Sécurité Informatique

Cryptage à clé publique, Hachage, MAC, Signature Numérique

October 17, 2018

Houcemeddine HERMASSI

houcemeddine.hermassi@enit.rnu.tn

École Nationale d'Ingénieurs de Carthage ENI-CARTHAGE
Université Carthage
Tunisie

Plan de cour

Cryptographie asymètrique
Principe

Problèmes de la cryptographie symètrique

- la cryptographie symétrique utilise une seule clé pour le cryptage/décryptage
- cette clé est partagée par l'émetteur et le récepteur
- 2 clés ou cryptosystème à clé publique (asymétrique)
- Si cette clé est divulguée, toute la communication est compromise
- Ne protège pas l'émetteur du récepteur qui peut modifier un message et prétend l'avoir recu de l'émetteur

Principes de la cryptographie asymétrique

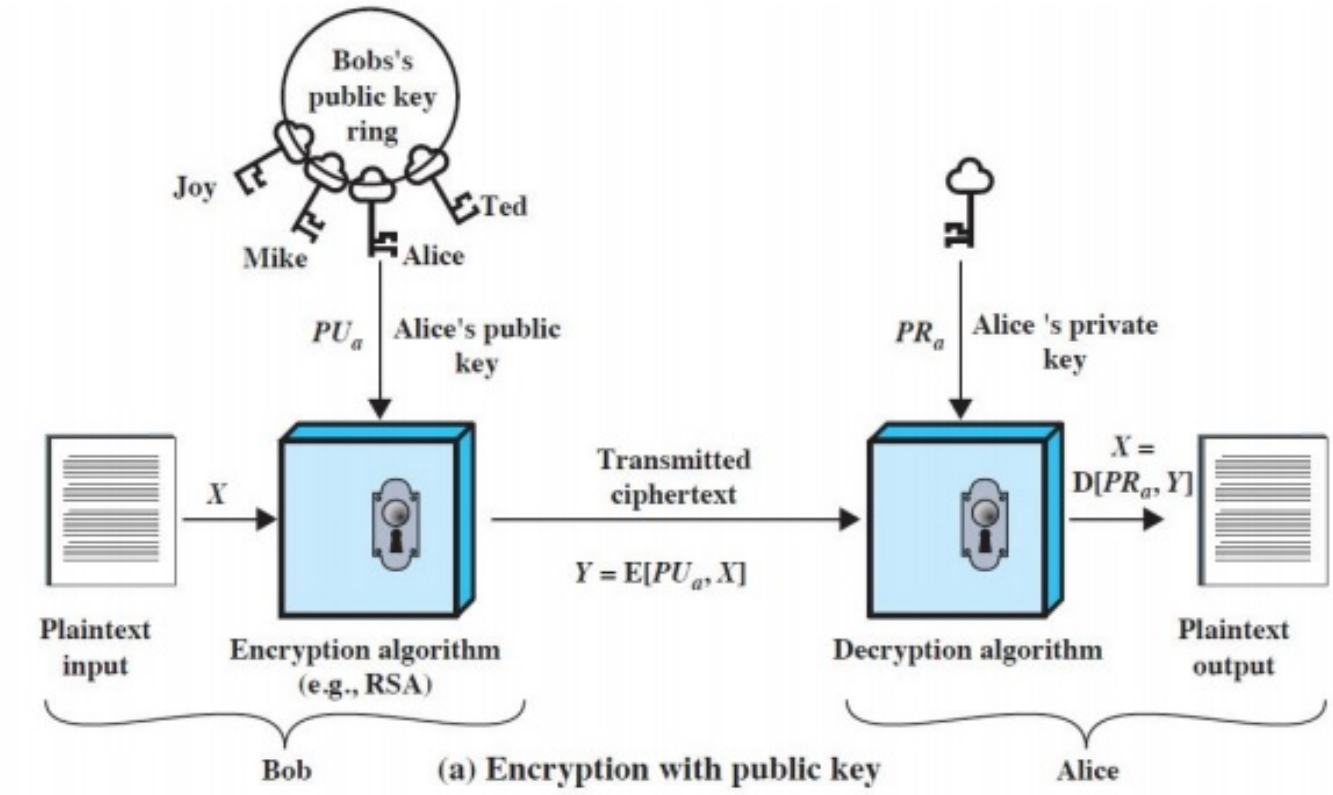
- La conception de la cryptographie asymétrique vient du besoin de résoudre deux grands problèmes :
  - **Distribution des clés** : Comment faire une communication sécurisé sans passer par un KDC (Key distribution Center)
  - **Signature numérique** : Comment vérifier que le message est recu intact depuis l'émetteur légitime
- Whitfield Diffie et Martin Hellman from Stanford University ont proposé en 1976 une approche qui peut résoudre les deux problèmes

