliser dans la pratique. Le but d'un système à clé publique est de résoudre ce problème. La clé publique est généralement publiée dans un répertoire. L'avantage est donc qu'Alice peut envoyer un message à Bob sans communication privée préalable (elle choisit sa clé privée, et la clé publique de Bob). Bob est la seule personne à pouvoir déchiffrer le message en appliquant sa clé secrète et personnelle, et la clé publique d'Alice. On dit généralement que chaque clé déverrouille le code produit par l'autre. Une remarque intéressante à faire est qu'avec ce système, même Alice qui a chiffré un message pour Bob, ne pourra déchiffrer le message ainsi codé. C'est un des systèmes les plus évolués que l'on peut actuellement trouver.

La première application à ce principe fut le chiffrement RSA. Depuis, plusieurs systèmes ont été proposés. Leur sécurité repose sur divers problèmes calculatoires, et notamment la théorie des grands nombres.

En "opposition" à la cryptographie à clé secrète, on parle de cryptographie à **clé publique**:

- Chiffrement asymétrique
- On utilise deux clés: une clé publique et une clé privée
- Les deux parties d'un échange ne peuvent inverser leurs rôles
- Basé sur des problèmes mathématiques complexes

**Avantages:**

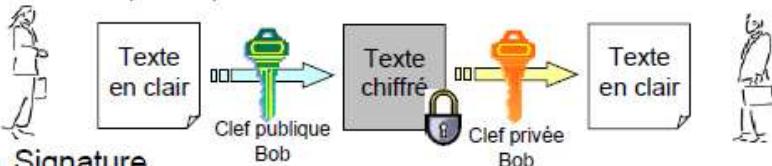
- Distinction des rôles
- Diffusion "aisée" des clés

**Inconvénients:**

- "Lent" et coûteux en ressources
- Problème d'authenticité des clés
- Pas de *confidentialité persistante*

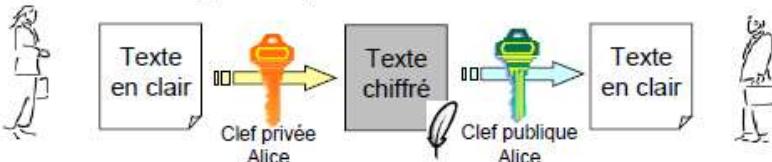
## ■ Chiffrement

- ◆ Clef publique utilisée pour le chiffrement, seul le détenteur de la clef privée peut déchiffrer



## ■ Signature

- ◆ Clef privée utilisée pour le chiffrement, seul son détenteur peut chiffrer, mais tout le monde peut déchiffrer (et donc en fait vérifier la "signature")



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

Le concept de cryptographie à clef publique a été inventé par Whitfield Diffie et Martin Hellman en 1976, dans le but de résoudre le problème de distribution des clefs posé par la cryptographie à clef secrète. De nombreux algorithmes permettant de réaliser un cryptosystème à clef publique ont été proposés depuis. Ils sont le plus souvent basés sur des problèmes mathématiques difficiles à résoudre, donc leur sécurité est conditionnée par ces problèmes, sur lesquels on a maintenant une vaste expertise. Mais, si quelqu'un trouve un jour le moyen de simplifier la résolution d'un de ces problèmes, l'algorithme correspondant s'écroulera.

Tous les algorithmes actuels présentent l'inconvénient d'être bien plus lents que les algorithmes à clef secrète ; de ce fait, ils sont souvent utilisés non pour chiffrer directement des données, mais pour chiffrer une clef de session secrète. Certains algorithmes asymétriques ne sont adaptés qu'au chiffrement, tandis que d'autres ne permettent que la signature. Seuls trois algorithmes sont utilisables à la fois pour le chiffrement et pour la signature : RSA, ElGamal et Rabin.

Il existe trois grands usages de la cryptographie asymétrique:

- Le chiffrement asymétrique Comme pour le chiffrement symétrique, cela permet de garantir la confidentialité d'un message.
- Les signatures numériques Basé sur les mêmes algorithmes que le chiffrement symétrique, les signatures numériques permettent de garantir l'authenticité d'un message.
- L'échange de clés En plus du chiffrement symétrique, il est possible d'utiliser les systèmes à clé publique pour procéder à des échanges de clés secrètes.

Il existe trois grands usages de la cryptographie asymétrique:

- Le chiffrement asymétrique
- Les signatures numériques
- L'échange de clés Certains algorithmes ont plusieurs usages, d'autres sont limités.

Exemple:

- ElGamal (chiffrement)
- RSA (chiffrement, signature)
- DSS/DSA (signature, pouvant utiliser ElGamal...)
- Diffie-Hellman (échange de clé)

Les clés utilisées en cryptographie asymétrique ont les caractéristiques suivantes:

- Il est très difficile de trouver la clé privée à partir de la clé publique
- Les opérations (chiffrement/déchiffrement) sont peu coûteuses en connaissance des clés, mais très complexes dans le cas contraire
- Les deux clés (privée/publique) ne sont pas interchangeables lors d'un même échange Sur le dernier point:
- Certains algorithmes permettent d'utiliser indifféremment la clé privée pour chiffrer/déchiffrer (RSA par exemple)

- ... mais pas sur le même échange!

Propriétés de la clé publique: ● Elle n'est pas secrète: elle peut (et doit) être diffusée pour être utile

- Elle est lié à une identité (réelle ou non)
- Elle doit donc être authentique

La diffusion des clés publiques est un élément clé des Infrastructures à Clés Publiques (PKI - Public Key Infrastructure). Quelques exemples de méthodes pour cela:

- Annuaire d'entreprise
- Serveur de clés publiques
- Réseau de confiance

Propriétés de la clé privée: ● Elle représente le secret associé à une clé publique

- Sa génération doit être fiable (générateur aléatoire cryptographiquement sûr par exemple)
- Elle doit rester secrète, et donc circuler le moins possible (voir jamais)

Idéalement, la génération d'une paire de clé doit se faire sur le poste de l'utilisateur (/entité) où elle sera utilisé, et ne jamais être communiquée.

On peut voir les clés utilisées de la façon suivante:

- La clé publique représente une identité pour les autres
- La clé privée permet de prouver aux autres son identité

En effet, dans le cas du chiffrement: on chiffre avec la clé publique, à destination d'une identité.

Ensuite, on déchiffre avec la clé privée, ce qui prouve que l'on dispose bien de cette identité. Pour les signatures numériques, la comparaison est directe: la signature est réalisée avec une clé privée, prouvant l'identité du signataire; cette information d'identité pouvant alors être vérifiée avec la clé publique.

**ElGamal** est un algorithme de chiffrement asymétrique. Il fonctionne en trois phases:

- Génération de clé
- Chiffrement
- Déchiffrement

**RSA** fonctionne sur les mêmes étapes que ElGamal:

- Génération de clé
  - Chiffrement
  - Déchiffrement
- /// Une des différences dans son utilisation est que RSA permet d'effectuer le chiffrement indifféremment avec la clé privée ou publique (et le déchiffrement avec la clé correspondante associée).

#### Faiblesses de sécurité

- RSA comme ElGamal ont été plusieurs fois affaibli, principalement par l'évolution des performances de calcul. Pour pallier à cela, on augmente les tailles de clés (chaque bit de clé en plus augmente considérablement la difficulté).
- Comme tout système à clé publique, l'authenticité des clés est un maillon faible
- Les opérations mathématiques non équilibrées exposent des vulnérabilités particulières (canaux cachés)
- Attaque sur RSA: utilisation courantes des mêmes valeurs pour p et q, permettant d'exploiter plusieurs clés publiques pour retrouver une clé privée!
- ElGamal est probabiliste: cela signifie que deux chiffrés d'un même message seront différents.
- RSA est déterministe, mais certains usages (signature électronique) introduisent un élément aléatoire.
- Test d'indistinguabilité: on chiffre deux messages. On prend au hasard un message et on donne le chiffré à l'attaquant. Ce dernier ne doit pas pouvoir déterminer quel message est le message d'origine avec plus de 50% de réussite.

Champs d'applications de la cryptographie asymétrique:

- Échange de clé secrètes: une clé secrète représente un volume de donnée suffisamment faible pour fonctionner ici.
- Chiffrement: même si il n'est pas pratique à grand volume, il reste possible
- Signature numérique (voir le cours correspondant, ainsi que les calculs d'empreintes)
- Gestion de clés: utilisation d'une PKI

La cryptographie à clé publique fait intervenir deux clés (privée/publique)

- Elle est plus coûteuse que la cryptographie à clé secrète
- Les deux peuvent fonctionner ensemble pour garantir la sécurité d'un système
- Deux exemples: RSA, ElGamal
- La cryptographie à clé publique permet de réaliser la confidentialité et l'authenticité à travers le chiffrement et la signature numérique

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.

Le concept date officiellement de 1976 de Diffie et Hellman. Officieusement, les bases existent depuis 1969 par Ellis. La première implémentation a lieu en 1978 par Rivest, Shamir et Adleman sous la forme de l'algorithme RSA bien que, là aussi, les fondements de ce système datent de 1973, par Cocks.

