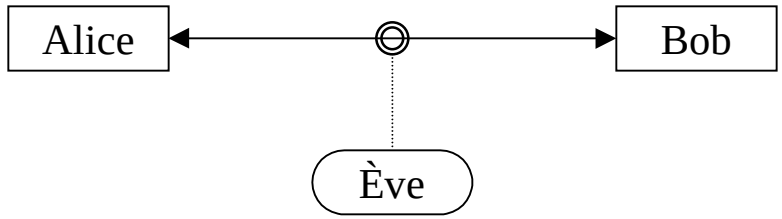
Chapitre 7
Cryptologie

Cryptographie

- Il y a 10 ans, la plupart des gens sérieux croyaient que la cryptographie n'avait que peu de rapports avec la sécurité.
- Aujourd'hui, c'est le contraire: elle est vue comme la panacée, le remède à tous nos problèmes de sécurité.
- La *cryptographie* étudie les manières de camoufler les données.
- La *cryptanalyse* étudie les manières de déjouer ces techniques de camouflage.
- La *cryptologie* réunit ces deux aspects.

Le vieux paradigme



- Puise ses racines dans la sécurité de la communication
  - Deux partenaires veulent communiquer, mais le canal utilisé n'est pas sécurisé (ex: téléphone, courrier).
  - En effet, un adversaire peut écouter, bloquer ou modifier les transmissions effectuées sur ce canal.
- Mécanismes cryptographiques:
  - Confidentialité des données, via un algorithme de chiffrement de données (ex: chiffre de César)
  - Intégrité des données, via une fonction de vérification d'intégrité (ex: SHA-1)
  - Authentification de l'origine des données, via un algorithme de signature (ex: RSA)

De nouvelles approches

- Dans le commerce électronique, nous avons
  - Un marchant qui se méfie des escrocs
  - Un consommateur qui se méfie des marchants malhonnêtes
- Pour régler la question de la confiance, ils utilisent un protocole qui ne repose pas sur une confiance mutuelle.
- Un tiers parti de confiance servira à arbitrer les litiges.
- Dans plusieurs pays, les forces policières peuvent mettre un suspect sous écoute, après obtention d'un mandat.
  - Si les communications sont chiffrés, il faudra permettre aux policiers de déchiffrer la communication sans permettre à personne d'autre de le faire.
  - La mise en tutelle des clés cryptographiques utilisées auprès d'agences accréditées permet d'atteindre ce but.

Clés cryptographiques

- L'analogie *serrurière* est très populaire:
  - Pour verrouiller et déverrouiller une serrure, il faut une clé.
  - Les clés diffèrent en force et en taille.
  - Certaines serrures sont faciles à faciles à crocheter.
  - D'autres sont si difficiles qu'il est plus facile d'essayer la force brutale, en défonçant la porte ou en passant par la fenêtre.
- Les algorithmes cryptographiques utilisent des clés pour protéger les données:
  - Elles varient en taille et en force.
  - Certaines peuvent être brisées par simple analyse difficile, alors d'autres sont hors de portée des outils d'analyse d'aujourd'hui.
  - La force brutale correspond à la fouille systématique de toutes les valeurs possibles des clés.

Clés cryptographiques (2)

- La cryptographie moderne ne se base plus sur le secret de ses algorithmes:
  - La clé doit être le seul élément qui doit être protégé.
  - Un algorithme qui doit sa force à son secret sera défait par une fuite, un pot-de-vin, un chantage ou un vol.
  - La publication des algorithmes cryptographiques instaure une sélection naturelle, semblable à celle de Darwin: c'est la survie du plus apte à résister aux attaques.
  - De plus en plus, les algorithmes cryptographiques font l'objet de normes et les nouveaux algorithmes sont soumis à l'évaluation publique pour une période donnée.
    - Ex: Sélection de AES, successeur à DES = Rijndael

Gestion de clés

- La gestion des clés cryptographiques est la clé (!) de voûte de la sécurité des schémas cryptographiques:
  - Où les clés sont-elles générées?
  - Comment les clés sont-elles générées?
  - Où les clés sont-elles emmagasinées?
  - Comment y parviennent-elles?
  - Où les clés sont-elles véritablement utilisées?
  - Comment les clés sont-elles révoquées et remplacées?

Gestion de clés (2)

- À ce point, nous revenons à la sécurité informatique:
  - Les clés cryptographiques sont des données sensibles dans un système informatique.
  - Les mécanismes de contrôle d'accès doivent protéger ces clés. Quand le contrôle d'accès lâche, la protection cryptographique est compromise.
  - Dans plusieurs systèmes actuels, la cryptographie est la composante la plus forte, donc souvent moins attaquée.
  - La cryptographie traduit un problème de sécurité de la communication en un problème de gestion de clés, et donc en un problème de sécurité informatique.
  - La cryptographie n'est pas la sécurité informatique à elle toute seule, mais elle peut la renforcer.

Arithmétique modulaire

- Plusieurs algorithmes cryptographiques reposent sur des opérations d'arithmétique modulaire.
- L'arithmétique modulaire est similaire à l'arithmétique ordinaire ($+$, $-$, $\times$, $\div$), sauf qu'à chaque opération, on ramène le résultat modulo un entier qu'on appele le *module*.
  - Par exemple, en arithmétique modulo 17, on aura $5 + 14 = 2$, $6 - 13 = 10$, $4 * 8 = 15$, $3 \div 14 = 16$
- Lorsque le module est un nombre premier p, l'ensemble $\{0, \ldots, p - 1\}$ forme un corps:
  - L'addition est commutative et associative.
  - La multiplication est commutative, associative et distributive sur l'addition.
  - Chaque élément possède un inverse additif et multiplicatif.

Arithmétique modulaire (2)

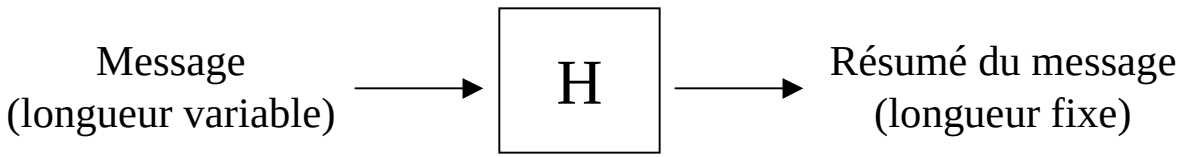
- 2 opérations modulaires sont particulièrement utilisées:
  - L'inverse multiplicatif
  - L'exponentiation (modulaire)
- Un résulat intéressant nous vient du Petit Théorème de Fermat:
  - Si p est un nombre premier, alors pour tout élément $a \neq 0$, on aura que $a^{p-1} = 1 \bmod p$.
  - Par corollaire, on aura que $a^{p} = a \bmod p$.
- C'est ce genre de propriété qui sera utilisée pour construire plusieurs cryptosystèmes.

Arithmétique modulaire (3)

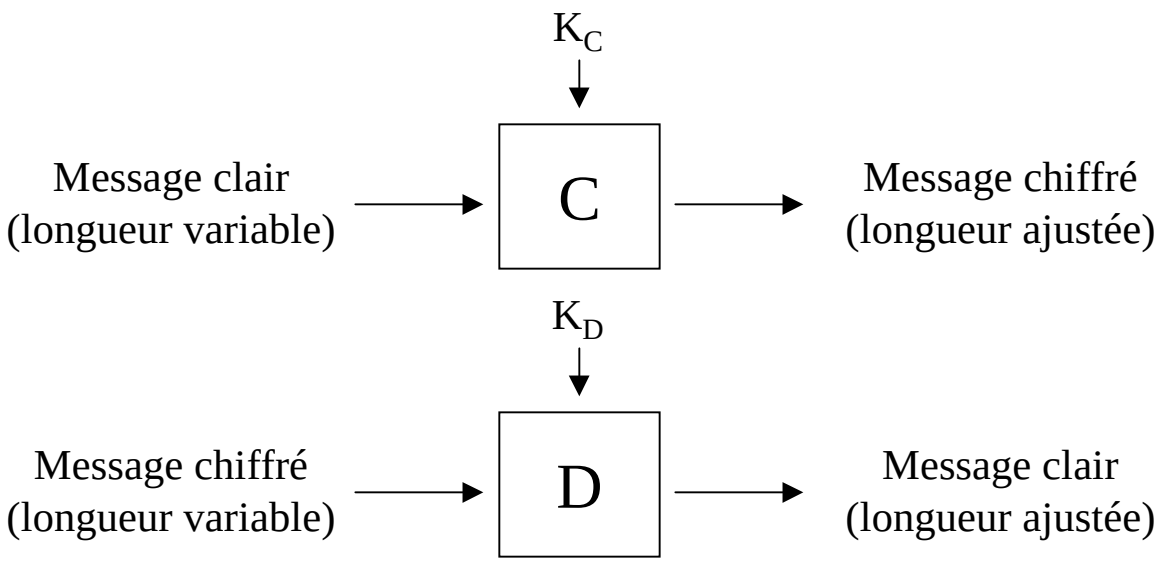
- **Problème du logarithme discret**: Étant donnés un nombre premier p, une base a et un nombre y, il faut trouver x tel que $a^x \equiv y \bmod p$.
- **Problème de la n$^{\text{ième}}$ racine**: Étant donnés des entiers m, n et a, il faut trouver b tel que $b^n \equiv a \bmod p$.
- **Problème de factorisation**: Étant donné un entier n, il faut trouver ses facteurs.
- En choisissant avec soin les paramètres, ces problèmes forment une base adéquate pour plusieurs algorithmes cryptographiques.
  - Si p ou n sont petits, ces problèmes sont facilement résolus.
  - Aujourd'hui, 1024 bits correspond à une taille sécuritaire.