Composants de la cryptographie asymétrique

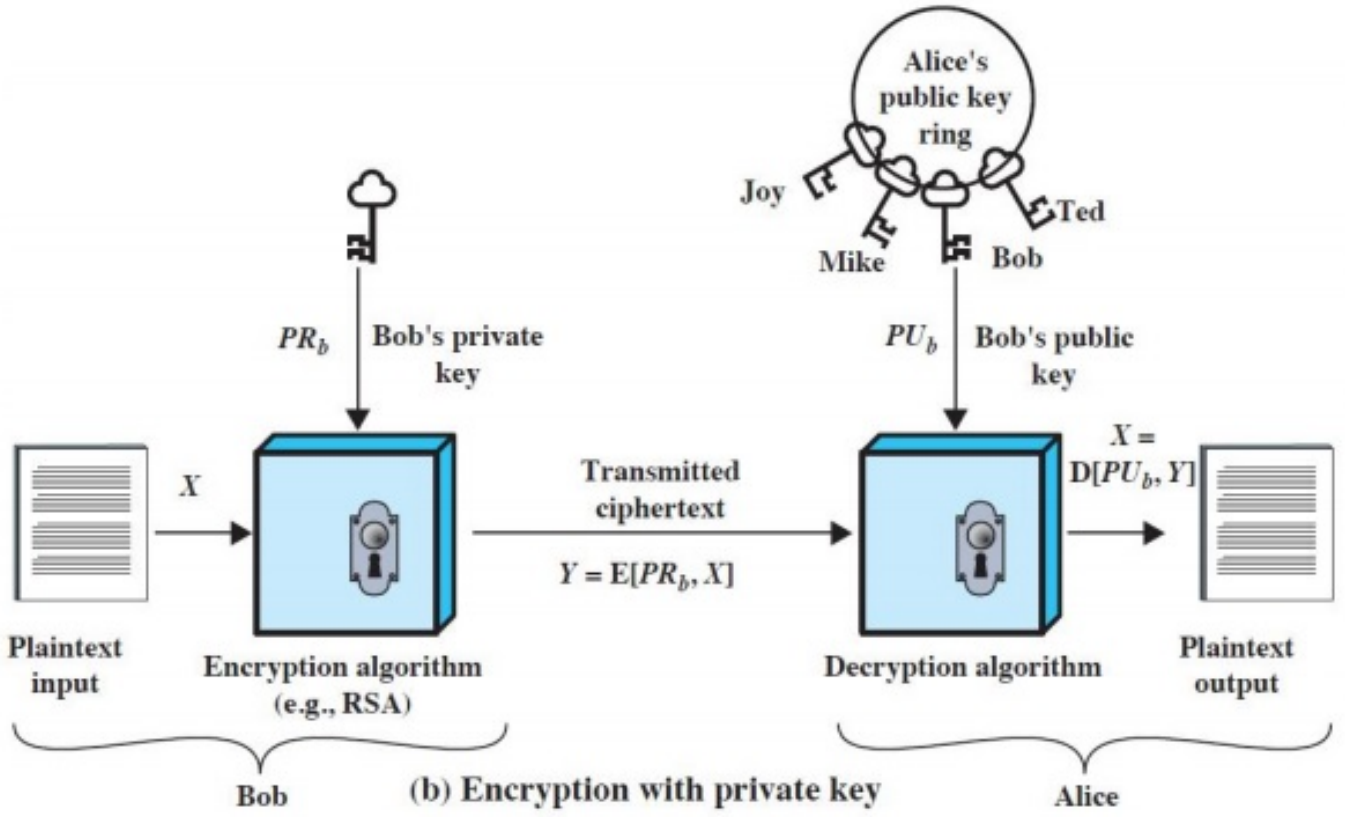
- **plaintext** : texte clair
- **ciphertext** : texte chiffré
- **algorithme de cryptage** : opérations faites sur le plaintext
- **algorithme de décryptage** : opérations faites sur le ciphertext
- **clé publique** : utilisé par l'algorithme de cryptage (si confidentialité)
- **clé privé** : utilisé par l'alg de décryptage (coté récepteur)