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)

## Cryptographie asymétrique

### Le protocole de Diffie-Hellman

La distribution d'une clé secrète est le problème crucial de la cryptographie

Ainsi un chiffrement symétrique sûr est fragilisé dans la pratique par le problème de la distribution des clés. La multiplication des contacts nécessaires et la prolifération des réseaux rendent cette problématique de plus en plus coûteuse et hasardeuse

La solution est apparue en 1976 lorsque W Diffie et M Hellman proposèrent une solution au problème de l'échange des clés secrètes

- Cette méthode utilise une fonction à sens unique pour tout le monde excepté pour son créateur qui peut l'inverser grâce à la connaissance d'une information particulière
- Cette fonction se base sur l'arithmétique modulaire. L'idée de base consiste à calculer des valeurs du type  $x^a \text{ modulo } p$ . L'opération inverse est très difficile.
- Même si l'on connaît les nombres  $x$ ,  $p$  et  $x^a \text{ modulo } p$  il est impossible en pratique de retrouver le nombre  $a$

La sécurité de ce protocole est calculatoire

Elle se fonde sur l'hypothèse qu'avec une puissance de calcul et un temps limité, un adversaire ne peut inverser la fonction exponentielle modulaire et donc retrouver le secret  $a$  à partir des éléments échangés.

$a$  est la clé secrète  $x^a \text{ modulo } p$  est une information publique

## L'algorithme RSA

RSA est le premier système de chiffrement à clé publique

Il a été concu en 1977 par Rivest, Shamir et Aldeman du MIT.  
Rapidement devenu un standard international, la technique RSA a été commercialisée par plus de 400 entreprises et l'on estime que plus de 400 millions de logiciels l'utilisent

Le système RSA est fondé sur la difficulté de factoriser des grands nombres et la fonction à sens unique utilisée est une fonction puissance

- Soit  $p$  et  $q$  deux grands nombres premiers. Il est très difficile de retrouver ces 2 nombres en connaissant leur produit  $n = p \cdot q$
- Pour le calcul des clés publique et privée, il faut choisir deux grands nombres premiers  $p$  et  $q$ .

On calcul le produit  $n = pq$ . On choisi un grand nombre  $e$  premier avec  $(p-1)(q-1)$ .

- On calcul ensuite un nombre  $d$  tel que  $ed \equiv 1 \pmod{(p-1)(q-1)}$
- Le couple  $(n, e)$  est la clé publique,  $d$  est clé privée

Le message chiffré s'obtient en calculant

$y = x^e \text{ modulo } n$

Pour déchiffrer le message il suffit de calculer

$z = y^d \text{ modulo } n$  qui vaut  $x$  puisque  $y^d = x^{ed} = x \text{ modulo } n$

R.S.A. :

Cet algorithme a été inventé par Rivest R., Shamir A., et Adleman L. du M.I.T. (Massachusetts Institute of Technology). C'est l'algorithme à clé publique le plus commode qui existe. Comme pour le D.E.S. sa sécurité repose sur l'utilisation de clés suffisamment longue (512 bits n'est pas assez, 768 est modérément sûr, et 1024 bits est une bonne clé). C'est la difficulté que l'on a à factoriser les entiers premiers (le problème des logarithmes discrets est souvent considérés comme insurmontable) qui font que l'on ne peut que difficilement casser cet algorithme. Cependant de larges avancées en matière de factorisation des entiers larges, ou une augmentation considérable de la puissance de nos super-calculateurs rendront RSA très vulnérable.

RSA est aujourd’hui utilisé dans une large variété de produits (téléphones, réseaux Ethernet, etc...) , de logiciels de différentes marques (Microsoft, Apple, Novell, Sun), dans des industries et enfin dans les télécommunications.

RSA, du nom de ces inventeurs, est un algorithme de chiffrement appartenant à la grande famille "Cryptographie asymétrique".

RSA peut être utilisé pour assurer :

- la confidentialité : seul le propriétaire de la clé privée pourra lire le message chiffré avec la clé publique correspondante.
- la non-altération et la non-répudiation : seul le propriétaire de la clé privée peut signer un message (avec la clé privée). Une signature déchiffrée avec la clé publique prouvera donc l'authenticité du message.

### Résumé

1. Génération de 2 nombres premiers p et q
2. Calcul de  $n = p * q$
3. Déterminer e tel que  $3 < e < \varphi(n)$  et  $(e, \varphi(n)) = 1$
4. Calculer d tel que  $e * d \equiv 1 \pmod{\varphi(n)}$
5. Clé publique :  $(e, n)$
6. Clé privée :  $(d, n)$
7. p et q doivent rester secrets, voire supprimés
8.  $C = M^e \pmod{n}$  et  $M = C^d \pmod{n}$

L'algorithme le plus célèbre d'algorithme à clef publique a été inventé en 1977 par Ron Rivest, Adi Shamir et Len Adleman, à la suite de la publication de l'idée d'une cryptographie à clef publique par Diffie et Hellman. Il fut appelé RSA, des initiales de ces inventeurs.

RSA est basé sur la difficulté de factoriser un grand nombre en produit de deux grands facteurs premiers. L'algorithme fonctionne de la manière suivante :

Imaginons que Bob souhaite recevoir d'Alice des messages en utilisant RSA.

#### 1. génération des clefs :

- a. p et q, deux grands nombres premiers sont générés au hasard grâce à un algorithme de test de primalité probabiliste, avec  $n = pq$ .
  - b. Un nombre entier e premier avec  $(p-1)(q-1)$  est choisi. Deux nombres sont premiers entre eux s'ils n'ont pas d'autre facteur commun que 1.
  - c. L'entier d est l'entier de l'intervalle  $[2, (p-1)(q-1)]$  tel que  $ed \equiv 1 \pmod{(p-1)(q-1)}$ , c'est-à-dire tel que  $ed - 1$  soit un multiple de  $(p-1)(q-1)$ .
2. **distribution des clefs** : le couple  $(n, e)$  constitue la clef publique de Bob. Il la rend disponible à Alice en lui envoyant ou en la mettant dans un annuaire. Le couple  $(n, d)$  constitue quand à lui sa clef privée.
3. **chiffrement du message** : Pour crypter le message Alice représente le message sous la forme d'un ou plusieurs entiers M compris entre 0 et  $n-1$ . Elle calcule  $C = M^e \pmod{n}$  grâce à la clef publique  $(n, e)$  de Bob et envoie C à Bob.
4. **déchiffrement du message** : Bob reçoit C et calcule grâce à sa clef privée  $C^d \pmod{n}$ . Il obtient ainsi le message initial M.

1. **Création des clés :** Bob crée 5 nombres p,q,n,e et d :

- $p$  et  $q$  sont deux grands nombres premiers distincts. Leur génération se fait au hasard.
  - $n$  est un entier tel que  $n = pq$ .
  - $e$  est un entier premier et  $d$  un entier tel que  $ed=1$  modulo  $[(p-1)(q-1)]$ . Autrement dit,  $ed-1$  est un multiple de  $(p-1)(q-1)$ .

- Exemple simplifié

- Deux petits nombres 1<sup>er</sup> : p = 5 et q = 7
  - $n = 5 * 7 = 35$
  - $n' = (5 - 1) * (7 - 1) = 24$
  - On cherche e et d tels que :
    - e est premier
    - et  $ed = 1$  modulo 24
      - $ed = 1$  Non, trop petit
      - $ed = 25$  Ok, mais  $e = d = 5$  et alors Clé privée = clé publique => Faille de sécurité.
      - $ed = 49$  Pareil,  $e = d$
      - $ed = 73$  73 est 1<sup>er</sup>, raté
      - $ed = 97$  97 est 1<sup>er</sup>
      - $ed = 121$  11 au caré, encore raté
      - $ed = 165$   $165 = 5 * 33$ , et 5 est 1<sup>er</sup> : Ok
  - On à Clé publique = RSA(35, 5) et clé privée = RSA(35, 33).

**2. Distribution des clés :** Le couple  $(n, e)$  constitue la clé publique de Bob. Il la rend disponible par

exemple en la mettant dans un annuaire. Le couple  $(n,d)$  constitue sa clé privée. Il la garde secrète.

3. **Envoi du message codé :** Alice veut envoyer un message codé à Bob. Elle le représente sous la forme d'un ou plusieurs entiers  $M$  compris entre 0 et  $n-1$ . Alice possède la clé publique  $(n, e)$  de Bob. Elle calcule  $C = M^e$  modulo  $n$ . C'est ce dernier nombre qu'elle envoie à Bob.

4. **Réception du message codé** : Bob reçoit C, et il calcule grâce à sa clé privée  $D=C^d \pmod{n}$  Il a donc reconstitué le message initial.

## 1. Choix de la clef

**Alice** choisit deux entiers premiers  $p$  et  $q$  (Ici on prend des nombres à 7 bits soit inférieurs à  $2^7$  (128)) et fait leur produit  $n = p \cdot q$ . Puis elle choisit un entier  $e$  premier avec  $(p-1) \cdot (q-1)$ . Enfin, elle publie dans un annuaire, par exemple sur le web, sa clef publique: **(RSA, n, e)**.

p = 53      q=97      donc n = 5141      e = 7      et d = 4279

## 2. Chiffrement

**Bob** veut donc envoyer un message à **Alice**. Il cherche dans l'annuaire la clef de chiffrement qu'elle a publiée. Il sait maintenant qu'il doit utiliser le système RSA avec les deux entiers **5141** et **7**. Il transforme en nombres son message en remplaçant par exemple chaque lettre par son rang dans l'alphabet.

"J'ÉVOUSAIME" devient : "10 05 22 15 21 19 01 09 13 05".

Puis il découpe son message chiffré en blocs de même longueur (En partant de la droite) représentant chacun un nombre le plus grand possible tout en restant plus petit que  $n$ . Cette opération est essentielle, car si on ne faisait pas des blocs assez longs (par exemple si on laissait des blocs de 2 dans notre exemple), on retomberait sur un simple chiffre de substitution que l'on pourrait attaquer par l'analyse des fréquences.