Mécanismes cryptographiques

- Fonctions de hachage cryptographiques

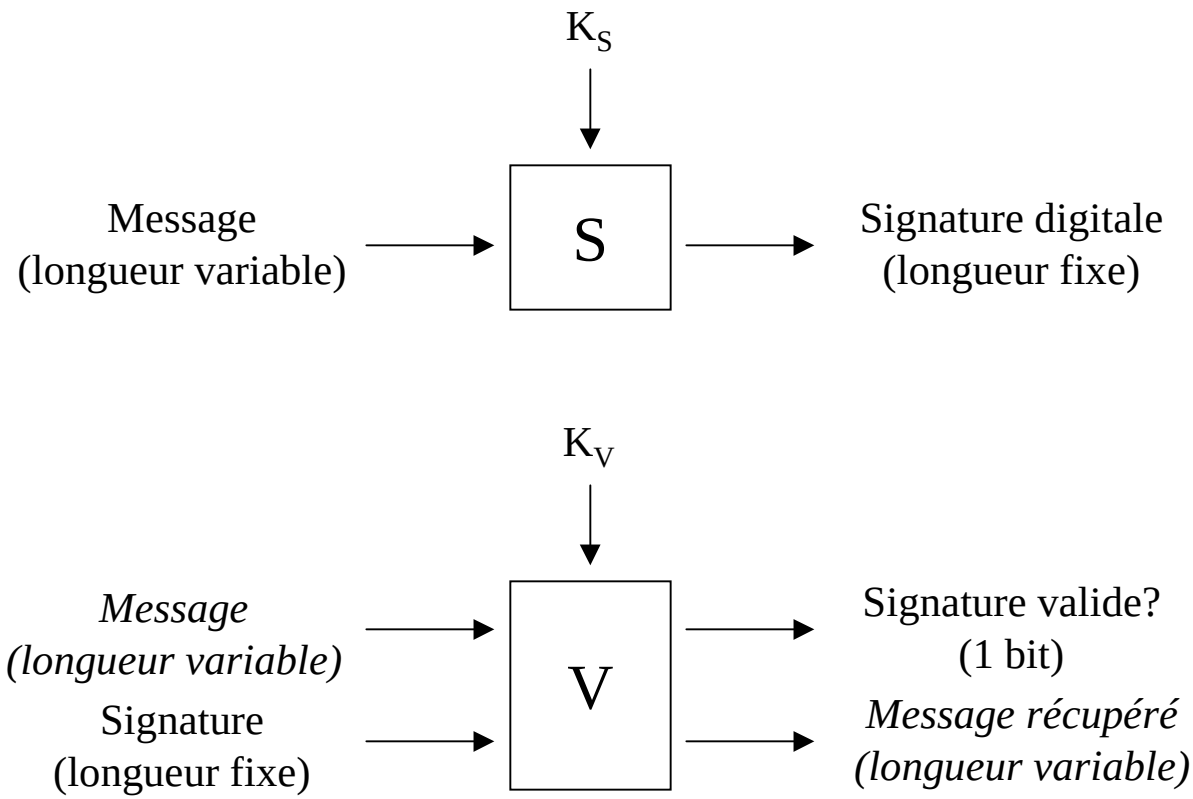


- Algorithmes de chiffrement



Mécanismes cryptographiques (2)

- Algorithmes de signature digitale



Fonctions de hachage

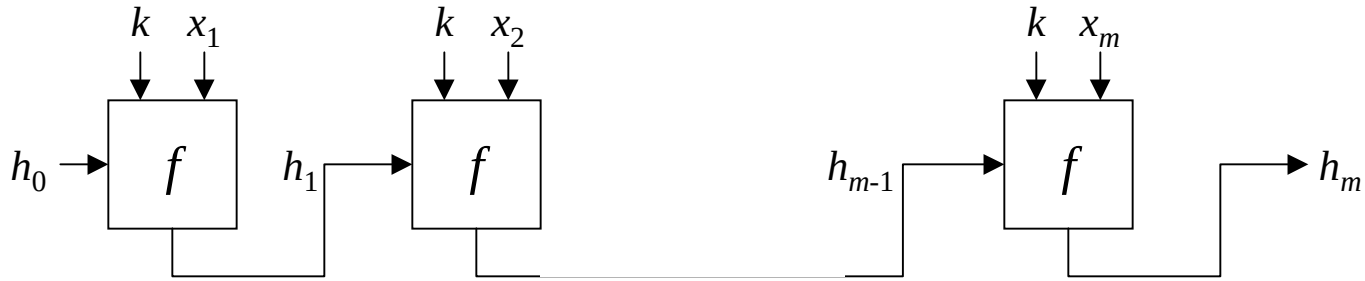
- Les *fonctions de hachage cryptographiques* sont des fonctions possédant des propriétés particulières:
  - Elles sont facile à calculer.
  - Elles compressent un entrée de longueur arbitraire vers un résultat de longueur fixe.
    - On aura un découpage de l'entrée en blocs et une récurrence impliquant une fonction f appliquée sur chaque nouveau bloc:

      $h_i = f(x_i \,||\, h_{i-1})$, pour $i = 1, ...$
  - Elles sont résistantes au calcul de la pré-image.
    - Étant donné y, il est "impossible" de trouver x tel que $h(x) = y$.
    - Étant donnés x et $h(x)$, il est "impossible" de trouver $x' \neq x$ tel que $h(x) = h(x')$.
  - Elles sont résistantes aux collisions.
    - Il est difficile de trouver x et x', $x \neq x'$ tels que $h(x) = h(x')$.

Fonctions de hachage (2)

- Une *fonction de hachage à sens unique* possèdent les 3 premières propriétés, alors qu'une *fonction de hachage résistante aux collisions* les a toutes.
- On utilise souvent les fonctions de hachage comme valeurs auto-vérificatrices (checksums).
  - À ne pas confondre avec les LRC ou CRC utilisés en télécommunications.
- Un *code d'authentification de message* (MAC) est calculé en incorporant un clé cryptographique dans le calcul de la fonction de hachage.
  - Pour toute valeur de clé inconnue de l'adversaire, étant données un ensemble de paires $(x_i, h_k(x_i))$, il doit être "impossible" de calculer $h_k(x)$ pour toute nouvelle valeur de x.

Fonctions de hachage (3)



- Il y a 2 grandes familles de MAC:
  - Basé sur des fonctions de hachage (HMAC)
    - SHA-1, MD4, MD5
  - Basé sur un algorithme cryptographique avec mode de chaînage
    - DES avec mode CBC

Fonctions de hachage (4)

- SHA-1 est une fonction de hachage cryptographique.
  - Présentée dans la norme FIPS 180-1.
  - Elle accepte des blocs de 512 bits et produit un résultat de 160 bits.
    - Le dernier ne doit comprendre que 448 bits, car l'algorithme lui ajoute la longueur totale de l'entrée sur 64 bits.
  - Son état initial (h_0) est constant.
  - Chaque nouveau bloc est traité dans un boucle de 80 étapes, découpée en 4 sous-boucles de 20 étapes.
    - Le bloc est d'abord étendu dans un tableau de 80 entrées.
    - Pour une étape donnée, on effectue une opération propre à la sous-boucle et on incorpore l'entrée pour cette étape.
  - L'état à la fin de la boucle correspond au résultat de SHA-1.
- Un pseudo-code pour l'algorithme est inclus dans les 2 prochaines pages (pour les curieux...)