Cryptographie asymètrique
Principe

Cryptage avec clé publique

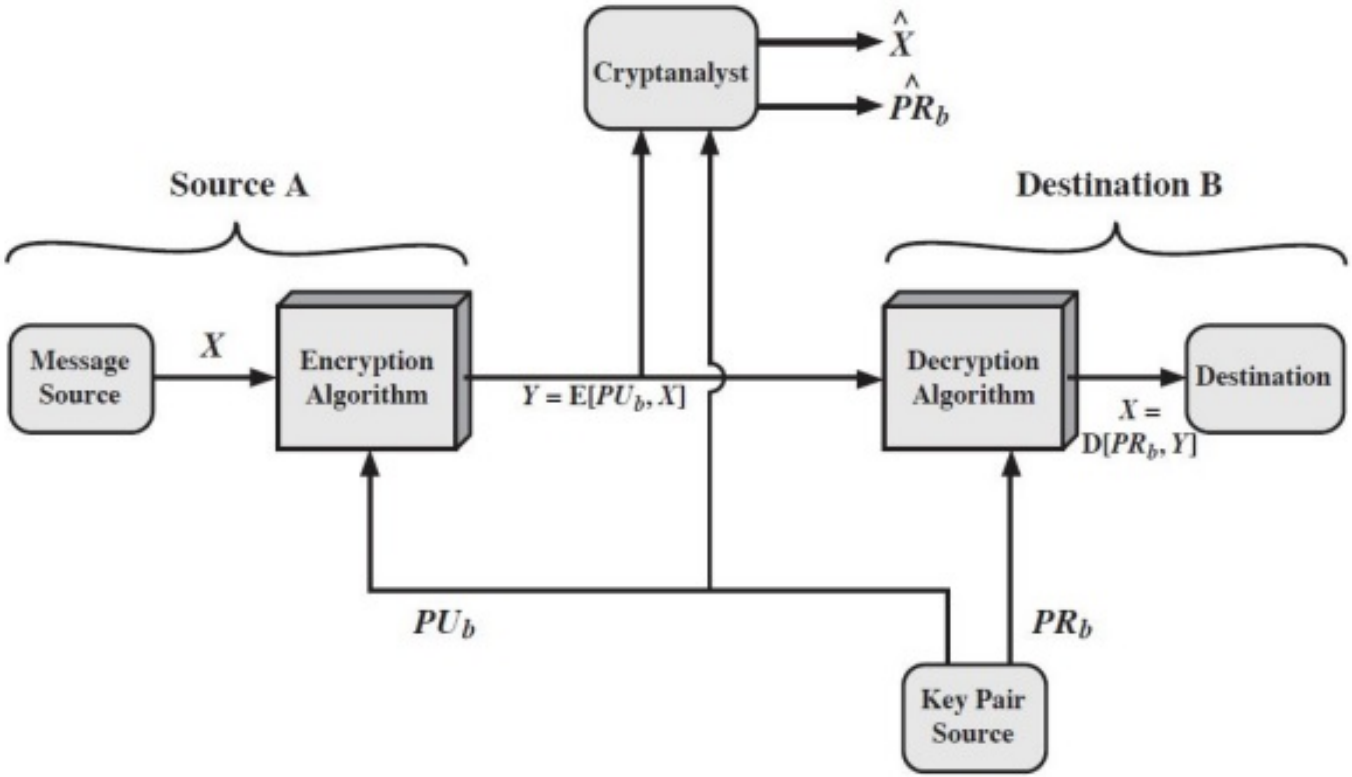


Décryptage avec clé privée

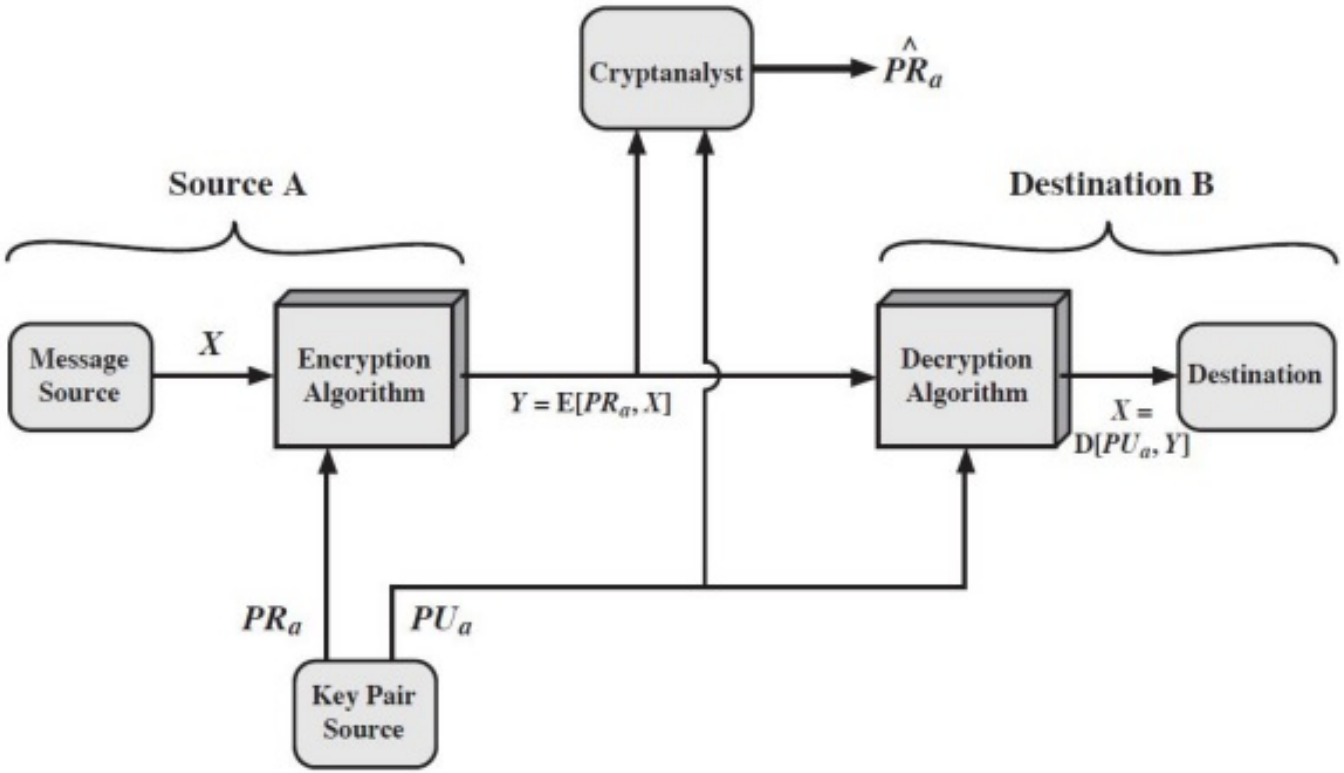


(b) Encryption with private key

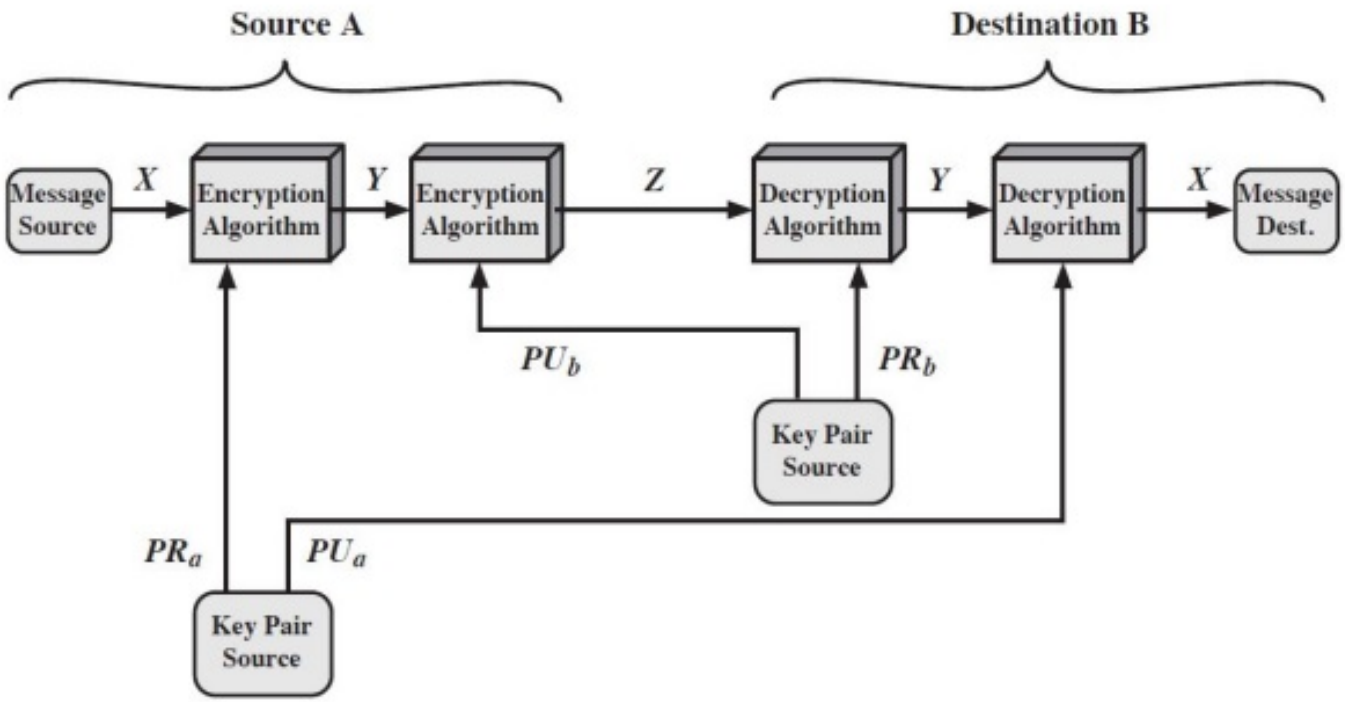
Cryptographie asymètrique: confidentialité



Cryptographie asymètrique: authentification



Cryptographie asymètrique: confidentialité & authentification



Applications de la cryptographie asymétrique

- Les algorithmes à clés publiques sont utilisés dans trois applications :
  - **Cryptage/décryptage** : L'émetteur chiffre un plaintext par la clé publique du récepteur
  - **Signature numérique** : L'émetteur signe un message par sa clé privée
  - **Partage des clés** : émetteur et récepteur coopèrent pour partager une clé de session
- il y a des algorithmes qui sont appropriés pour les trois applications, et d'autres ne sont appropriés que pour une ou deux applications parmi les trois

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Exigences de la cryptographie asymétrique

Ces algorithmes doivent avoir les exigences suivantes

- besoin d'une fonction trappe à sens unique
- une fonction à sens unique vérifie le suivant :
  - $Y = f(X)$ est **facile**
  - $X = f^{-1}(Y)$ est **non-faisable**
- une fonction trappe à sens unique est une famille f_k de fonctions inversibles vérifiant :
  - $Y = f_k(X)$ est facile si **k** et **X** sont connus
  - $X = f_k^{-1}(Y)$ est facile si **k** et **Y** sont connus
  - $X = f_k^{-1}(Y)$ est non-faisable si **Y** est connu et **k** non connu
- un algorithme à clé publique est basé donc sur une fonction trappe à sens unique