Son message devient : "010 052 215 211 901 091 305"

Un bloc **B** est chiffré par la formule  $C = B^e \text{ mod } n$ , où **C** est un bloc du message chiffré que **Bob** enverra à **Alice**.

- |       |                              |                        |
|-------|------------------------------|------------------------|
| • 010 | $(10^7) \equiv 5141 = 755$   | 0755 (Sur 4 positions) |
| • 052 | $(52^7) \equiv 5141 = 1324$  | 1324                   |
| • 215 | $(215^7) \equiv 5141 = 1324$ | 2823                   |
| • ... |                              |                        |

Après avoir chiffré chaque bloc, le message chiffré s'écrit : "0755 1324 2823 3550 3763 2237 2052".

### 3. Déchiffrement

**Alice** calcule à partir de **p** et **q**, qu'elle a gardés secrets, la clef **d** de déchiffrage (c'est sa **clef privée**). Celle-ci doit satisfaire l'équation  $e \cdot d \text{ mod } ((p-1)(q-1)) = 1$ . Ici, **d**=4279

Chacun des blocs **C** du message chiffré sera déchiffré par la formule  $B = C^d \text{ mod } n$ .

- |        |                                   |     |
|--------|-----------------------------------|-----|
| • 0755 | $(755^{4279}) \equiv 5141 = 10$   | 010 |
| • 1324 | $(1324^{4279}) \equiv 5141 = 52$  | 052 |
| • 2823 | $(2823^{4279}) \equiv 5141 = 215$ | 215 |
| • ...  |                                   |     |

Elle retrouve : "010 052 215 211 901 091 305"

En regroupant les chiffres deux par deux et en remplaçant les nombres ainsi obtenus par les lettres correspondantes, elle sait enfin que Bob l'aime secrètement, sans que personne d'autre ne puisse le savoir.

Sa robustesse réside dans la difficulté à factoriser un grand nombre.

La sécurité de l'algorithme RSA repose sur la difficulté à factoriser  $n$ .

Pour déchiffrer le message, il est nécessaire de trouver  $d$  connaissant  $e$ , ce qui nécessite de recalculer  $\varphi$ , et donc de connaître  $p$  et  $q$ , les deux facteurs premiers de  $n$ .

Or, la factorisation d'un entier (de très grande taille) en facteurs premiers est extrêmement difficile, cette opération nécessitant une capacité de calcul très importante.

Pour exemple : en 2010, l'INRIA et ses partenaires ont réussi à factoriser une clé de 768 bits. Il leur a fallu deux ans et demi de recherche, et plus de  $10^{20}$  calculs. C'est à ce jour le meilleur résultat connu de factorisation.

Afin de se prémunir contre les puissances de calculs grandissantes, il est régulièrement recommandé d'utiliser des tailles de clés de plus en plus grandes (actuellement de 2048 bits).

### Mise en oeuvre de RSA

#### Alice et Bob

- Bob possède un message confidentiel qu'il souhaite transmettre à Alice.
- Alice construit deux clés :
  - une clé de chiffrement publique qu'elle transmet à Bob.
  - une clé de déchiffrement privée qu'elle conserve soigneusement.
- Bob utilise la clé publique pour chiffrer le message, et le transmet à Alice.
- Alice utilise la clé privée pour déchiffrer le message reçu.

### Génération des clés

- Soient deux grands nombres premiers « aléatoirement » choisis :  $p$  et  $q$ .
- Notons :  $n = p \cdot q$  et  $\varphi = (p-1) \cdot (q-1)$
- Soient  $d$  un grand entier « aléatoirement » choisi, premier avec  $\varphi$ . Et  $e$  l'inverse de  $d$  modulo  $\varphi$ .
- La clé publique de chiffrement est le couple  $(n, e)$ , la clé privée de déchiffrement le couple  $(n, d)$ .

### Chiffrement/Déchiffrement

#### Chiffrement

- Avant d'être chiffré, le message original doit être décomposé en une série d'entiers  $M$  de valeurs comprises entre 0 et  $n-1$ .
- Pour chaque entier  $M$  il faut calculer  $C \equiv M^e \pmod{n}$ .
- Le message chiffré est constitué de la succession des entiers  $C$ .

#### Déchiffrement

- Conformément à la manière dont il a été chiffré, le message reçu doit être composé d'une succession d'entiers  $C$  de valeurs comprises entre 0 et  $n-1$ .
- Pour chaque entier  $C$  il faut calculer  $M \equiv C^d \pmod{n}$ .
- Le message original peut alors être reconstitué à partir de la série d'entiers  $M$ .

### Applications avec une cryptographie asymétrique

■ Transmission sécurisée pour une cryptographie symétrique

■ Mécanismes d'authentification

■ Certificats numériques

■ Infrastructure à clé publique (IGC)

■ Signatures numériques.

Exemples de tels protocoles :

- ▶ PGP
- ▶ GPG
- ▶ Internet Key Exchange
- ▶ ZKIP (protocole VoIP)
- ▶ Secure Socket Layer
- ▶ SLLC
- ▶ SSH
- ▶ Secure Electronic Transaction (SET)
- ▶ Bitcoin

### Chiffrement mixte et authentification :

Les algorithmes à clé publique sont assez lents. La méthode généralement utilisée pour envoyer un message, est de tirer au hasard une clé secrète, chiffrer le message avec un algorithme à clé privée en utilisant cette clé, puis chiffrer cette clé aléatoire elle-même avec la clé publique du destinataire. Ceci permet d'avoir la sécurité des systèmes à clé publique, avec la performance des systèmes à clé privée. Il existe un logiciel qui effectue toutes ces opérations et de manière transparente, et qui, de plus, est gratuit et téléchargeable à partir de dizaines de sites de par le monde : le

célèbre **PGP** de **Phil Zimmermann**. (Il y a d'autres logiciels aussi performants, mais PGP est sûrement le plus connu.). Correctement utilisé, il est sûr, même contre les meilleurs cryptanalystes du monde (c'est d'ailleurs pour cette raison que son utilisation est formellement interdite en France).

**PGP** (*Pretty Good Privacy*) est un cryptosystème (système de chiffrement) inventé par Philip Zimmermann, un analyste informaticien. Philip Zimmermann a travaillé de 1984 à 1991 sur un programme permettant de faire fonctionner RSA sur des ordinateurs personnels (PGP).

Cependant, étant donné que celui-ci utilisait RSA sans l'accord de ses auteurs, cela lui a valu des procès pendant 3 ans, il est donc vendu environ 150\$ depuis 1993.

Il est très rapide et sûr ce qui le rend quasiment impossible à cryptanalyser.

## Le principe de PGP

PGP est un système de cryptographie *hybride*, utilisant une combinaison des fonctionnalités de la cryptographie à clé publique et de la cryptographie symétrique.

Lorsqu'un utilisateur chiffre un texte avec PGP, les données sont d'abord compressées. Cette compression des données permet de réduire le temps de transmission par tout moyen de communication, d'économiser l'espace disque et, surtout, de renforcer la sécurité cryptographique.

La plupart des cryptanalystes exploitent les modèles trouvés dans le texte en clair pour casser le chiffrement. La compression réduit ces modèles dans le texte en clair, améliorant par conséquent considérablement la résistance à la cryptanalyse.

Ensuite, l'opération de chiffrement se fait principalement en deux étapes :

- PGP crée une clé secrète IDEA de manière aléatoire, et chiffre les données avec cette clé
- PGP crypte la clé secrète IDEA et la transmet au moyen de la clé RSA publique du destinataire.

L'opération de déchiffrement se fait également en deux étapes :

- PGP déchiffre la clé secrète IDEA au moyen de la clé RSA privée.
- PGP déchiffre les données avec la clé secrète IDEA précédemment obtenue.

Cette méthode de chiffrement associe la facilité d'utilisation du cryptage de clef publique à la vitesse du cryptage conventionnel. Le chiffrement conventionnel est environ 1000 fois plus rapide que les algorithmes de chiffrement à clé publique. Le chiffrement à clé publique résoud le problème de la distribution des clés. Utilisées conjointement, ces deux méthodes améliorent la performance et la gestion des clefs, sans pour autant compromettre la sécurité.

## Les fonctionnalités de PGP

PGP offre les fonctionnalités suivantes :

- **Signature électronique et vérification d'intégrité de messages** : fonction basée sur l'emploi simultané d'une fonction de hachage (MD5) et du système RSA. MD5 hache le message et fournit un résultat de 128 bits qui est ensuite chiffré, grâce à RSA, par la clef privée de l'expéditeur.
- **Chiffrement des fichiers locaux** : fonction utilisant IDEA.
- **Génération de clefs publiques et privées** : chaque utilisateur chiffre ses messages à l'aide de clefs privées IDEA. Le transfert de clefs électroniques IDEA utilise le système RSA; PGP offre donc des mécanismes de génération de clefs adaptés à ce système. La taille des clefs RSA est proposée suivant plusieurs niveaux de sécurité : 512, 768, 1024 ou 1280 bits.
- **Gestion des clefs** : fonction s'assurant de distribuer la clef publique de l'utilisateur aux correspondants qui souhaiteraient lui envoyer des messages chiffrés.
- **Certification de clefs** : cette fonction permet d'ajouter un sceau numérique garantissant l'authenticité des clefs publiques. Il s'agit d'une originalité de PGP, qui base sa confiance sur une notion de proximité sociale plutôt que sur celle d'autorité centrale de certification.
- **Révocation, désactivation, enregistrement de clefs** : fonction qui permet de produire des certificats de révocation.

## Le format des certificats PGP

Un certificat PGP comprend, entre autres, les informations suivantes :

- **Le numéro de version de PGP** : identifie la version de PGP utilisée pour créer la clef associée au certificat.