Fonctions de hachage (5)

- État initial

```
A = 0x67452301
B = 0xEFCDAB89
C = 0x98BADCFE
D = 0x10325476
E = 0xC3D2E1F0
```

- Calcul du tableau étendu

```
pour t = 0 à 15
   w[t] = m[t]
pour t = 16 à 79
   w[t] = (w[t-3] ^ w[t-8] ^ w[t-14] ^ w[t-16]) <<< 1
```

- Fonctions pour chaque sous-boucle

$f_t(X, Y, Z) = (X \& Y) \mid (\sim X \& Z)$ t = 0 à 19

$f_t(X, Y, Z) = X \wedge Y \wedge Z$ t = 20 à 39

$f_t(X, Y, Z) = (X \& Y) \mid (X \& Z) \mid (Y \& Z)$ t = 40 à 59

$f_t(X, Y, Z) = X \wedge Y \wedge Z$ t = 60 à 79

Fonctions de hachage (6)

- Constantes de sous-boucle

  K_t = 0x5A827999 t = 0 à 19
  K_t = 0x6ED9EBA1 t = 20 à 39
  K_t = 0x8F1BBCDC t = 40 à 59
  K_t = 0xCA62C1D6 t = 60 à 79

- Boucle principale

```
pour t = 0 à 79
   temp = (A <<< 5) + f_t (B, C, D) + E + w[t] + K_t
   E = D
   D = C
   C = B <<< 30
   B = A
   A = temp
```

Algorithmes de chiffrement

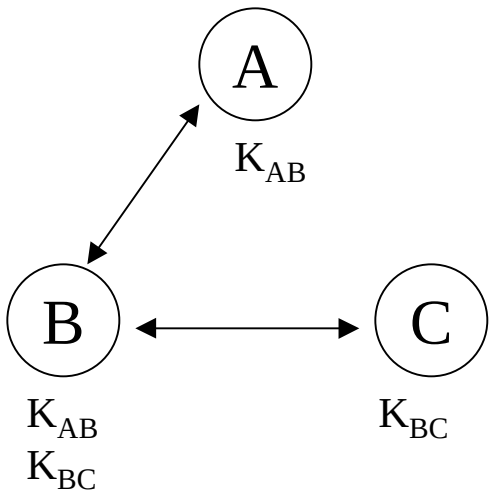
- Un *algorithme de chiffrement* permet de chiffrer un message clair sous le contrôle d'une clé cryptographique de chiffrement, souvent notée K_C.
- Le *déchiffrement* d'un message chiffré en un message clair se fait sous le contrôle d'une clé de déchiffrement, souvent notée K_D.
- Plus formellement, le chiffrement et le déchiffrement sont des fonctions. Nous utiliserons la notation suivante:
  - Chiffrement d'un message clair *m* par la clé K_C
    - $C[K_C](m) = c$
  - Déchiffrement d'un message chiffré *c* par la clé K_D
    - $D[K_D](c) = m$

Algorithmes de chiffrement (2)

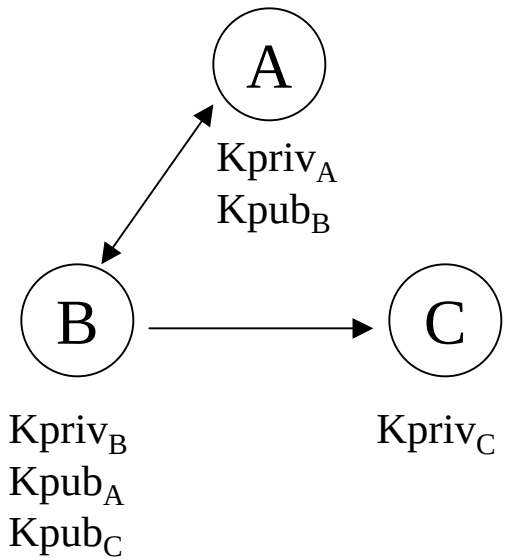
- Un algorithme *symétrique* de chiffrement utilise la même clé pour chiffrer et déchiffrer ($K_C = K_D$).
  - Aussi appelé *algorithme à clé secrète*
  - La clé *secrète* sera notée K_{AB}, si elle est partagée entre A et B, ou simplement K.
  - Ex: DES, IDEA
- Un algorithme *asymétrique* de chiffrement n'utilise pas les mêmes clés pour chiffrer et déchiffrer ($K_C \neq K_D$).
  - Aussi appelé *algorithme à clé publique*
  - On dira que la clé de chiffrement est *publique* (notée Kpub) et la clé de déchiffrement est *privée* (notée Kpriv).
  - Les 2 clés sont reliées de façon algorithmique, mais il doit être "impossible" de déterminer la clé privée à partir de la clé publique.
  - Ex: RSA, El Gamal

Algorithmes de chiffrement (3)

Symétrique

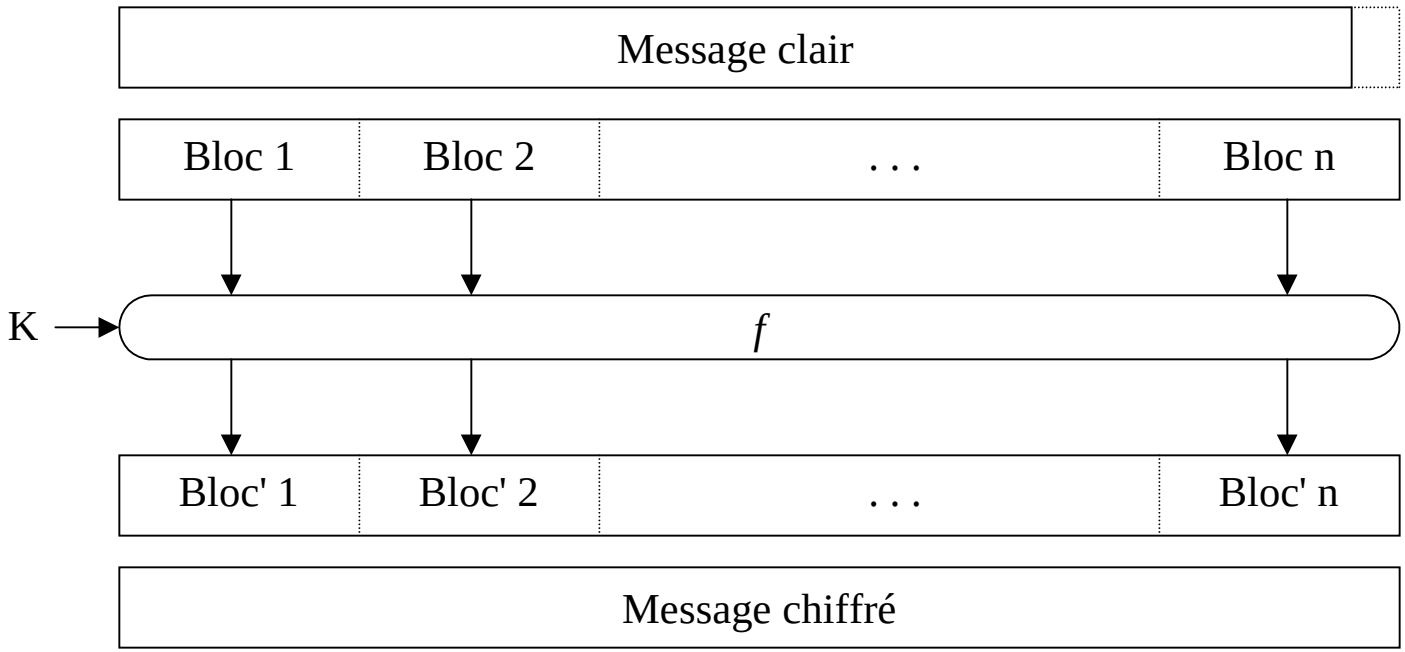


Asymétrique



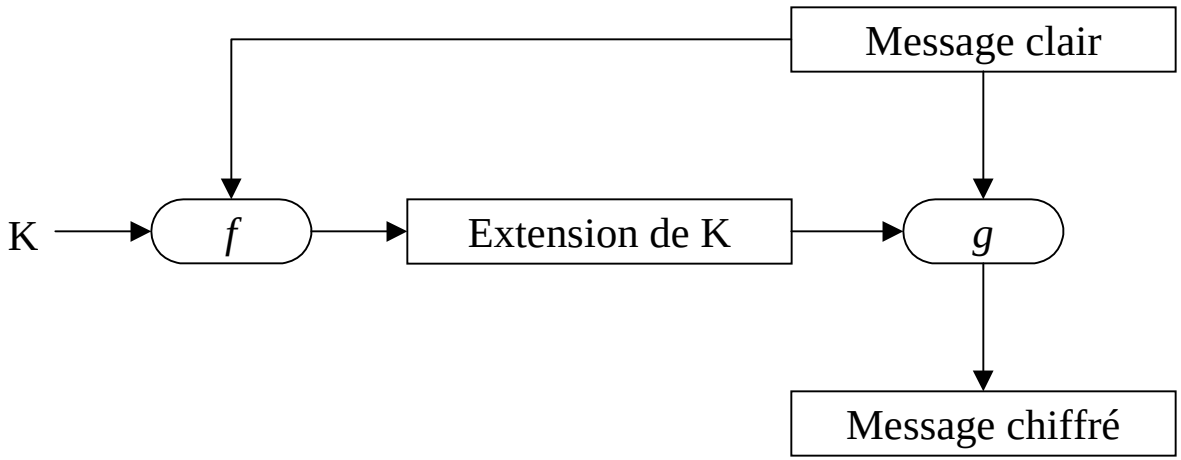
Algorithmes symétriques

- Algorithmes de chiffrement par bloc



Algorithmes symétriques (2)

- Algorithmes de chiffrement par flux



Masque jetable (one-time pad)

- L'algorithme de chiffrement le plus sécuritaire.
  - Taille de la clé = taille du message à chiffrer
- Pour chiffrer, on calcule le OU exclusif bit à bit entre le message et la clé.