Cryptographie asymètrique
RSA

Principe

- Developpé en 1977 au MIT par Ron Rivest, Adi Shamir & Len Adleman
- le plus utiisé des algorithme à clé publique
- c'est un alg dont le plaintext et le ciphertext sont des entiers entre 0 et n-1
- n est un nombre de taille 1024 bits ou 309 digit décimal

Algorithme RSA

- le plaintext est crypté en blocs, chaque bloc a une valeur inférieure à **n**
- Cryptage d'un bloc de plaintext est comme suit :

$$C = M^e \, mod(n)$$

- décryptage est comme suit :

$$M = C^d \, mod(n) = (M^e)^d \, mod(n) = M^{ed} \, mod(n)$$

- émetteur et récepteur connaissent la valaur de **n**
- L'émetteur connait la valeur de **e**
- seulement le récepteur connaît la valeur de **d**
- la clé publique est la paire **(e, n)**
- la clé privée est la paire **(d, n)**

Génération des clés

- chaque utilisateur génère ses propres clés (privée et publique) par :
- selectionner deux grands nombres premiers **p** et **q**
- calculer $n = p \times q$
- calculer $\phi(n) = (p-1) \times (q-1)$
- sélectionner aléatoirement un nombre e avec : $1 < e < \phi(n)$ et $GCD(e, \phi(n)) = 1$
- résoudre cette équation pour trouver d avec $0 \leq d \leq n$:

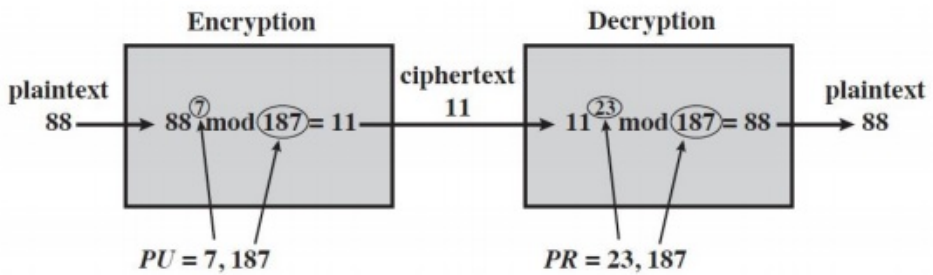
$$e \times d = 1 mod \phi(n)$$

- publier la paire $PU = \{e, n\}$ comme clé publique
- garder en secret la paire $PR = \{d, n\}$ comme clé privée

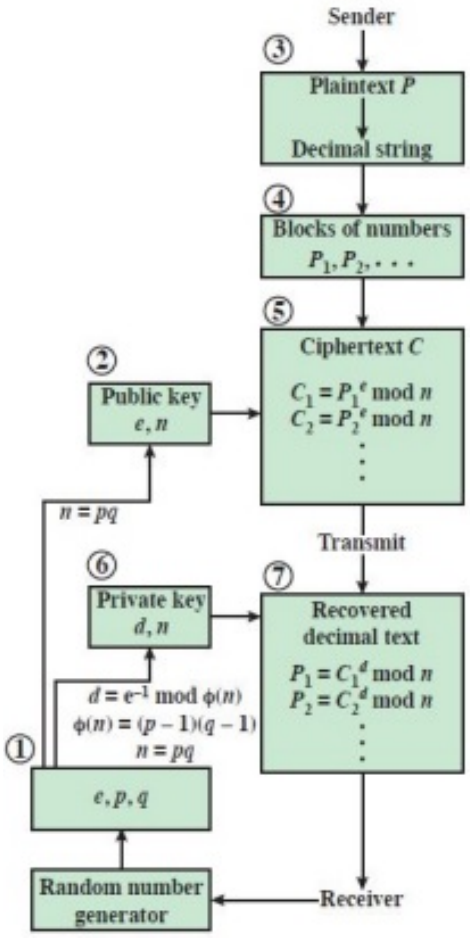
Génération des clés: exemple

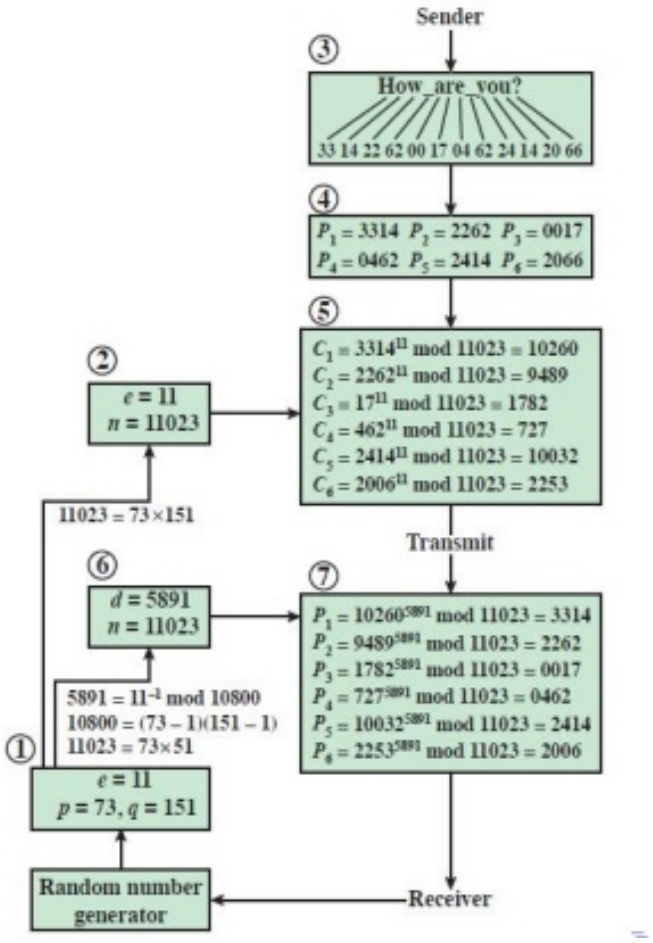
- Sélectionner les nb premiers : $p = 17$ & $q = 11$
- Calculer $n = p \times q = 17 \times 11 = 187$
- Calculer $\phi(n) = (p-1) \times (q-1) = 16 \times 10 = 160$
- Sélectionner e : $gcd(e, 160) = 1$; choisir $e = 7$
- Déterminer d tel que $d \times e = 1 mod 160$ et $d < 160$: la valeur est $d = 23$ puisque $23 \times 7 = 161 = 10 \times 160 + 1$
- Publier la clé publique $PU = \{7, 187\}$
- Garder en secret la clé privée $PR = \{23, 187\}$