- **La clef publique du détenteur du certificat** : partie publique de votre paire de clefs associée à l'algorithme de la clef, qu'il soit RSA, DH (Diffie-Hellman) ou DSA (Algorithme de signature numérique).
- **Les informations du détenteur du certificat** : il s'agit des informations portant sur l'« identité » de l'utilisateur, telles que son nom, son ID utilisateur, sa photographie, etc.
- **La signature numérique du détenteur du certificat** : également appelée autosignature, il s'agit de la signature effectuée avec la clef privée correspondant à la clef publique associée au certificat.
- **La période de validité du certificat** : dates/ heures de début et d'expiration du certificat. Indique la date d'expiration du certificat.
- **L'algorithme de chiffrement symétrique** préféré pour la clef : indique l'algorithme de chiffrement que le détenteur du certificat préfère appliquer au cryptage des informations. Les algorithmes pris en charge sont CAST, IDEA ou DES triple

Le fait qu'un seul certificat puisse contenir plusieurs signatures est l'un des aspects uniques du format du certificat PGP. Plusieurs personnes peuvent signer la paire de clefs/ d'identification pour attester en toute certitude de l'appartenance de la clef publique au détenteur spécifié. Certains certificats PGP sont composés d'une clef publique avec plusieurs libellés, chacun offrant un mode d'identification du détenteur de la clef différent (par exemple, le nom et le compte de messagerie d'entreprise du détenteur, l'alias et le compte de messagerie personnel du détenteur, sa photographie, et ce, dans un seul certificat).

Dans un certificat, une personne doit affirmer qu'une clef publique et le nom du détenteur de la clef sont associés. Quiconque peut valider les certificats PGP. Les certificats X. 509 doivent toujours être validés par une autorité de certification ou une personne désignée par la CA. Les certificats PGP prennent également en charge une structure hiérarchique à l'aide d'une CA pour la validation des certificats.

Plusieurs différences existent entre un certificat X. 509 et un certificat PGP. Les plus importantes sont indiquées ci-dessous :

Pour créer votre propre certificat PGP, vous devez demander l'émission d'un certificat X. 509 auprès d'une autorité de certification et l'obtenir ;

- Les certificats X. 509 prennent en charge un seul nom pour le détenteur de la clef ;
- Les certificats X. 509 prennent en charge une seule signature numérique pour attester de la validité de la clef ;

## Les modèles de fiabilité de PGP

En règle générale, la CA (*Certification authority* - autorité de certification) inspire une confiance totale pour établir la validité des certificats et effectuer tout le processus de validation manuelle. Mais, il est difficile d'établir une ligne de confiance avec les personnes n'ayant pas été explicitement considérées comme fiables par votre CA.

Dans un environnement PGP, tout utilisateur peut agir en tant qu'autorité de certification. Il peut donc valider le certificat de clef publique d'un autre utilisateur PGP. Cependant, un tel certificat peut être considéré comme valide par un autre utilisateur uniquement si un tiers reconnaît celui qui a validé ce certificat comme un correspondant fiable. C'est-à-dire, si l'on respecte par exemple mon opinion selon laquelle les clefs des autres sont correctes uniquement si je suis considéré comme un correspondant fiable. Dans le cas contraire, mon opinion sur la validité d'autres clefs est controversée.

Supposons, par exemple, que votre trousseau de clefs contient la clef d'Alice. Vous l'avez validée et, pour l'indiquer, vous la signez. En outre, vous savez qu'Alice est très pointilleuse en ce qui concerne la validation des clefs d'autres utilisateurs. Par conséquent, vous affectez une fiabilité complète à sa clef. Alice devient ainsi une autorité de certification. Si elle signe la clef d'un autre utilisateur, cette clef apparaît comme valide sur votre trousseau de clefs.

## La révocation d'un certificat PGP

Seul le détenteur du certificat (le détenteur de sa clef privée correspondante) ou un autre utilisateur, désigné comme autorité de révocation par le détenteur du certificat, a la possibilité de révoquer un certificat PGP. La désignation d'une autorité de révocation est utile, car la révocation, par un utilisateur PGP, de son certificat est souvent due à la perte du mot de passe complexe de la clef privée

correspondante. Or, cette procédure peut uniquement être effectuée s'il est possible d'accéder à la clef privée. Un certificat X.509 peut uniquement être révoqué par son émetteur.

Lorsqu'un certificat est révoqué, il est important d'en avertir ses utilisateurs potentiels. Pour informer de la révocation des certificats PGP, la méthode habituelle consiste à placer cette information sur un serveur de certificats. Ainsi, les utilisateurs souhaitant communiquer avec vous sont avertis de ne pas utiliser cette clef publique.

**CERTIFICAT :** L'**authentification** permet de prouver son identité à travers le réseau. Il utilise généralement la technique des signatures électroniques. L'envoyeur joint une signature électronique à son message.

Les algorithmes de chiffrement asymétrique sont basés sur le partage entre les différents utilisateurs d'une clé publique. Généralement le partage de cette clé se fait au travers d'un [annuaire électronique](#) (généralement au format [LDAP](#)) ou bien d'un site web.

Toutefois ce mode de partage a une grande lacune : **rien ne garantit que la clé est bien celle de l'utilisateur à qui elle est associée**. En effet un pirate peut corrompre la clé publique présente dans l'annuaire en la remplaçant par sa clé publique. Ainsi, le pirate sera en mesure de déchiffrer tous les messages ayant été chiffrés avec la clé présente dans l'annuaire.

Ainsi un certificat permet d'associer une clé publique à une entité (une personne, une machine, ...) afin d'en assurer la validité. Le certificat est en quelque sorte la carte d'identité de la clé publique, délivré par un organisme appelé *autorité de certification* (souvent notée **CA** pour *Certification Authority*).

L'autorité de certification est chargée de délivrer les certificats, de leur assigner une date de validité (équivalent à la date limite de péremption des produits alimentaires), ainsi que de révoquer éventuellement des certificats avant cette date en cas de compromission de la clé (ou du propriétaire).

## Structure d'un certificat ?

Les certificats sont des petits fichiers divisés en deux parties :

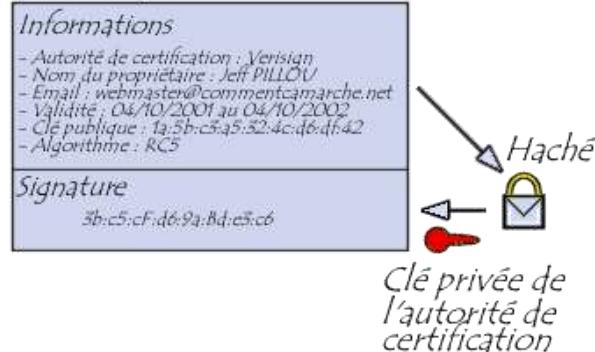
- La partie contenant les informations
- La partie contenant la signature de l'autorité de certification

La structure des certificats est normalisée par le standard **X.509** de l'[UIT](#) (plus exactement X.509v3), qui définit les informations contenues dans le certificat :

- La version de X.509 à laquelle le certificat correspond ;
- Le numéro de série du certificat ;
- L'algorithme de chiffrement utilisé pour signer le certificat ;
- Le nom (DN, pour *Distinguished Name*) de l'autorité de certification émettrice ;
- La date de début de validité du certificat ;
- La date de fin de validité du certificat ;
- L'objet de l'utilisation de la clé publique ;
- La clé publique du propriétaire du certificat ;
- La signature de l'émetteur du certificat (*thumbprint*).

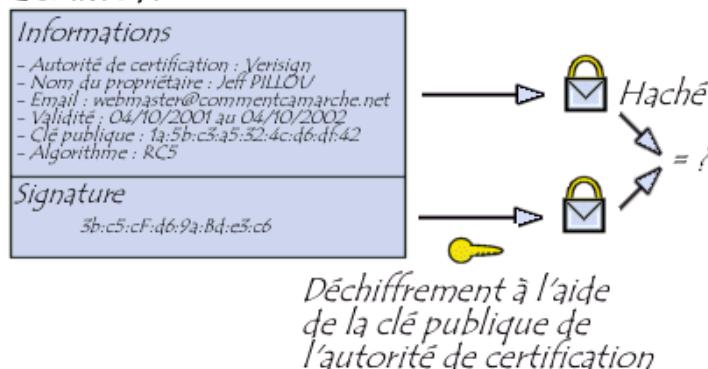
L'ensemble de ces informations (informations + clé publique du demandeur) est signé par l'autorité de certification, cela signifie qu'une [fonction de hachage](#) crée une empreinte de ces informations, puis ce condensé est chiffré à l'aide de la clé privée de l'autorité de certification; la clé publique ayant été préalablement largement diffusée afin de permettre aux utilisateurs de vérifier la signature avec la clé publique de l'autorité de certification.

## Certificat



Lorsqu'un utilisateur désire communiquer avec une autre personne, il lui suffit de se procurer le certificat du destinataire. Ce certificat contient le nom du destinataire, ainsi que sa clé publique et est signé par l'autorité de certification. Il est donc possible de vérifier la validité du message en appliquant d'une part la fonction de hachage aux informations contenues dans le certificat, en déchiffrant d'autre part la signature de l'autorité de certification avec la clé publique de cette dernière et en comparant ces deux résultats.