Message clair:	1	0	1	1	1	0	1	0	1	0	1	1	0	1	0	1
								⊕								
Clé (masque):	0	1	1	0	1	0	1	1	0	0	1	0	0	1	1	0
								=								
Message chiffré:	1	1	0	1	0	0	0	1	1	0	0	1	0	0	1	1

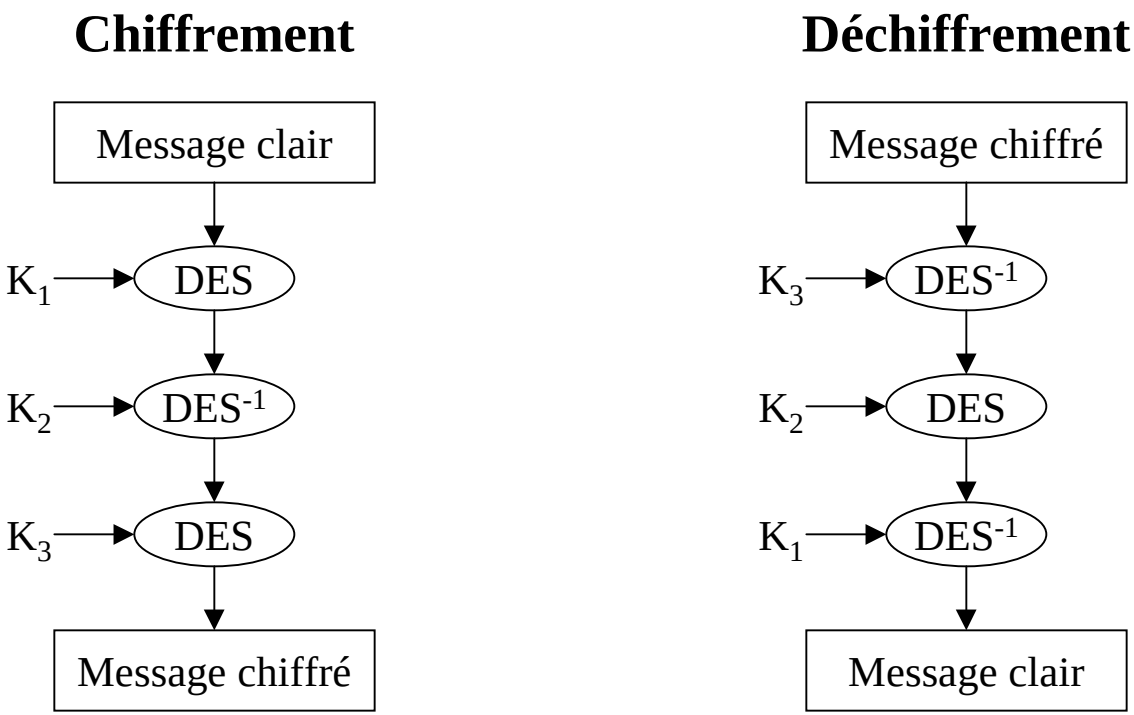
- Problème: On ne peut réutiliser la même plus d'une fois (d'où l'expression "masque jetable").
  - En calculant le OU exclusif entre 2 messages chiffrés, on annule l'effet de la clé.

Algorithme DES

- Vient de l'anglais: Data Encryption Standard
- Développé dans les années '70 en tant que standard américain pour la protection des données non classifiés.
- Publié dans FIPS 86.
- Estimation de sa vie "utile": 15 ans
- Accepte un bloc de 64 bits en entrée, utilise une clé de 56 bits et produit un bloc de 64 bits en sortie.
- Certains systèmes utilisent encore DES, même si sa sécurité n'est plus assurée.
  - Le "DES Cracker" de Gilmour

Variation sur le DES

- Une façon de renforcer la sécurité du DES est d'augmenter la taille de l'espace de clé:
  - Triple DES, combinant 3 clés simples



AES: Successeur à DES

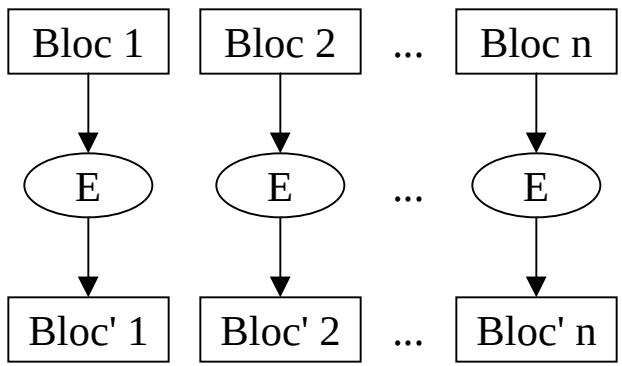
- Choisi parmi 15 algorithmes candidats lors d'un concours organisé par le NIST (National Institute of Standards and Technology).
- Aussi connu sous le nom de Rijndael
- Algorithme de chiffrement par bloc, mais qui permet d'utiliser des clés de 3 longueurs: 128, 192 et 256 bits
- Pas encore accepté par le gouvernement américain comme un standard FIPS (Federal Institute on Products and Services), mais çà ne saurait tarder...
- Le NIST estime qu'AES devrait rester sécure pour au moins 20 ans.

Modes de chaînage

- Les algorithmes de chiffrement par bloc ne peuvent que chiffrer ou déchiffrer un bloc à la fois.
- Pour chiffrer un document, il faudra:
  - Découper le document en bloc accepté par l'algorithme
  - Chiffrer les blocs en appliquant un *mode de chaînage* entre ces blocs.
  - Combiner les blocs résultants
- Les modes de chaînage les plus utilisés sont:
  - ECB: Electronic Code Book
  - CBC: Cipher Block Chaining

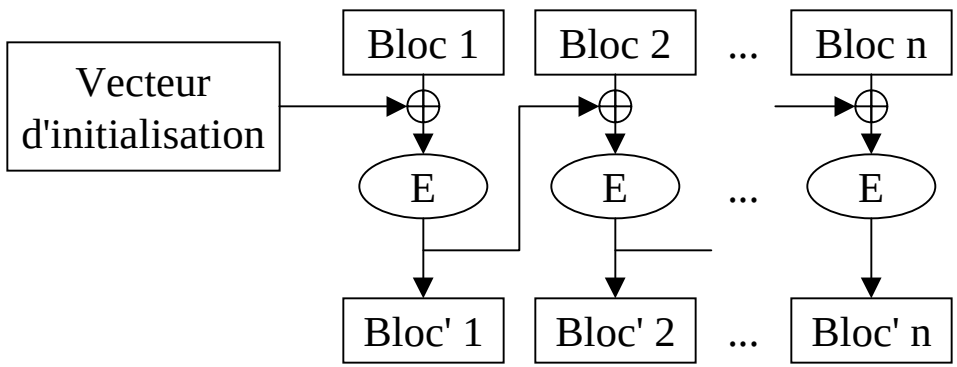
Modes de chaînage (2)

- Le mode ECB (Electronic Code Book) correspond au chiffrement bloc à bloc d'un message.
  - 2 blocs identiques génèrent un chiffrement identiques.
  - Peu de protection sur l'intégrité du message clair (interversion, duplication ou retrait de blocs)



Modes de chaînage (3)

- Le mode CBC (Cipher Block Chaining) enchaîne le chiffrement du bloc précédent avec celui du bloc courant.
  - 2 blocs identiques ont peu de chances de générer le même chiffrement.
  - Si corruption du message chiffré, l'erreur ne touchera qu'à 2 blocs de message clair.