Exemple RSA



Exemple RSA avec un long message





Pourquoi RSA fonctionne

- A cause du théorème d'Euler : $a^{\phi(n)} mod n = 1$ avec $gcd(a, n) = 1$
- Dans RSA on a :
  - $n = p \times q$
  - $\phi(n) = (p-1) \times (q-1)$
  - choisir prudemment e & d pour etre des inverses $mod \phi(n)$
  - donc $e \times d = 1 + k \times \phi(n)$ pour k donnée
- donc :

$$C^d = M^{e \times d} = M^{1+k \times \phi(n)} = M^1 \times (M^{\phi(n)})^k = M^1 \times (1)^k = M^1 = M mod(n)$$

Exponentiation dans RSA

- le cryptage et le decryptage manipule des exponentiations de grand nombres modulo n
- on peut utiliser des propriétés de l'arithmetique modulaire :

$$(amod(n)) \times (bmod(n)) = (a \times b)mod(n)$$

- il faut chercher aussi à faire l'exponentiation le plus vite possible
- on prend rendre l'exponentiation en $O(log_2 n)$ multiplications pour un nombre **n**

ex1 : $7^5 = 7^4 \times 7^1 = 3 \times 7 = 10mod11 \Rightarrow log_2 5 = 3$ multiplications ex2 :
$3^{129} = 3^{128} \times 3^1 = 5 \times 3 = 4mod11 \Rightarrow log_2 129 = 8$ multiplications

Calcul de a^b mod n

```
c ← 0; f ← 1
for i ← k downto 0
    do  c ← 2 × c
        f ← (f × f) mod n
    if  b_i = 1
        then c ← c + 1
             f ← (f × a) mod n
return f
```

avec b est un entier converti en binaire en $b_k b_{k1} \ldots b_0$ Ex de calcul de $a^b mod n$, pour $a = 7$, $b = 560 = (1000110000)_2$, et $n = 561$

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
f	7	49	157	526	160	241	298	166	67	1

Attaque sur RSA

Trois approches pour attaquer RSA :

- Factoriser n en deux nombres premiers p et q. Ceci va mener à trouver $\phi(n) = (p-1)(q-1)$ ce qui mène à déterminer $d = e - 1 mod \phi(n)$
- déterminer $\phi(n)$ directement sans trouver p et q, ce qui mène à déterminer $d = e1 mod \phi(n)$
- Déterminer directement d sans déterminer $\phi(n)$

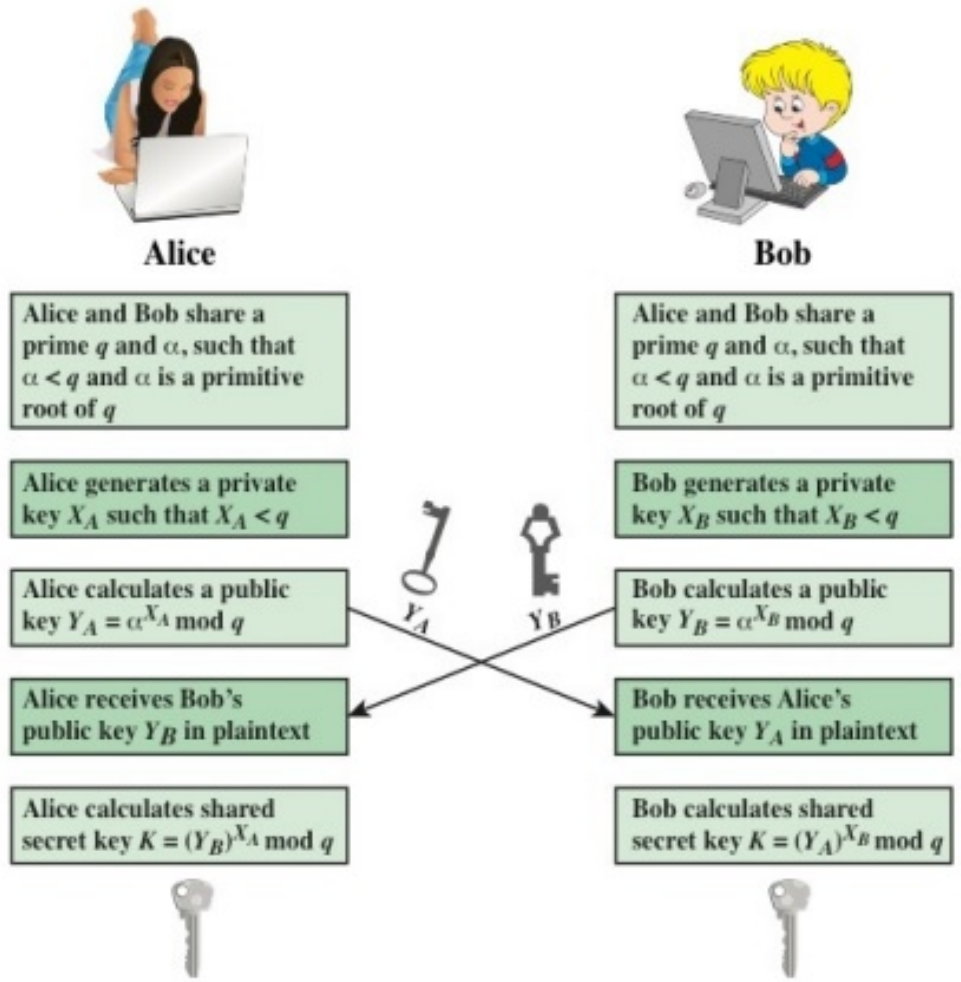
Partage de clé de Diffie-Hellman

Échange de clé de Diffie-Hellman

Principe

- Premier algorithme à clé publique
- un très grand nombre de produits commercials utilisent ce protocole
- But : permettre à deux utilisateurs d'échanger en toute sécurité une clé qui peut par la suite être utilisée pour le chiffrement symétrique de messages
- Son efficacité est liée à la difficulté de calculer des logarithmes discrets

Échange de clé de Diffie-Hellman



Échange de clé de Diffie-Hellman

- La clé partagée entre deux entités A et B, est K_{AB}

$$K_{AB} = a^{x_A \times x_B} \, mod q$$

$$= y_A^{x_B} \, mod q(calculparB)$$

$$= y_B^{x_A} \, mod q(calculparA)$$

- K_{AB} est utilisée comme clé de session dans un alg symétrique entre A et B
- si Alice et Bob continuent à communiquer, ils auront la même clé comme avant, à moins qu'ils ne choisissent de nouvelles clés publiques
- un adversaire doit résoudre le problème du logarithme discret pour compromettre cet algorithme (difficile)