## Certificat



## Signatures de certificats

On distingue différents types de certificats selon le niveau de signature :

- Les **certificats auto-signés** sont des certificats à usage interne. Signés par un serveur local, ce type de certificat permet de garantir la confidentialité des échanges au sein d'une organisation, par exemple pour le besoin d'un intranet. Il est ainsi possible d'effectuer une authentification des utilisateurs grâce à des certificats auto-signés.
- Les **certificats signés par un organisme de certification** sont nécessaires lorsqu'il s'agit d'assurer la sécurité des échanges avec des utilisateurs anonymes, par exemple dans le cas d'un site web sécurisé accessible au grand public. Le certificateur tiers permet d'assurer à l'utilisateur que le certificat appartient bien à l'organisation à laquelle il est déclaré appartenir.

## Types d'usages

Les certificats servent principalement dans trois types de contextes :

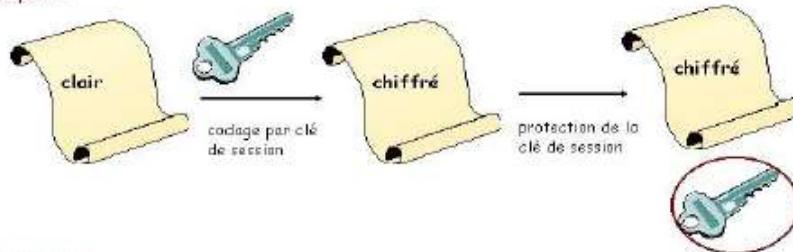
- Le **certificat client**, stocké sur le poste de travail de l'utilisateur ou embarqué dans un conteneur tel qu'une carte à puce, permet d'identifier un utilisateur et de lui associer des droits. Dans la plupart des scénarios il est transmis au serveur lors d'une connexion, qui affecte des droits en fonction de l'accréditation de l'utilisateur. Il s'agit d'une véritable carte d'identité numérique utilisant une paire de clé asymétrique d'une longueur de 512 à 1024 bits.
- Le **certificat serveur** installé sur un serveur web permet d'assurer le lien entre le service et le propriétaire du service. Dans le cas d'un site web, il permet de garantir que l'URL et en particulier le domaine de la page web appartiennent bien à telle ou telle entreprise. Par ailleurs il permet de sécuriser les transactions avec les utilisateurs grâce au protocole SSL.
- Le **certificat VPN** est un type de certificat installé dans les équipements réseaux, permettant de chiffrer les flux de communication de bout en bout entre deux points (par exemple deux sites d'une entreprise). Dans ce type de scénario, les utilisateurs possèdent un certificat client,

les serveurs mettent en oeuvre un certificat serveur et les équipements de communication utilisent un certificat particulier (généralement un certificat [IPSec](#)).

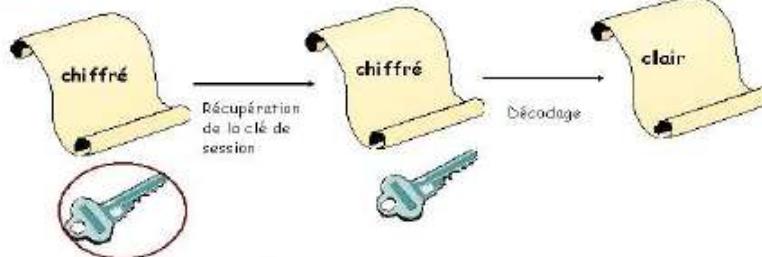
PGP utilise le meilleur de la cryptographie symétrique (rapide du chiffrement) et de la cryptographie asymétrique (sécurité de l'échange de clés). Il fonctionne suivant le principe suivant :

1. Compression : le texte à envoyer est compressé. Cette étape permet de réduire le temps de transmission des données, et améliore également la sécurité. En effet, la compression détruit les modèles du texte (fréquences des lettres, mots répétés). On sait que ces modèles sont souvent utilisés dans les analyses cryptographiques.
2. Chiffrement du message : un mot de passe aléatoire est générée (on parle ici de clé de session), et le message est chiffré par un algorithme symétrique à l'aide de cette clé de session. L'algorithme utilisé varie au cours du temps : il s'agissait au début de DES, puis de CAST.
3. Chiffrement de la clé de session : la clé de session est chiffrée en utilisant la clé publique du destinataire (et l'algorithme RSA).
4. Envoi et réception du message : l'expéditeur envoie le couple message chiffré / clé de session chiffrée au destinataire. Celui récupère d'abord la clé de session, en utilisant sa clé privée, puis il déchiffre le message grâce à la clé de session.

Au départ :



A l'arrivée :



#### Principe de chiffrement du PGP

Le PGP implemente aussi les fonctions de certificat et de signature électroniques. Il n'y a pas d'autorité centrale de certification, mais un grand rôle joue à la proximité sociale : les amis de mes amis sont mes amis !

Un certificat X509 : • prouve l'identité d'une personne au même titre qu'une carte d'identité, dans le cadre fixé par l'autorité de certification qui l'a validé ; • pour une application, il assure que celle-ci n'a pas été détournée de ses fonctions ; • pour un site, il offre la garantie lors d'un accès vers celui-ci que l'on est bien sur le site auquel on veut accéder.

**S/MIME** (pour *Secure MIME*, soit *Secure Multipurpose Mail Extension*, que l'on pourrait traduire par *extensions du courrier électronique à but multiples et sécurisées*) est un procédé de sécurisation des échanges par courrier électronique permettant de garantir la confidentialité et la non-répudiation des messages électroniques.

S/MIME est basé sur le standard MIME, dont le but est de permettre d'inclure dans les messages électroniques des fichiers attachés autres que des fichiers texte ([ASCII](#)). C'est ainsi grâce au standard MIME qu'il est possible d'ajouter des pièces jointes de tous types aux courriels électroniques.

S/MIME a été mis au point à l'origine par la société *RSA Data Security*. Ratifié en juillet 1999 par l'IETF, S/MIME est devenu un standard, dont les spécifications sont contenues dans les RFC 2630 à 2633.

## Principe de fonctionnement de S/MIME

Le standard S/MIME repose sur le principe de chiffrement à clé publique. S/MIME permet ainsi de chiffrer le contenu des messages mais ne chiffre pas la communication.

Les différentes parties d'un message électronique, codées selon le standard MIME, sont chacunes chiffrées à l'aide d'une clé de session.

Dans chaque en-tête de partie est insérée la clé de session, chiffrée à l'aide de la clé publique du destinataire. Seul le destinataire peut ainsi ouvrir le corps du message, à l'aide de sa clé privée, ce qui assure la confidentialité et l'intégrité du message reçu.

Par ailleurs, la signature du message est chiffrée à l'aide de la clé privée de l'expéditeur. Toute personne interceptant la communication peut lire le contenu de la signature du message, mais cela permet de garantir au destinataire l'identité de l'expéditeur, car seul l'expéditeur est capable de chiffrer un message (avec sa clé privée) déchiffrable à l'aide de sa clé publique.

SSL (**Secure Sockets Layers**, que l'on pourrait traduire par *couche de sockets sécurisée*) est un procédé de sécurisation des transactions effectuées via Internet. Le standard SSL a été mis au point par *Netscape*, en collaboration avec *Mastercard*, *Bank of America*, *MCI* et *Silicon Graphics*. Il repose sur un procédé de [cryptographie par clef publique](#) afin de garantir la sécurité de la transmission de données sur internet. Son principe consiste à établir un canal de communication sécurisé (chiffré) entre deux machines (un client et un serveur) après une étape d'[authentification](#).

Le système SSL est indépendant du [protocole utilisé](#), ce qui signifie qu'il peut aussi bien sécuriser des transactions faites sur le Web par le [protocole HTTP](#) que des connexions via le [protocole FTP, POP](#) ou [IMAP](#).

En effet, SSL agit telle une couche supplémentaire, permettant d'assurer la sécurité des données, située entre la couche [application](#) et la couche [transport \(protocole TCP\)](#) par exemple).

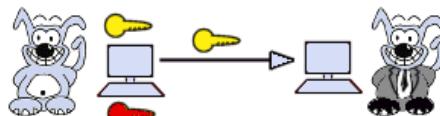
De cette manière, SSL est transparent pour l'utilisateur (entendez par là qu'il peut ignorer qu'il utilise SSL). Par exemple un utilisateur utilisant un [navigateur internet](#) pour se connecter à un site de commerce électronique sécurisé par SSL enverra des données chiffrées sans aucune manipulation nécessaire de sa part.

La quasi intégralité des navigateurs supporte désormais le protocole SSL. *Mozilla Firefox* affiche par exemple un cadenas verrouillé pour indiquer la connexion à un site sécurisé par SSL et un cadenas ouvert dans le cas contraire, tandis que *Microsoft Internet Explorer* affiche un cadenas uniquement lors de la connexion à un site sécurisé par SSL.

## Fonctionnement de SSL 2.0

La sécurisation des transactions par SSL 2.0 est basée sur un échange de clés entre [client](#) et [serveur](#). La transaction sécurisée par SSL se fait selon le modèle suivant :

- Dans un premier temps, le client se connecte au site marchand sécurisé par SSL et lui demande de s'authentifier. Le client envoie également la liste des cryptosystèmes qu'il supporte, triée par ordre décroissant selon la longueur des clés.
- Le serveur à réception de la requête envoie un [certificat](#) au client, contenant la clé publique du serveur, signée par une autorité de certification (CA), ainsi que le nom du cryptosystème le plus haut dans la liste avec lequel il est compatible (la longueur de la clé de chiffrement - 40 bits ou 128 bits - sera celle du cryptosystème commun ayant la plus grande taille de clé).