Modes de chaînage (4)

- Voici un exemple d'utilisation des modes de chaînage:
  - Clé = 01234567 89ABCDEF
  - Vecteur d'initialisation CBC = 00000000 00000000
  - Donnée (texte) = "le mode CBC est plus secure que le mode ECB"

Donnée (hex)	Chiffrement ECB (hex)	Chiffrement CBC (hex)
6C65206D 6F646520	**FBE6E473 F2315C30** ⟷	**FBE6E473 F2315C30**
43424320 65737420	2F4E6ED9 FB724E64	85320411 F1390A95
706C7573 20736563	89E9A0C2 9792F466	B4D29D75 0930DD3D
75726520 71756520	7776A373 B1F15F0D	5C3D1CB7 5D93761D
6C65206D 6F646520	**FBE6E473 F2315C30** ⟷	**4D700722 F8B9067C**
45434280 00000000	3B1148BC 9E67064B	697E82C7 A8E8269C

Padding

Algorithme de chiffrement RSA

- Conçu par Rivest, Shamir et Adleman (1978)
- C'est un algorithm de chiffrement asymétrique.
- La clé de chiffrement (clé publique) est formée de:
  - Un module, noté *n*, correspondant au produit de 2 grands nombres premiers
  - Un exposant public, noté *e* (souvent 3 ou 65537)
- La clé de déchiffrement (clé privée) est formée de:
  - Un exposant privé, noté *d*, calculé à partir de *n* et *e*
- Pour chiffrer un message, on le découpe en blocs tels que chaque bloc est un entier inférieur à *n*. Pour chaque bloc *m*,
  - $c = m^e \bmod n$
  - $c^d = m \bmod n$

Algorithme de chiffrement RSA (2)

- Dans le cas du déchiffrement, un algorithme plus rapide de RSA requiert le précalcul d'éléments. On les appelle les éléments CRT (Chinese Remainder Theorem):
  - Les nombres premiers p et q, ayant servi à calculer n
  - Les exposants $dp = d \bmod p - 1$ et $dq = d \bmod q - 1$
  - Le coefficient $iq = q^{-1} \bmod p$
- Si on a ces 5 éléments, on n'a plus besoin du module n, ni de l'exposant privé d.
- Cette version de l'algorithme de déchiffrement est plus rapide, mais les éléments occupent 125% plus d'espace.

Algorithme de chiffrement El Gamal

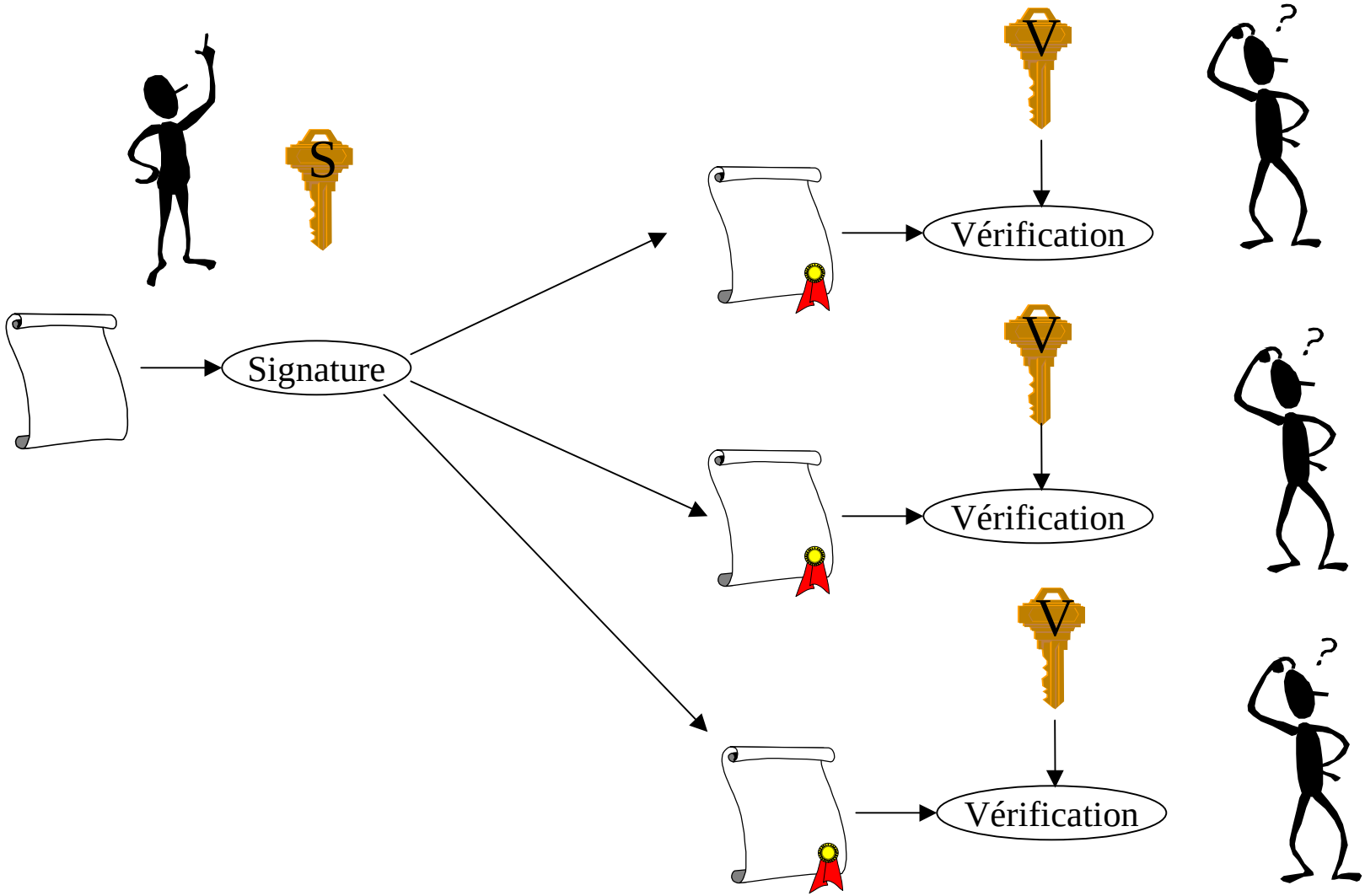
- Publié par El Gamal en 1987
- Soit p, un entier premier de grande taille.
- Soit g, un entier de grand ordre modulo p.
- Soit a, la clé de déchiffrement de A et $y_a = g^a \bmod p$, sa clé publique de chiffrement.
- Pour chiffrer un message, on le découpe en blocs tels que chaque bloc est un entier inférieur à n. Pour chaque bloc m,
  - Générer au hasard k entre 0 et p - 1 et calculer $r = g^k \bmod p$.
  - Le chiffrement est $(c_1, c_2) = (r, my_a^k)$.
- Pour déchiffrer, A utilise sa connaissance de a et calcule:

$$\frac{c_2}{c_1^a} = \frac{my_a^k}{r^a} = \frac{mg^{ak}}{g^{ak}} = m$$

Signatures digitales

- Un *schéma de signature digitale* comprend un algorithme de signature et un algorithme de vérification.
  - Une signature dépendra du contenu du message et de la clé privée de signature (K_S).
  - Elle pourra être vérifiée en tout temps, par n'importe qui, à l'aide de la clé publique de vérification (K_V).
- Certains schémas supportent la non-répudiation.
- La signature sert à authentifier l'identité du signataire en assurant que seul lui/elle aurait pu calculer cette signature.
- Les MAC constituent une forme limitée de signature digitale:
  - Comme la clé qui a servi à calculer le MAC doit rester secrète, le MAc ne pourra être vérifiée par n'importe qui.

Signatures digitales (2)



Signatures jetables

- Pas besoin d'outils mathématiques puissants pour construire un schéma de signature digitale.
- Soit h, une fonction de hachage cryptographique.
- On veut signer un message m de n bits.
- La clé privée est formée de $2n$ valeurs aléatoires $x_{i,0}$, $x_{i,1}$.
- On calcule ensuite la clé publique: $y_{i,0} = h(x_{i,0})$, $y_{i,1} = h(x_{i,1})$.
- La signature du message est la concaténation des s_i, où

$$s_i = \begin{cases} x_{i,0} \text{ si } m_i = 0 \\ x_{i,1} \text{ si } m_i = 1 \end{cases}$$

- Pour vérifier la signature, il suffit de vérifier que:

$$y_{i,0} = h(s_i) \text{ si } m_i = 0,$$
$$y_{i,1} = h(s_i) \text{ si } m_i = 1$$

Signatures de El Gamal

- El Gamal a aussi publié un schéma de signature, qui n'est pas sans rappeler son algorithme de chiffrement...
- Soit p, un entier premier de grande taille et g, un entier de grand ordre modulo p.
- Soit a, la clé (privée) de signature de A et $y_a = g^a \bmod p$, sa clé publique de vérification.
- Pour signer le message $0 \leq m < p$, A génère un nombre aléatoire k, relativement premier avec $p - 1$, calcule $r = g^k \bmod p$ et trouve s tel que $a \cdot r + k \cdot s \equiv m \bmod p - 1$.
- La signature correspond à la paire (r, s).
- Pour vérifier la signature, le vérificateur utilise la clé publique de A y_a et vérifie si $y_a^r \cdot r^s \equiv g^m \bmod p$.
  - En effet, $y_a^r \cdot r^s = g^{ar+ks} \equiv g^m \bmod p$

DSA

- Le *standard de signature digitale* (DSA), présenté dans FIPS 186 est fortement inspiré de El Gamal.
- Pour générer une nouvelle clé DSA, il faut suivre les étapes suivantes:
  - Soit q, un nombre premier tel que $2^{159} < q < 2^{160}$ (entier de 160 bits significatifs).
  - Choisir un entier $0 \leq t \leq 8$ et un premier p tel que q divise p - 1 et $2^{511+64t} < p < 2^{512+64t}$ (entier de $512+64t$ bits significatifs).
  - Choisir un entier $1 < \alpha < p - 1$ et calculer $g = \alpha^{(p-1)/q}$. Si le résultat donne 1, recommencer avec un autre α.
  - Choisir la clé privée $1 \leq a \leq q - 1$.
  - Calculer $y = g^a \bmod p$.
  - La clé publique sera formée de (p, q, g, y).