Exemple

- Alice et Bon veulent partager une clé
- Ils partagent un nombre premier q=353 et un nombre a=3
- choisir des nombres aléatoires secrètement : $X_A = 97$, $X_B = 233$
- calculer les clés publiques respectives :

$$y_A = 3^{97} mod 353 = 40(Alice)$$

$$y_B = 3^{233} mod 353 = 248(Bob)$$

- calculer la clé de session partagée K_{AB} :

$$K_{AB} = y_B^{x_A} mod 353 = 248^{97} = 160(Alice)$$

$$K_{AB} = y_A^{x_B} mod 353 = 40^{233} = 160(Bob)$$

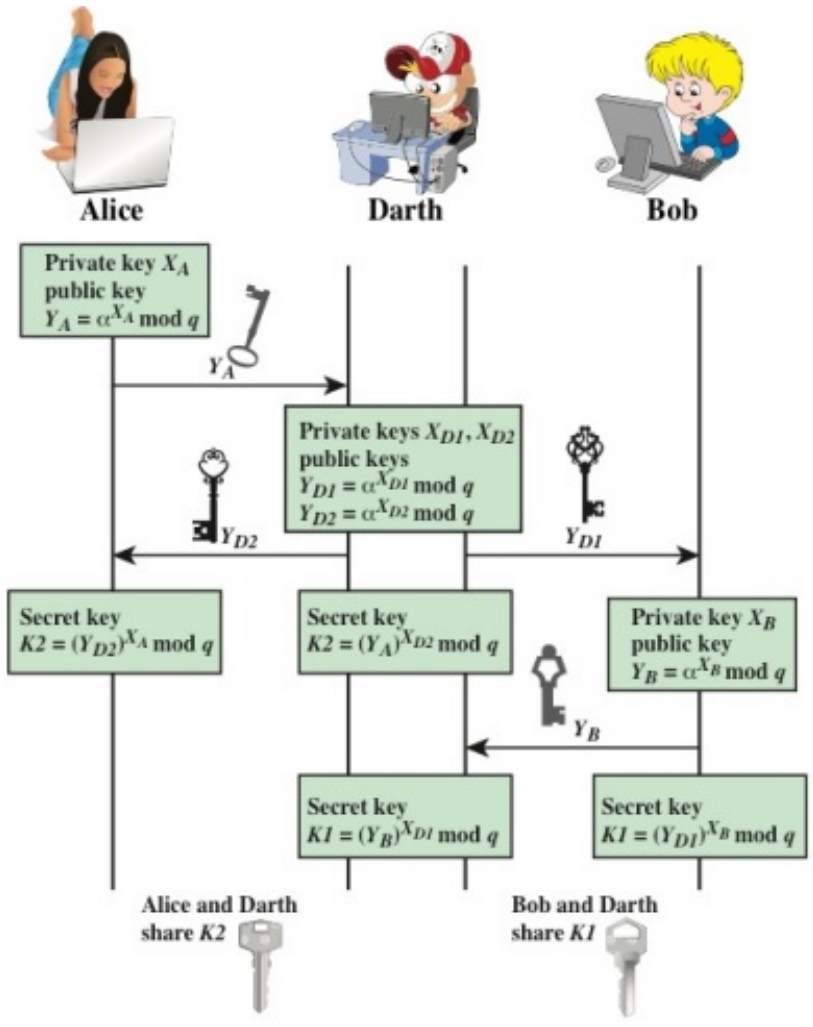
Partage de clé de Diffie-Hellman
Attaque

Man-in-the-Middle Attack

1. Darth se prépare en créant 2 clés privée/publique
2. Alice transmet sa clé publique à Bob
3. Darth intercepte cette clé et transmet sa première clé publique à Bob. Darth calcule alors la clé partagée K_2 avec Alice.
4. Bob recoit la clé publique et calule la clé partagée K_1 (avec Darth au lieu de la faire avec Alice !)
5. Bob transmet sa clé publique à Alice
6. Darth intercepte ce message et transmet sa seconde clé publique à Alice. Darth calcule alors la clé partagée K_1 avec Bob.
7. Alice recoit la clé et calcule la clé partagée K_2 avec Darth (au lieu de Bob)
8. Darth peut alors intercepter, déchiffrer, re-chiffrer, transmettre tous les messages entre Bob et Alice.

Partage de clé de Diffie-Hellman
Attaque

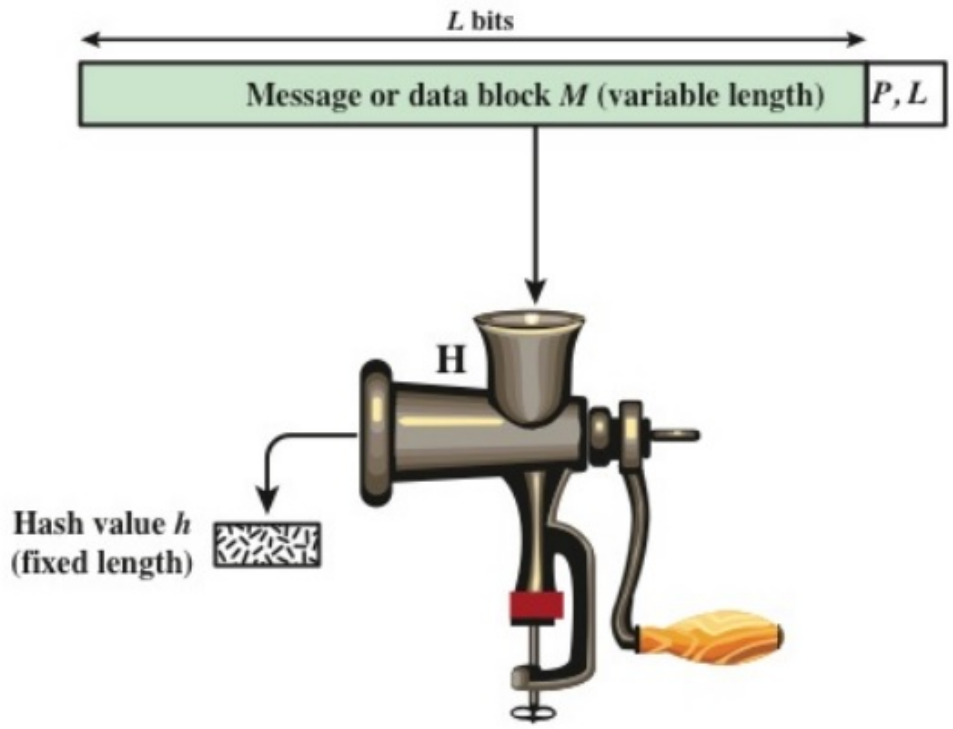
Man-in-the-Middle Attack



Principe

- Une fonction de hachage accepte une entrée de longueur variable et fait sortir un condensé ou empreinte de longueur fixe
- h = H(M)
- Son principal but est la vérification d'intégrité
- Une fonction de hachage cryptographique est un algorithme dont il est mathématiquement difficile de :
  - Trouver une entrée qui donne un condensé bien spécifié (Propriété : fonction à sens unique)
  - trouver deux entrées qui donne le même condensé (Propriété : Sans collision)