- Le client vérifie la validité du certificat (donc l'authenticité du marchand), puis crée une clé secrète aléatoire (plus exactement un bloc prétendument aléatoire), chiffre cette clé à l'aide de la clé publique du serveur, puis lui envoie le résultat (la [clé de session](#)).
- Le serveur est en mesure de déchiffrer la clé de session avec sa clé privée. Ainsi, les deux entités sont en possession d'une clé commune dont ils sont seuls connaiseurs. Le reste des transactions peut se faire à l'aide de clé de session, garantissant l'intégrité et la confidentialité des données échangées.

SSL 3.0 vise à authentifier le serveur vis-à-vis du client et éventuellement le client vis-à-vis du serveur.

## CERTIFICATS D'AUTHENTIFICATION

Nous avons vu lors de la cryptographie à clé publique et à clé secrète, qu'il subsistait une faille dans le système de transmission des clés au début de la communication.

Ainsi dans les deux systèmes, la faille réside dans le fait que quelqu'un de malveillant puisse se substituer à l'interlocuteur réel et envoyer ainsi soit une fausse clé secrète, soit une fausse clé publique (en fonction des cas).

Ainsi, un certificat d'authenticité permet d'associer une clé à une entité (une personne, une machine, ...) afin d'en assurer la validité. Le certificat est en quelque sorte la carte d'identité de la clé ou la "**signature numérique**", délivré par un organisme appelé "**autorité de certification**".

La technologie faisant usage des signatures numériques fait partie d'un ensemble plus vaste connu sous l'acronyme "**PKI**" (Public Key Infrastructure). L'ensemble se déroule moyennant des certificats que vous pouvez obtenir auprès d'une Autorité de certification (voir exemple plus bas). Lorsque vous demandez votre certificat, votre ordinateur crée la paire de clefs composées d'une clé privée (la jaune sur le schéma) et une clé publique (la noire). Votre clé privée est secrète et c'est seulement vous qui y avez accès alors que la clé publique est librement disponible pour tout le monde. Votre clef publique sera attachée à votre certificat que vous obtiendrez de la part de l'autorité de certification à qui vous avez remis votre demande de certificat.

Le PKI vise essentiellement 4 points importants:

1. l'authentification (le destinataire de votre e-mail doit pouvoir vérifier que c'est bien vous qui avez envoyé l'objet et pas un autre). Une personne peut intercepter votre mail, extraire votre mot de passe etc..).
2. l'intégrité (s'assurer que le contenu n'a pas été changé en chemin).
3. la confidentialité (s'assurer que le contenu n'est lisible que par le destinataire).
4. la non-réputation (découlant des 3 premiers points)

L'autorité de certification est chargée de délivrer les certificats, de leur assigner une date de validité (1 jour), ainsi que de révoquer éventuellement des certificats avant cette date en cas de compromission de la clé.

Les certificats sont des petits fichiers divisés en deux parties :

- La partie contenant les informations
- La partie contenant la signature de l'autorité de certification (voir Internet Explorer)

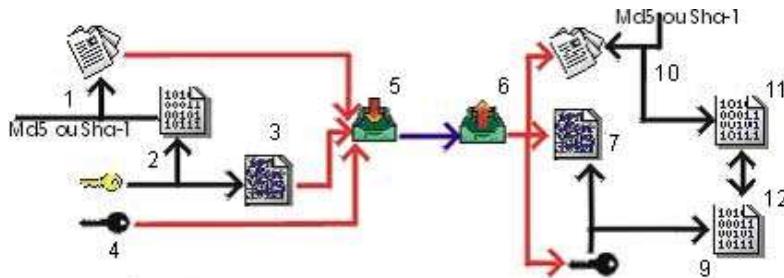
La structure des certificats est normalisée par le standard X.509 de l'UIT, qui définit les informations contenues dans le certificat :

- Le nom de l'autorité de certification (VeriSign)
- Le nom du propriétaire du certificat (UBS)
- La date de validité du certificat (1 jour à partir de la date courante)
- L'algorithme de chiffrement utilisé (MD5RSA)
- La clé publique du propriétaire

Voici un très bon exemple venant d'un confrère (T. Taglicht - [www.taglicht.com](http://www.taglicht.com)):

Pour signer le message que vous expédiez (point (5) sur le schéma), il suffit en effet de lui appliquer une fonction de hachage (point (1) sur le schéma) qui produit un résumé (code haché) du message (les algorithmes de hachage les plus connus sont MD5 (128 bits (Message Digest 5)) et SHA-1 (160

bits (Secure Hash Algorithm 1)). Le résumé obtenu est propre à chaque message, à l'image d'une empreinte digitale. Cet algorithme assure que si un seul bit du texte original est modifié et si l'on refaisait un nouveau hachage (empreinte), ce dernier serait radicalement différent du premier. Le code haché peut ensuite être chiffré à l'aide de votre clé privée et annexé à votre message (points (2) et (3) sur le schéma). C'est ce code qui constitue la signature numérique. Le destinataire du message peut ensuite vérifier que vous en êtes bien l'expéditeur en déchiffrant la signature numérique, au moyen de votre clé publique (que vous lui avez transmis automatiquement avec le mail (point (4) sur le schéma), pour obtenir le code haché (point (9) sur le schéma). Le destinataire applique ensuite la même fonction de hachage au message reçu (point (10) sur le schéma); si les deux codes (points 11 et 12 sur le schéma) sont identiques, vous êtes bien l'expéditeur du message (authentification) et le message n'a pas été altéré (intégrité).



Les fonctions de hachage permettent de s'assurer de l'intégrité d'un message mais un autre problème se pose : comment être certain que personne n'a usurpé l'identité de l'expéditeur pour vous envoyer un message ? Ou que l'expéditeur ne va pas nier vous l'avoir envoyé ?

C'est le rôle de la signature numérique, celle-ci fournissant donc les services d'intégrité des données, d'authentification de l'origine des données et de non-répudiation.

La façon la plus simple de signer un message est d'utiliser la cryptographie asymétrique pour le chiffrer en utilisant sa clef privée : seul le possesseur de cette clef peut générer la signature et toute personne ayant accès à la clef publique correspondante peut la vérifier. Mais cette méthode est très lente et en pratique elle n'est que peu utilisée.

La méthode réellement utilisée repose non pas sur le chiffrement du message lui-même mais sur l'empreinte (empreinte issue d'une fonction de hachage comme MD5 par exemple) de celui-ci. En effet, cette méthode est beaucoup plus rapide du fait de la quantité réduite des données à chiffrer.

Une signature numérique est plus sûre qu'une signature papier car la signature change à chaque message. Elle est de ce fait inimitable (sans la connaissance de la clef secrète bien entendue).

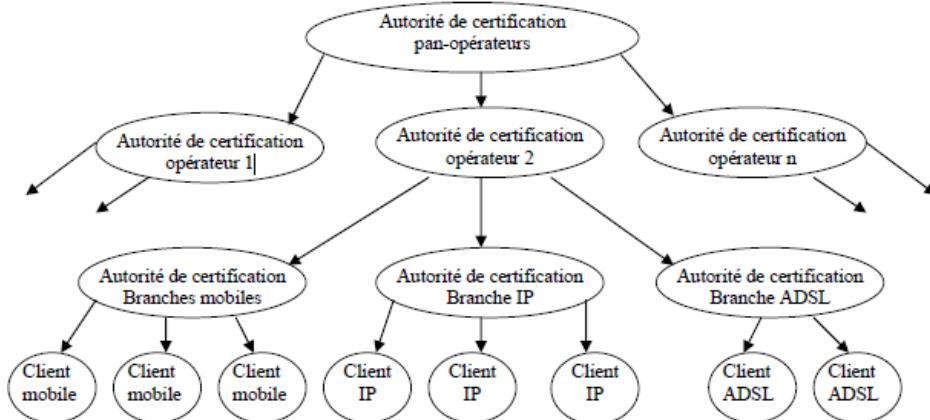
### **Infrastructure à clef publique (ICP)**

L'infrastructure à clef publique définit un ensemble de services de sécurité afin de rendre les échanges électroniques fiables.

A l'intérieur de l'ICP, les éléments sont organisés hiérarchiquement dans le but de garantir un niveau de sécurité élevé. Elle est composée de :

- une autorité d'enregistrement (Registration Authority)
- une autorité de certification
- un système de distribution des clefs

Les ICP sont évolutives et interopérables c'est-à-dire qu'elles sont capables de suivre la croissance du nombre d'utilisateurs et doivent supporter l'ajout de nouvelles autorités de certification et l'établissement de certification croisée entre plusieurs autorités.



#### Relation hiérarchique dans une ICP simple

Le problème qui a amené à créer les certificats est l'opposé de celui qui a amené à créer les signatures numériques. Un certificat permet d'attester l'identité du destinataire de la même façon qu'un papier d'identité permet d'identifier une personne. En effet, les certificats numériques fonctionnent sur le même principe : ils sont émis par un organisme supérieur (l'Etat pour le papier d'identité) appelés autorité de certification.

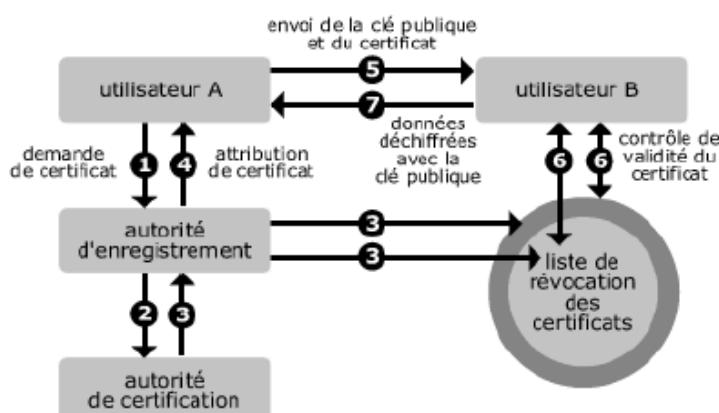
Une autorité d'enregistrement est un organisme approuvé pour vérifier que les autres organismes sont bien ce qu'ils prétendent être. Pour cela elle applique des procédures d'identification conformes aux règles définies par l'autorité de certification.