DSA (2)

- Pour calculer la signature d'un message m:
  - On commence par hacher le message avec SHA-1 et on convertit le résultat en un entier de 160 bits.
  - Choisir aléatoirement $1 \leq k \leq q - 1$ et calculer $r = g^k \bmod p$.
  - Calculer $k^{-1} \bmod q$, l'inverse de k.
  - Calculer $s = k^{-1}(h(m) + ar) \bmod q$. La signature correspond à (r, s).
- Pour vérifier une signature (r, s) sur un message m:
  - Vérifier que $1 \leq r < q$ et que $1 \leq s < q$.
  - Calculer $w = s^{-1} \bmod q$.
  - Calculer $u_1 = w \cdot h(m) \bmod q$ et $u_2 = r \cdot w \bmod q$.
  - Calculer $v = (g^{u1} y^{u2} \bmod p) \bmod q$.
  - La signature est valide ssi $v = r$.

Signatures RSA

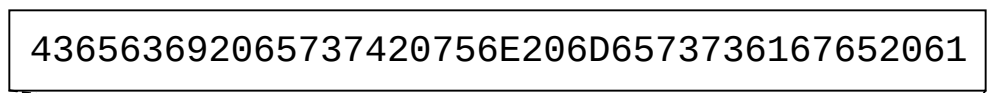
- L'algorithme de chiffrement RSA peut être transformé en un schéma de signature.
  - L'opération de signature sera la même que pour le déchiffrement. De même pour la vérification et le chiffrement.
- Donc, la signature d'un message $0 \leq m < n$ correspond à la formule $s = m^d \bmod n$.
- Pour vérifier la signature, on calcule $t = s^e \bmod n$ et on vérifie que $t = m$.
  - Ceci nous permet de recouvrir le message en même temps.
- Ce schéma de signature nous limite à des messages de taille inférieure à celle de la clé.
  - Aujourd'hui, une clé RSA typique utilise 1024 bits.
  - C'est plutôt court pour un message.

Signatures RSA (2)

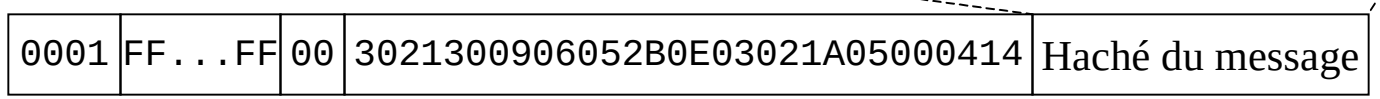
- Comment faire pour signer des messages plus longs?
  - En hachant le message avec une fonction de hachage cryptographique (ex: SHA-1)
  - En construisant un bloc de signature selon une technique d'encodage normalisée (ex: PKCS #1)

Message: “Ceci est un message à signer”

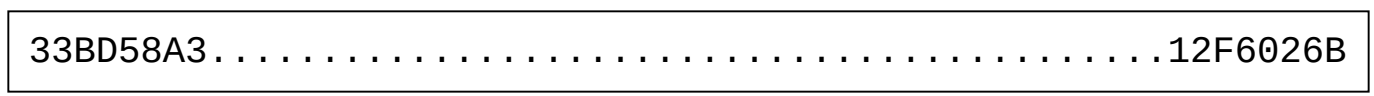
SHA-1 du message:
(20 octets)



PKCS #1:
(128 octets)



Signature:
(128 octets)



PKCS

- PKCS est une suite de recommandations publiés par RSA.
- Entre autres PKCS #1 indique comment utiliser RSA pour chiffrer, déchiffrer, signer et vérifier.
- Exemple: Signature RSA, utilisant SHA-1
  - On commence par hacher le message *m*, pour obtenir *h*(*m*), d'une longueur de 160 bits.
  - On construit le bloc à signer de la façon suivante:

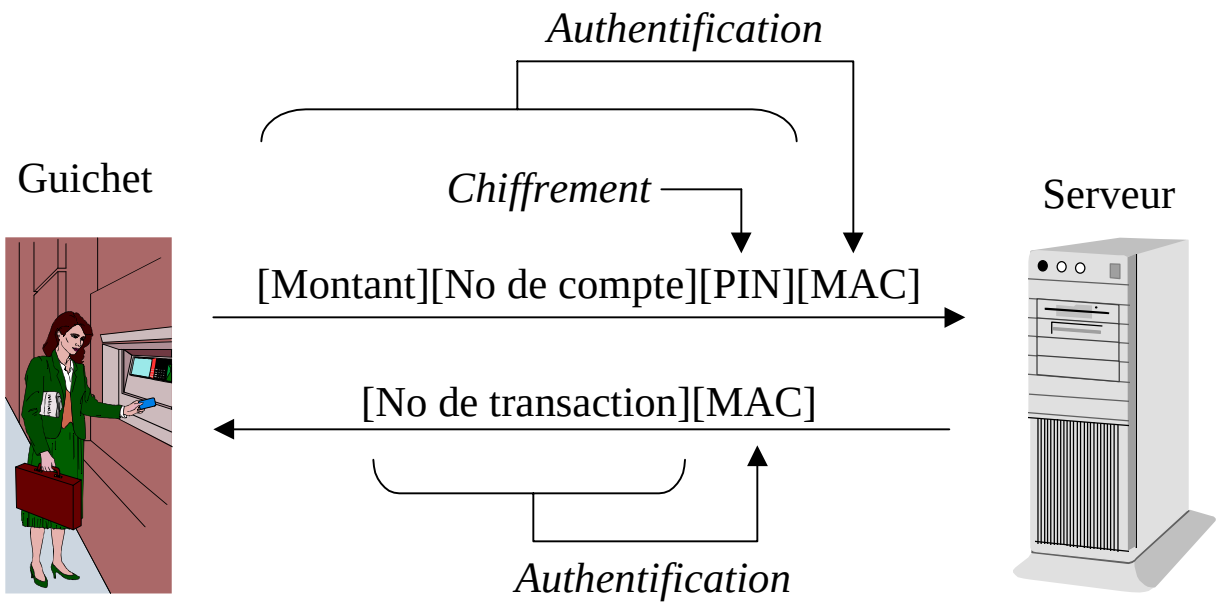
    01 || CR || 00 || FH || h(m)

    où CR est une chaîne de rembourrage (suite de 0xFF) et FH est une séquence identifiant la fonction de hachage (SHA-1).
  - La longueur du bloc doit égaler celle de la clé (1024 bits).

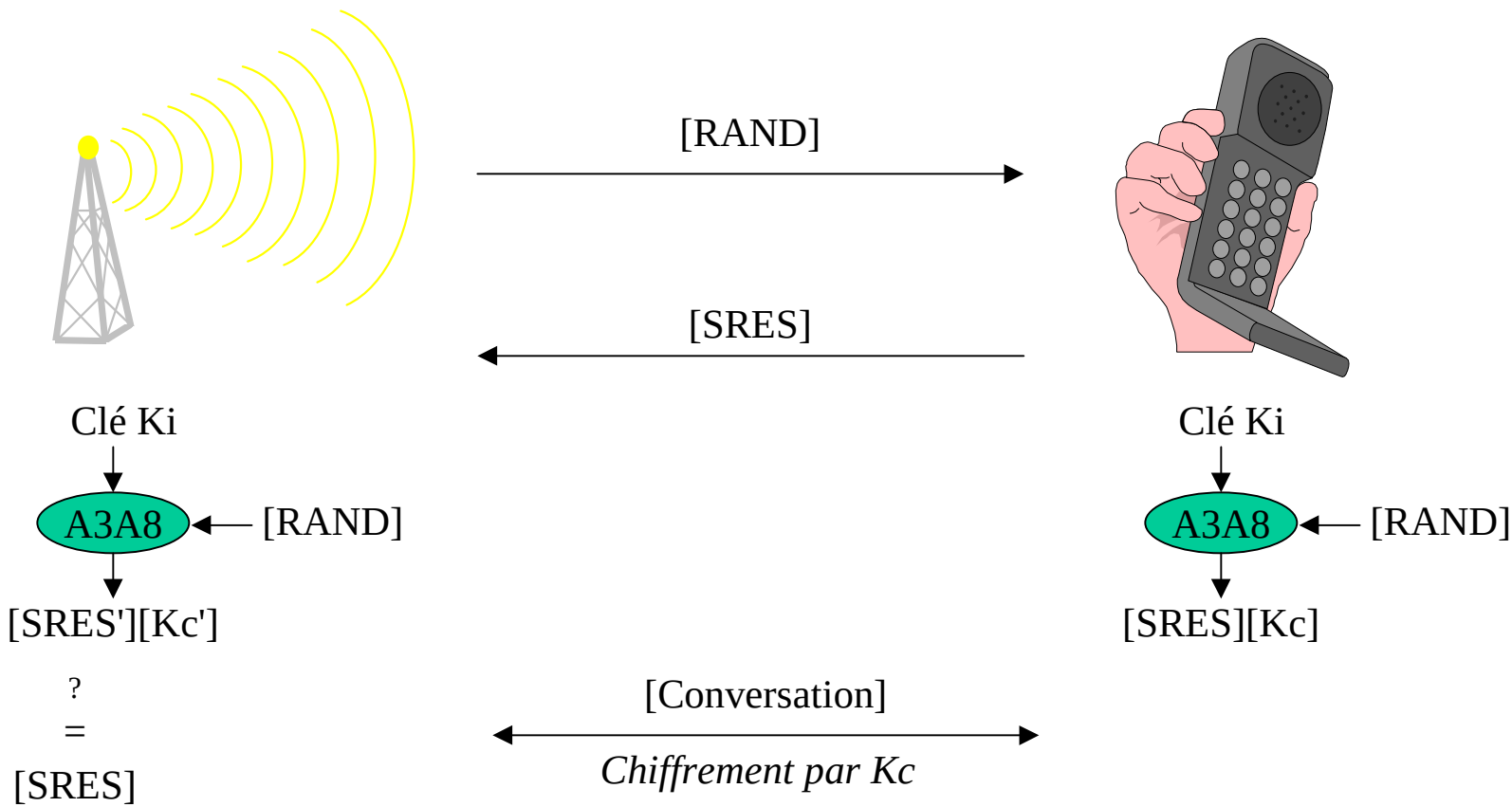
PKCS (2)