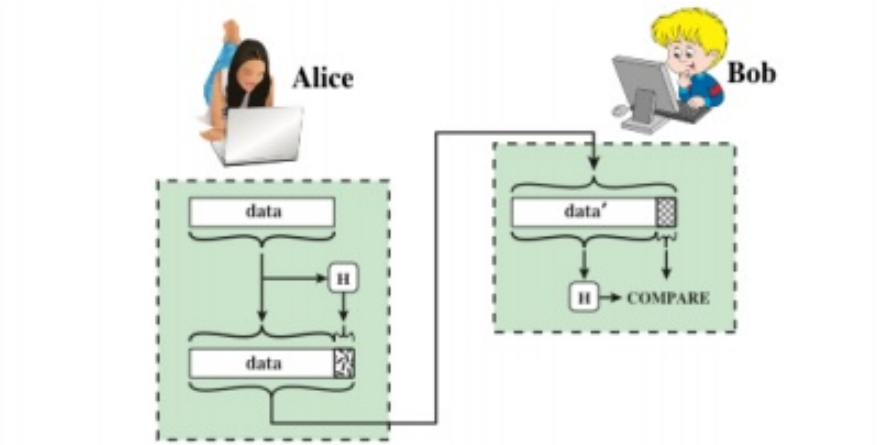
Fonction de hachage cryptographique h=H(M)



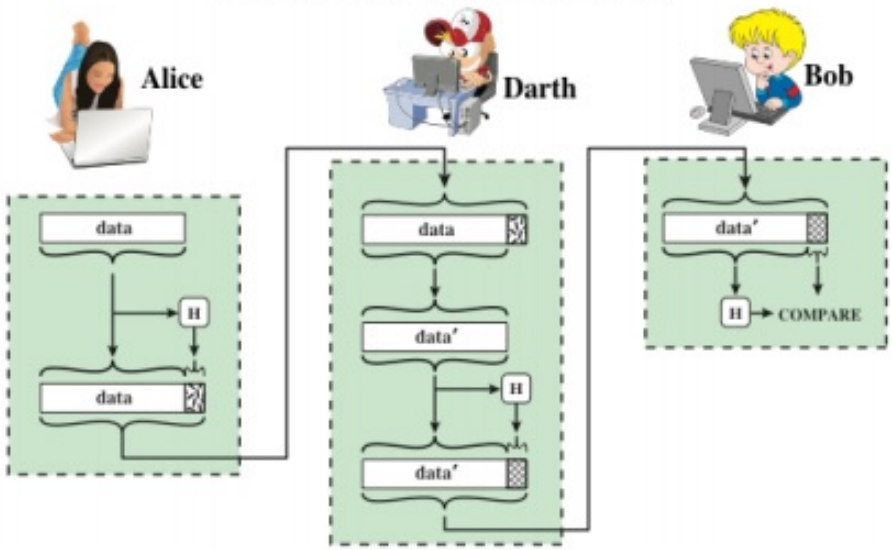
Le condensé du message (digest)

- Le condensé du message est son empreinte digitale
- Beaucoup plus petit que le message original
- facile à calculer
- impossible de retrouver le message depuis le condensé
- Changer le message fait automatiquement changer le condensé

Fonction de hachage et attaque



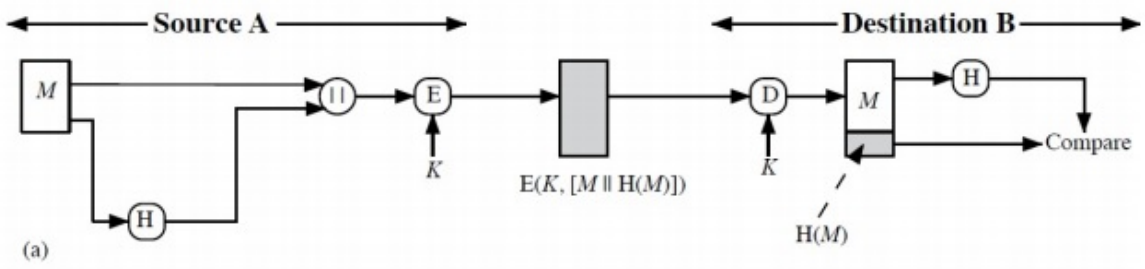
(a) Use of hash function to check data integrity



(b) Man-in-the-middle attack

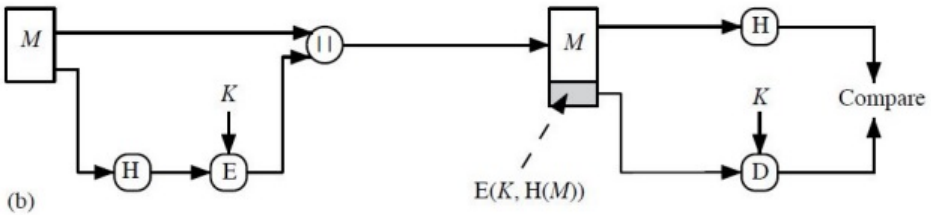
Authentification du message

- Vérifier l'intégrité du message
  - S'assurer que les données recues sont exactement comme envoyés
  - S'assurer que l'identité de l'expéditeur est valide
- **Exemple 1** : Chiffrer le message et son condensé par un cryptosystème symétrique



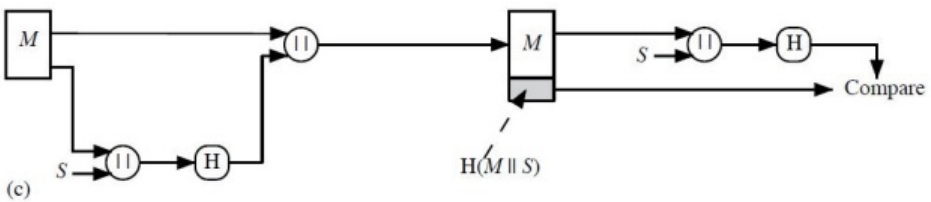
Authentification du message

- **Exemple 2** : Chiffrer seulement le condensé du message
- ca permet de réduire la complexité de calcule si la confidentialité n'est pas sollicitée



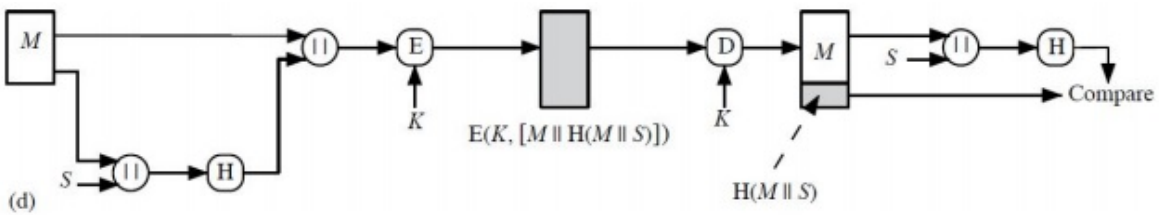
Authentification du message

- **Exemple 3** : Un secret partagé est hachéExemple 3 : Un secret partagé est haché
- Pas de besoin de cryptage