#### **Obtenir un certificat numérique**

Pour obtenir un certificat d'une autorité d'enregistrement, il faut donc fournir à celle-ci des informations (documents officiels) prouvant votre identité ou celle de votre organisation ainsi que votre clef publique. L'autorité vérifie alors votre identité mais aussi que la clef fournie est bien issue d'un algorithme de cryptage à clef publique particulier.

Il existe trois classes de certificats correspondant à différents niveaux de sécurité :

- certificats de classe 1 : Le demandeur ne fournit qu'une adresse e-mail.
- certificats de classe 2 : Ils requièrent une preuve d'identité du demandeur.
- certificats de classe 3 : Ces certificats ne peuvent être délivrés que si le demandeur est présent physiquement.



#### **Accéder aux informations du certificat**

Une fois le certificat racine de l'autorité de certification intégré au navigateur Internet, le destinataire peut déchiffrer celui-ci et identifier son auteur en comparant les deux condensés (condensé issu de la signature et condensé calculé grâce à une fonction de hachage – le plus souvent MD5 ou SHA-1).

#### **Cycle de vie d'un certificat**

Pour des raisons de sécurité, un certificat est accordé pour une durée limitée. Il peut être également remis en question dans la période de validité pour différentes raisons :

- volonté du détenteur du certificat
- changement de situation du détenteur
- volonté de l'autorité de certification
- sécurité

Le certificat est alors suspendu ou révoqué, la suspension ou la révocation étant notifiée dans un annuaire spécifique facilement accessible (en ligne).

Le format de certificats numériques X.509 de l'ISO (Organisation Internationale de Normalisation) est le plus répandu. Un certificat X.509 est composé de la signature de l'autorité de certification et d'informations.

Les informations contiennent :

- \_ la version de la norme X.509 désigné par un nombre entier : 0 pour la version 1, 1 pour la version 2 et 2 pour la version 3.
- \_ le numéro de série : chaque certificat a un numéro unique.
- \_ l'algorithme de signature utilisé par l'autorité : MD5, DSA,etc...
- \_ la période de validité du certificat
- \_ le nom de l'autorité de certification
- \_ le nom du sujet
- \_ renseignements sur la certification de la clef publique :
  - o algorithme utilisé
  - o chaîne de bits représentant la clef publique
- \_ informations optionnelles spécifiques à la version 3

La version 3 de X.509 est la version actuelle, mais toutes les versions sont utilisées. La version 3 donne la possibilité d'ajouter des extensions personnalisées (optionnelles) aux certificats.

La cryptographie a cle publique permet de s'affranchir du probleme de l'echange de la cle, facilitant le travail de l'expediteur. Mais comment s'assurer de l'authenticite de l'envoi? Comment etre sur que personne n'usurpe l'identite d'Alice pour vous envoyer un message? Comment etre sur qu'Alice ne va pas nier vous avoir envoyee ce message? La encore, la cryptographie a cle publique peut resoudre ce probleme. Alice veut donc envoyer un message crypte a Bob, mais Bob veut s'assurer que ce message provient bien d'Alice.

Ici on va appeler Pa la cle publique d'Alice, Sa sa cle privee. Pb et Sb pour Bob. Et M le message a envoyer par Alice.

· **Phase d'envoi :** Alice calcule  $SA(M)$ , a l'aide de sa cle secrete, puis  $PB(SA(M))$ , a l'aide de la cle publique de Bob.

· **Phase de réception :** A l'aide de sa cle privee, Bob calcule  $Sb(PB(SA(M)))=SA(M)$ . Seul lui peut effectuer ce calcul (=securite de l'envoi). Puis il calcule  $PA(SA(M))=M$ . Il est alors sur que c'est Alice qui lui a envoyee ce message, car elle-seule a pu calculer  $SA(M)$ .

Contrairement a certains utilisateurs, vous ne confondrez plus << signature electronique >> et << image d'une signature ajoutee en bas de page >>

Remarquons pour terminer qu'une signature numerique est plus sure qu'une signature papier, car elle est infalsifiable, inimitable : la signature change en effet a chaque message!

## Distribution des clefs publiques

- Idée de départ
  - ◆ Simple annuaire des clefs publiques
- Problèmes à résoudre
  - ◆ Distribuer les clefs de façon authentifiée et intègre
  - ◆ Stocker les clefs de façon sûre (protection en intégrité)
    - Limiter le nombre de clefs à stocker
- Solution = certificats et hiérarchies de certification
  - ◆ Élément de transport d'une clef publique, dont l'authenticité est vérifiable de façon autonome
  - ◆ Authentification : Lie une clef publique à son possesseur
  - ◆ Intégrité : Toute modification du certificat sera détectée

- Certificat = Structure de données
  - ◆ Permet de lier une clef publique à différents éléments, au moyen de la signature d'une autorité de confiance :
    - Nom du propriétaire de la clef
    - Dates de validité
    - Type d'utilisation autorisée
    - ...
  - ◆ Format actuel : X.509v3, profil PKIX
- Émis par une autorité de certification (*Certificate Authority – CA*)
  - ◆ Garantit l'exactitude des données
  - ◆ Certificats vérifiables au moyen de la clef publique de la CA, seule clef à stocker de façon sûre
- Listes de révocation (*Certificate Revocation List – CRL*)
  - ◆ Permettent de révoquer des certificats avant leur expiration normale



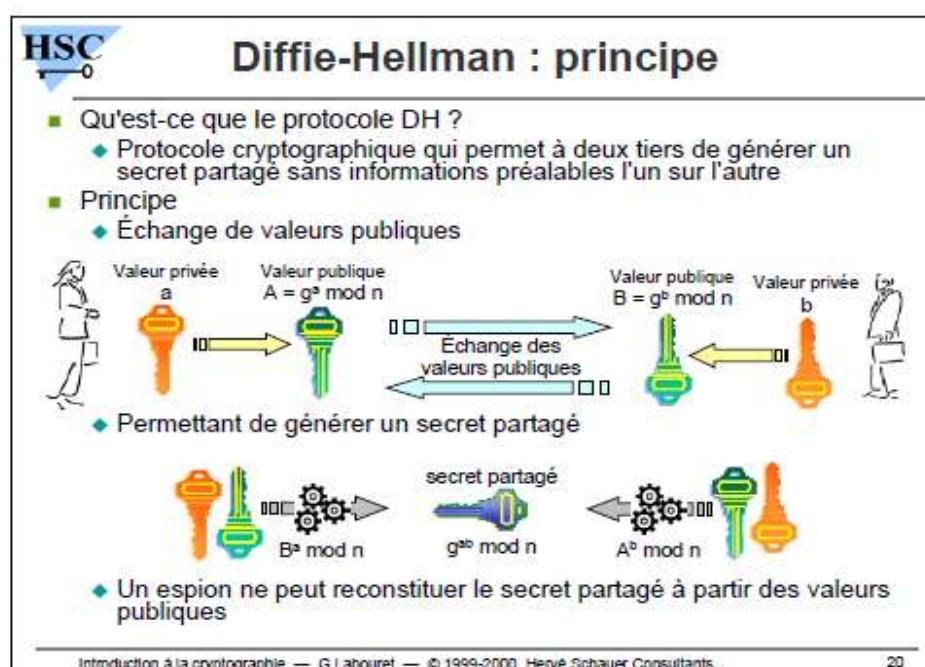
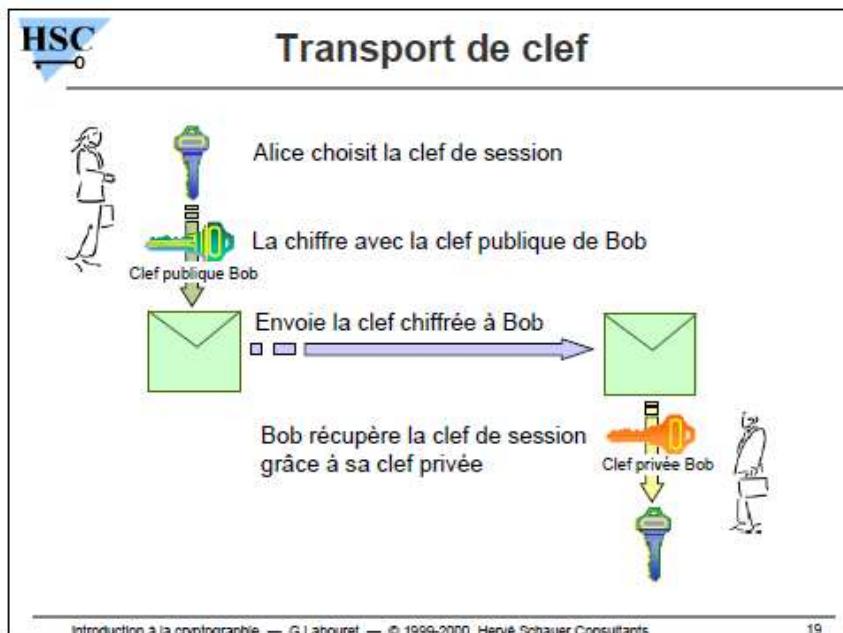
## Émission et vérification des certificats



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

- Relations entre échange de clefs et authentification mutuelle
  - ◆ L'échange de clefs doit être authentifié pour éviter les attaques
  - ◆ Une clef de session permet d'étendre l'authentification à l'ensemble de la communication
  - ◆ Protocole d'authentification mutuelle avec échange de clefs
    - fournit authentification mutuelle et un échange de clefs authentifié tout-en-un
- Types d'échange de clefs
  - ◆ Transport
    - Exemple : transport RSA (utilisé par SSL)
  - ◆ Génération
    - Exemple : Diffie-Hellman

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



Inventé en 1976 par Diffie et Hellman, ce protocole permet à deux tiers de générer un secret partagé sans avoir aucune information préalable l'un sur l'autre. Il est basé sur la cryptologie à clef publique (dont il est d'ailleurs à l'origine), car il fait intervenir des valeurs publiques et des valeurs privées. Sa sécurité dépend de la difficulté de calculer des logarithmes discrets sur un corps fini. Le secret généré à l'aide de cet algorithme peut ensuite être utilisé pour dériver une ou plusieurs clefs (clef secrète, clef de chiffrement de clefs...).