- #1: RSA Cryptography Standard
- #3: Diffie-Hellman Key-Agreement Standard
- #5: Password-Based Cryptography Standard
- #6: Extended-Certificate Syntax Standard
- #7: Cryptographic Message Syntax Standard
- #8: Private-Key Information Syntax Standard
- #9: Selected Object Classes and Attribute Types
- #10: Certification Request Syntax Standard
- #11: Cryptographic Token Interface Standard
- #12: Personal Information Exchange Syntax
- #15: Cryptographic Token Information Syntax Standard

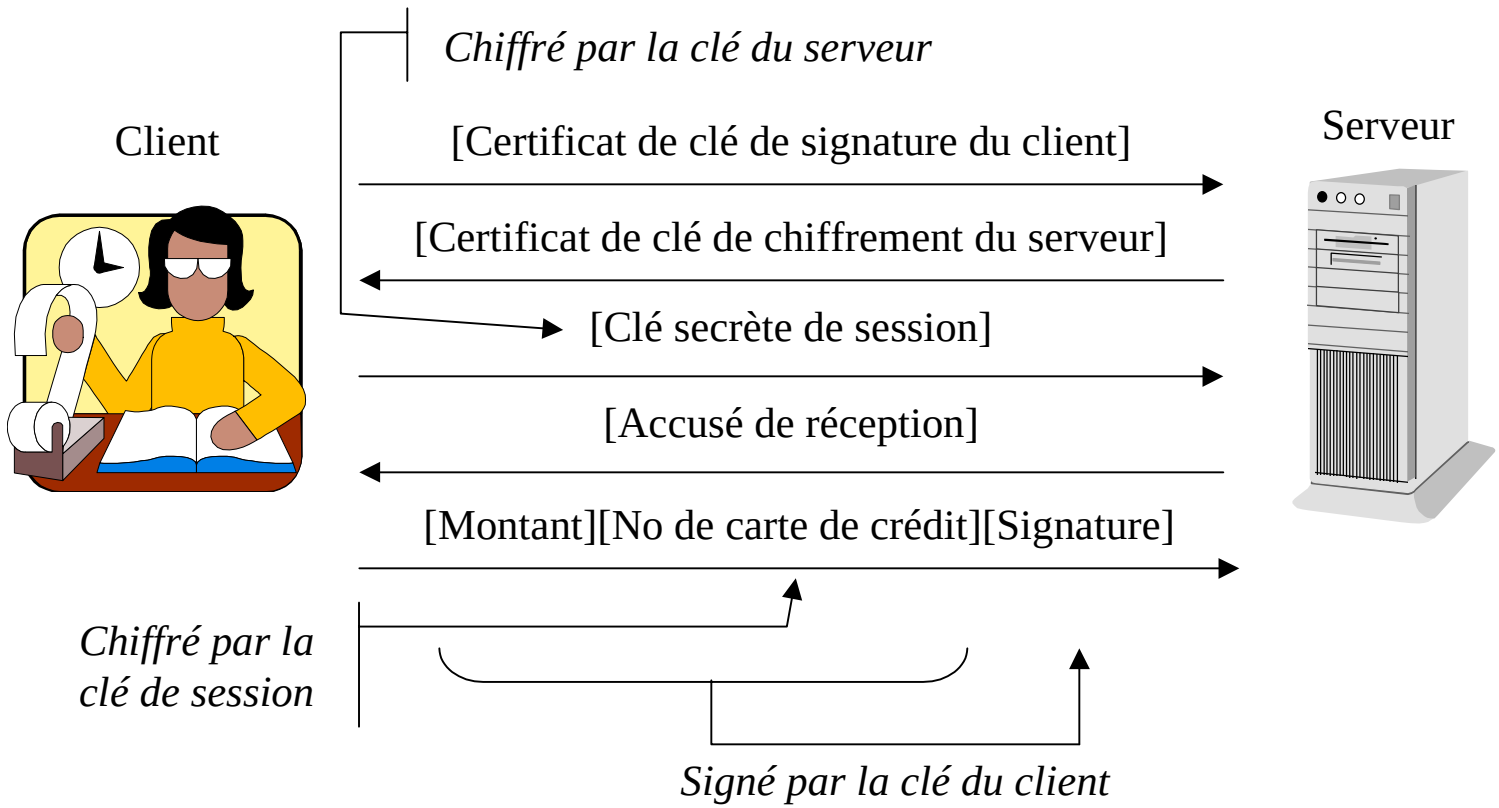
Exemple: Guichet automatique



Exemple: GSM


[RAND]
[SRES]
Clé Ki
A3A8
[RAND]
[SRES'][Kc']
?
=
[SRES]
Clé Ki
A3A8
[RAND]
[SRES][Kc]
[Conversation]
Chiffrement par Kc

Exemple: Achat électronique



Échange de clés

- Deux participants qui ne partagent aucune donnée secrète peuvent arriver à partager un secret, sans se rencontrer...
- On parle généralement de *protocole d'échange de clés*, puisque cela implique en général plusieurs étapes.
- Plusieurs protocoles existants:
  - Diffie-Hellman
  - Needham-Schroeder
  - Enveloppes à clé publique

Diffie-Hellman

- Conçu Diffie et Hellman en 1976.
- Avant de commencer, Alice et Bob choisissent un nombre premier p et un générateur g mod p, qui peuvent être publics.

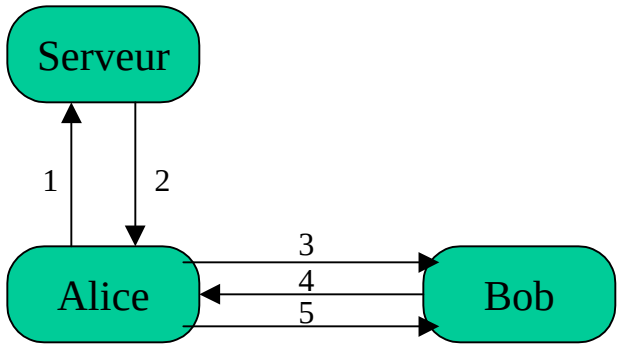
• Alice génère aléatoirement a entre 1 et p - 1.		• Bob génère aléatoirement b entre 1 et p - 1.
• Alice calcule $A = g^a \bmod p$.		• Bob calcule $B = g^b \bmod p$.
• Alice envoie A à Bob.	A →	
	← B	• Bob envoie B à Alice.
• Alice calcule $K = B^a \bmod p$.		• Bob calcule $K = A^b \bmod p$.

- Alice et Bob partage maintenant la même clé secrète K, car $B^a = A^b = g^{ab} \bmod p$.
- La sécurité repose sur la difficulté de calculer a et b, c'est-à-dire de calculer le logarithme discret mod p.

Needham-Schroeder

- Conçu par Needham et Schroeder en 1978.
- Situation particulière où un serveur central est disponible pour établir des clés de session.
- Utilisé pour les algorithmes de chiffrement symétriques.
- Quelques conventions:
  - A = une représentation de l'identité d'Alice
  - B = une représentation de l'identité de Bob
  - K_{as} = une clé secrète partagée entre Alice et le serveur
  - K_{bs} = une clé secrète partagée entre Bob et le serveur
  - K_{ab} = la clé secrète que veulent partager Alice et Bob
  - N_a, N_b = nonces (défis cryptographiques) pour éviter les attaques par répétition.