Voici le déroulement de l'algorithme :

1. Alice et Bob se mettent d'accord sur un grand entier  $n$  tel que  $(n-1)/2$  soit premier et sur un entier  $g$  primitif par rapport à  $n$ . Ces deux entiers sont publics.
2. Alice choisit de manière aléatoire un grand nombre entier  $a$ , qu'elle garde secret, et calcule sa valeur publique,  $A = g^a \text{ mod } n$ . Bob fait de même et génère  $b$  et  $B = g^b \text{ mod } n$ .
3. Alice envoie  $A$  à Bob ; Bob envoie  $B$  à Alice.
4. Alice calcule  $K_{AB} = B^a \text{ mod } n$  ; Bob calcule  $K_{BA} = A^b \text{ mod } n$ .  $K_{AB} = K_{BA} = g^{ab} \text{ mod } n$  est le secret partagé par Alice et Bob.

Une personne qui écoute la communication connaît  $g$ ,  $n$ ,  $A = g^a \text{ mod } n$  et  $B = g^b \text{ mod } n$ , ce qui ne lui permet pas de calculer  $g^{ab} \text{ mod } n$  : il lui faudrait pour cela calculer le logarithme de  $A$  ou  $B$  pour retrouver  $a$  ou  $b$ .

- Sensible à l'attaque de l'intercepteur
  - ◆ L'attaquant, Ingrid, envoie sa valeur publique la place d'Alice et de Bob et partage ainsi un secret avec chaque tiers.
  - ◆ Solution: authentifier les valeurs publiques ; le protocole résultant s'appelle Diffie-Hellman authentifié.
- Propriété de *Perfect Forward Secrecy (PFS)*
  - ◆ Principe
    - La découverte du secret à long terme ne compromet pas les clefs de session
    - Propriété fournie lorsque le secret à long terme n'intervient pas dans la génération ou la protection en confidentialité des clefs
  - ◆ DH authentifié fournit la PFS si les seules valeurs à long terme sont celles utilisées pour l'authentification (i.e. les valeurs privées/publiques sont à court terme)

En revanche, Diffie–Hellman est vulnérable à l'attaque active dite attaque de l'intercepteur .

Une façon de contourner le problème de l'attaque de l'intercepteur sur Diffie–Hellman est d'**authentifier les valeurs publiques utilisées pour la génération du secret partagé**. On parle alors de **Diffie–Hellman authentifié**. L'authentification peut se faire à deux niveaux :

- En utilisant des valeurs publiques authentifiées, à l'aide de certificats par exemple. Cette méthode est notamment à la base du protocole SKIP.
- En authentifiant les valeurs publiques après les avoir échangées, en les signant par exemple. Cette méthode est utilisée entre autres par les protocoles Photuris et IKE.

L'inconvénient, dans les deux cas, est que l'on perd un gros avantage de Diffie–Hellman, qui est la possibilité de générer un secret partagé sans aucune information préalable sur l'interlocuteur.

Mais Diffie–Hellman, même authentifié, fournit une propriété intéressante, appelée propriété de *Perfect Forward Secrecy (PFS)*. Cette propriété est garantie si la découverte par un adversaire du ou des secrets à long terme ne compromet pas les clefs de session générées précédemment : les clefs de session passées ne pourront pas être retrouvées à partir des secrets à long terme. On considère généralement que cette propriété assure également que la découverte d'une clef de session ne compromet ni les secrets à long terme ni les autres clefs de session.

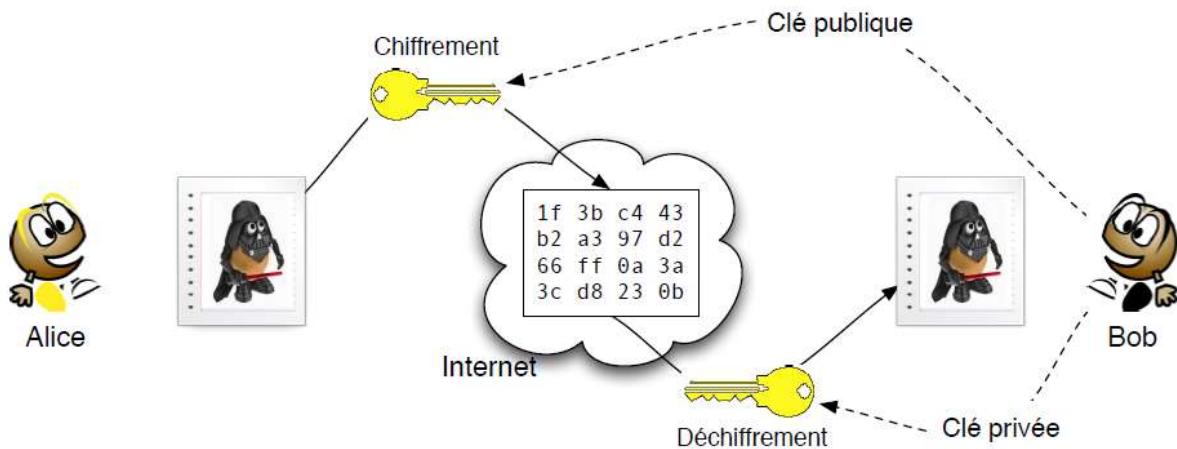
## Chiffrement

- On **chiffre** avec la clé **publique** de l'interlocuteur A
- On **déchiffre** avec la clé **privée** de A

## Signature

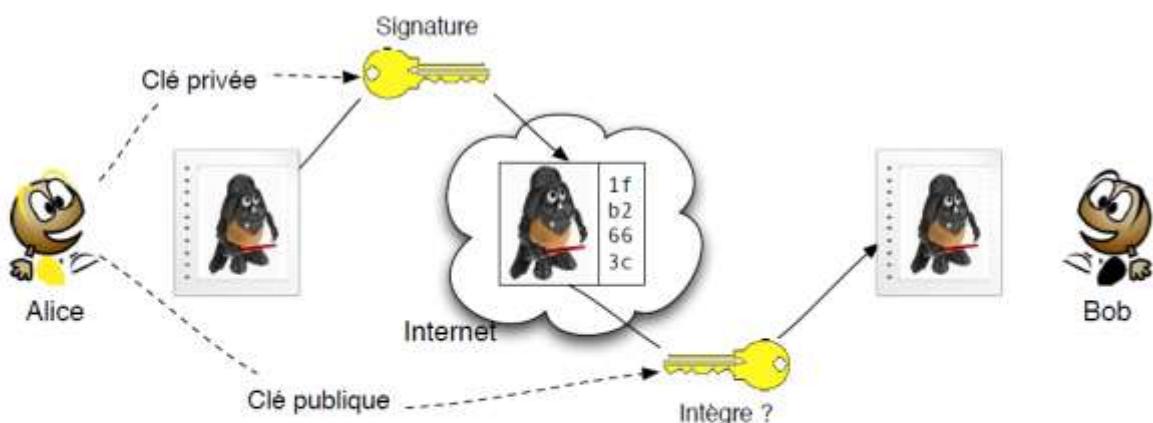
- On **signe** avec sa propre clé **privée**
- On **vérifie** la signature avec la clé **publique** associée

## CHIFFREMENT



Chiffrement avec la clé **publique** du destinataire,  
déchiffrement avec sa clé **privée**  
⇒ Seul le destinataire peut déchiffrer le message

## SIGNATURE



Signature avec la clé **privée** de l'émetteur,  
vérification avec sa clé **publique**  
⇒ Seul l'émetteur a pu signer le message de manière valide

DSA signifie Digital Signature Algorithm (Algorithm de Signature Digitale). Il s'agit d'un algorithme inventé en 1991 aux Etats-Unis par le **National Institute of Standards and Technology** (NIST) et adopté par le **Federal Information Processing Standard** (FIPS) en 1993. L'algorithme DSA fut utilisé en premier lieu pour signer électroniquement des données, mais on l'utilise désormais à la fois comme algorithme de signature et de chiffrement dans les certificats SSL.

La méthode DSA est essentiellement utilisée par les services publics américains, car il s'agit d'une méthode approuvée par les agences fédérales, et qui correspond donc aux critères de sécurité requis pour les services publics.

Il est tout à fait possible de **combiner les algorithmes RSA et DSA** sur un même serveur (c'est le cas notamment des serveurs sous Apache) pour garantir un niveau de sécurité optimal.

Le DSA est similaire à un autre type de signature développée par [Claus-Peter Schnorr](#) ([en](#)) en 1989. Il a aussi des points communs avec la signature [ElGamal](#). Le processus se fait en trois étapes :

- génération des clés //signature du document // vérification du document signé