Needham-Schroeder (2)



- 1. Alice - Serveur: A, B, N_a
- 2. Serveur - Alice: $\mathrm{E}[K_{as}](N_a \parallel B \parallel K_{ab} \parallel \mathrm{E}[K_{bs}](K_{ab}, A))$
- 3. Alice - Bob: $\mathrm{E}[K_{bs}](K_{ab}, A)$
- 4. Bob - Alice: $\mathrm{E}[K_{ab}](\mathrm{N}_b)$
- 5. Alice - Bob: $\mathrm{E}[K_{ab}](\mathrm{N}_b\text{-}1)$

Enveloppes à clé publique

- Utilisation d'un algorithme de chiffrement à clé publique pour la transmission d'une clé secrète.
- Exemple: une clé DES dans une "enveloppe" RSA
  - La clé DES = 0xDE 0xAD 0xBE 0xEF 0xCA 0xFE 0xCA 0xFE
  - On transforme la clé DES en un grand entier de 512 bits. On aura plusieurs possibilités:
    - 0x00...00DEADBEEFCAFECAFE ajusté avec des 0 non significatifs.
    - Format PKCS #1 (qui ressemble beaucoup à celui d'une signature)
- Il ne faut pas que la taille de la clé secrète que celle de la clé publique.

Certificats

- Lorsqu'on utilise une clé, il n'y a rien qui nous assure de l'identité de son détenteur.
  - Dans le cas des clés secrètes, on peut se fier à un serveur, comme dans le cas du protocole Needham-Schroeder.
  - Dans le cas des clés publiques, nous avons donc besoin de lier la clé publique avec son détenteur.
- Un *certificat* sert à lier une identité ou un rôle avec une clé publique.
- C'est une *autorité de certification* (en anglais CA) qui sera chargée d'émettre des certificats.
- Le format de certificat le plus répandu est celui décrit dans la norme X.509 v3.

Certificats (2)

- Que trouve-t-on dans un certificat?
  - L'identité du détenteur
  - La période de validité de la clé (activation et expiration)
  - La clé publique (évidemment!)
  - L'algorithme de la clé publique
  - Autres attributs
- Le certificat structure ces informations selon les règles d'encodage de données décrites dans la norme ASN.1.
- Comment assurer l'intégrité des certificats?
  - Chaque CA possède une clé de signature digitale.
  - Elle signe chaque certificat avec sa clé privée.
  - Elle diffuse un certificat qui contient sa clé publique.

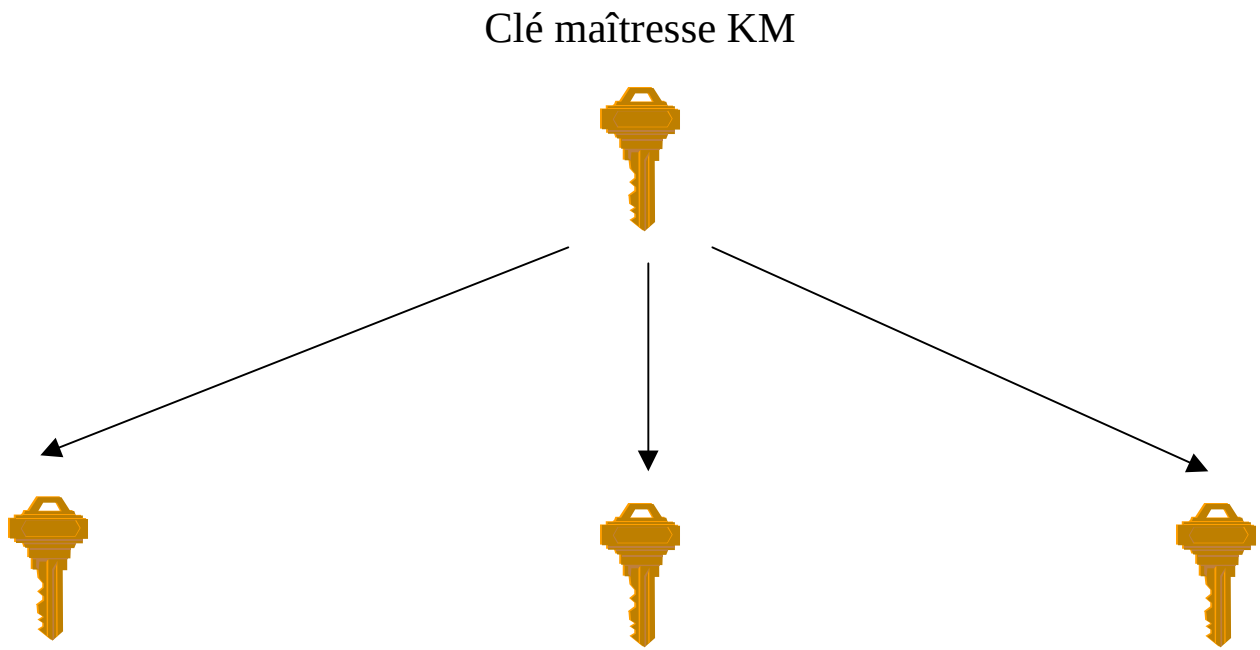
Certificats (3)

- Qui signe le certificat d'une CA?
- On a 2 possibilités:
  - La CA peut signer son propre certificat (auto-certification). On parlera alors de CA racine.
  - La CA fait signer son certificat par une autre CA, déplaçant la confiance vers cet autre CA.
- Pour vérifier un certificat, il faut obtenir la chaîne complète de certificats jusqu'à la racine.
- Au lieu de spécifier une identité dans le certificat, on pourrait y inclure des droits d'accès, implémentant donc le concept de Capacité.

Hiérarchie de clé secrète

- Considérons la situation suivante:
  - Un serveur bancaire centralisé
  - Un grand nombre de terminaux indépendants (>1000)
  - Communication entre un terminal et le serveur doit être sécurisée.
- Si chaque terminal contient une clé secrète, le serveur pourrait avoir à gérer 1000 clés indépendantes.
  - Si ces clés sont contenus dans un appareil cryptographiques, il faudra compter environ 8K, si ce sont des clés DES.
- Une solution plus efficace = hiérarchie de diversification
- À la racine, on a une *clé mère*.
- On assigne à chaque terminal un numéro de série unique et on utilise ce numéro et la clé maîtresse pour calculer une *clé fille*.

Hiérarchie de clés secrètes (2)



série:
11111111
Clé fille:
DES[KM]("1111111")

série:
22222222
Clé fille:
DES[KM]("22222222")

série:
33333333
Clé fille:
DES[KM]("33333333")

Normes en cryptographie

- À quoi ça sert?
  - Assurer une compatibilité des formats de données et des propriétés de clés
  - Décourager l'arrivée de solutions "propriétaires", incompatibles avec tout ce qui existe (ex: Microsoft).
- Normes en cryptographie
  - Public Key Cryptographic Standards (RSA Data Security Inc.)
    - Numéroté de 1 à 15, couvre beaucoup d'aspects cryptographiques
  - X.509 (CCITT)
    - De cette norme, on ne conserve que la définition des certificats.
  - PKIX: Public Key Infrastructure (X.509)
    - Groupe de travail de l'IETF (Internet Engineering Task Force)
  - ANSI: X9, X39.1
    - Utilisation des algorithmes par les agences gouvernementales US

Sécurité des mécanismes

- Lorsqu'un nouveau mécanisme cryptographique est introduit, les auteurs doivent prouver la sécurité de ce mécanisme.
- Plusieurs approches sont possibles:
  - Sécurité empirique
    - L'algorithme est soumis à des tests de tous genres, comme on le ferait pour un prototype de voiture.
    - Pas de preuve formelle que le mécanisme est vraiment sécuritaire.
    - Un tel mécanisme est vulnérable à un nouveau type d'attaque.
    - Par exemple, DES n'a pas de preuve formelle complète de sa sécurité. Certaines de ses propriétés ont été prouvés, mais pas toutes.

Sécurité des mécanismes (2)

- Approches (suite)
  - Sécurité prouvable
    - On définit un modèle d'utilisation du mécanisme (typiquement des limites sur la puissance de calcul de l'attaquant).
    - On démontre que la sécurité du mécanisme est directement liée à la difficulté de résoudre un problème.
    - Il faut s'assurer que l'utilisation réelle d'un mécanisme correspond aux hypothèses qui ont été faites dans la preuve.
    - Ex: El Gamal et le logarithme discret
  - Sécurité inconditionnelle
    - Comme la sécurité prouvable, mais la preuve ne dépend pas de la puissance de calcul de l'attaquant.
    - La preuve se base souvent sur la théorie de l'information.
    - Même remarque sur les hypothèses.
    - Ex: Masque jetable (one-time pad).