Authentification du message

- **Exemple 4** : Un secret partagé combiné avec confidentialité



D'autres Utilisations des fonctions de hachage

- Utilisée pour créer les fichiers de mots de passe
  - Lorsqu'un utilisateur tape un mot de passe, le condensé du password est comparé au condensé enregistré pour vérification
  - Cette approche est utilisée par la majorité des systèmes d'exploitation
- utilisé pour détecter les intrusions et les virus
  - enregistrer le H(f) de chaque fichier dans le disque
  - l'antivirus peut vérifier par la suite si le fichier a été altéré ou non en recalculant son condensé H(f)
  - Un intrus essayera de changer F sans changer H(f) : très difficile !
- peut être utilisé pour construire des générateurs de séquences pseudo-aléatoires PRNG
  - générer des keystreams, des clés secrètes

Exigences d'une fonction de hachage

- Entrée de longueur variable
- Sortie de longueur fixe
- Efficacité : étant donnée x, il est facile de générer le condensé H(x) en s/w ou h/w
- Fonction à sens unique (Preimage resistant) : Pour un condensé donné h, il est impossible de trouver y tel que H(y) = h
- Pas de collision en sens large (Second preimage resistant : weak collision resistant) : Quelque soit x donnée, il est impossible de trouver $y \neq x$ tel que H(y) = H(x)
- Pas de collision au sens strict (collision resistant : Strong collision resistant) : il est impossible de trouver une paire (x, y) tel que H(x) = H(y)
- Critère aléatoire : La sortie de H doit être aléatoire selon les tests standards (NIST : 16 tests du critère aléatoire)

NB : "impossible" = "mathématiquement ou par calcul difficile"

Exigences des applications d'intégrité

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

Paradoxe d'anniversaire

- Dans une classe, quelle est la probabilité pour que 2 élèves fêtent leurs anniversaires le même jour ?
- Avec 365 jours par an, une trentaine d'élèves dans la classe, on se dit qu'elle doit être faible...
- On va calculer la probabilité pour que, dans un groupe de k personnes, ces personnes aient toutes un jour d'anniversaire différent :
  - Si on a 2 personnes, la première peut avoir son anniversaire n'importe quand, la seconde n'importe quel autre jour. on a donc : $p_2 = \frac{364}{365} = 1 - \frac{1}{365}$
  - si mnt on a k personnes : $p_3 = (1 - \frac{1}{365})(1 - \frac{2}{365})$
  - dans un groupe de k personnes, $p_k = (1 - \frac{1}{365})(1 - \frac{2}{365})...(1 - \frac{k-1}{365})$

Nombre de personnes	Probabilité pour que les anniversaires tombent tous un jour différent
1	1
2	0.99
5	0.97
10	0.88
20	0.58
22	0.52
23	0.49
30	0.29
50	0.03

$\Rightarrow$ Il ne faut donc que 23 personnes pour qu'il y ait plus d'une chance sur 2 (chance > 0.5) pour que 2 personnes aient leur anniversaire le même jour.

Les attaques basés sur le paradoxe d'anniversaire

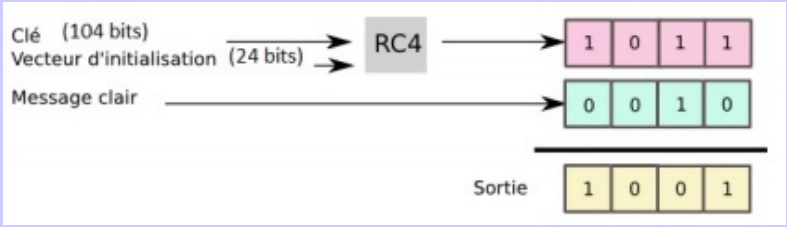
- Dans une attaque qui cherche des collisions, l'adversaire veut trouver 2 messages qui donnent le meme condensé.
- Dans une classe de 23 étudiants la probabilité de trouver 2 étudiants ayant le meme anniversaire est > 0.5
- Si le condensé est codé sur b bits, il y a 2^b empreinte possibles.
- si on prend k msg différents, la probabilité de trouver 2 msg ayant le meme condensé est :

$$p = 1 - (1 - \frac{1}{2^b})(1 - \frac{2}{2^b})..(1 - \frac{k-1}{2^b})$$

- pour que $p \geq \frac{1}{2}$, il suffit que $\overline{p} = 1 - (1 - \frac{1}{2^b})(1 - \frac{2}{2^b})..(1 - \frac{k-1}{2^b}) \leq \frac{1}{2}$
- on a $(1 - \frac{j}{2^b}) \sim e^{-\frac{j}{2^b}}$
- on a alors $(1 - \frac{1}{2^b})(1 - \frac{2}{2^b})..(1 - \frac{k-1}{2^b}) \sim e^{(\frac{-k(k-1)}{2^{b+1}})}$
- Il faut donc que $e^{(\frac{-k(k-1)}{2^{b+1}})} \leq \frac{1}{2}$

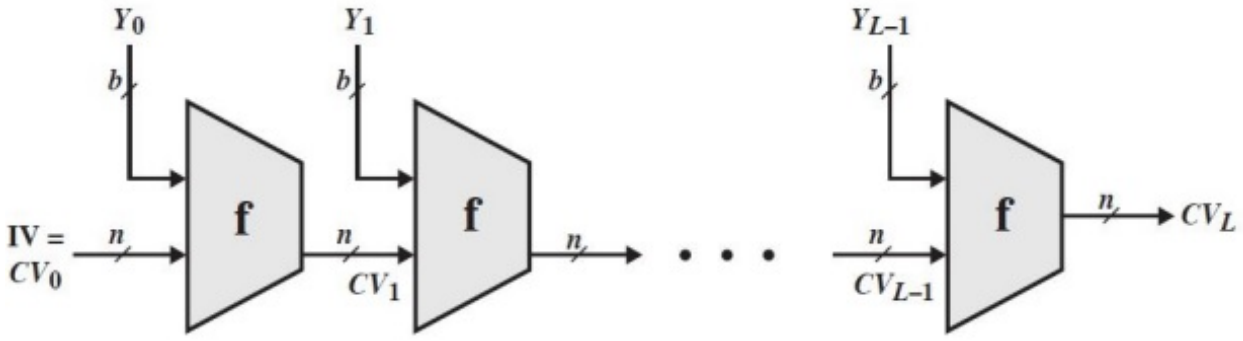
Taille du haché	Nombres de textes à essayer
8	20
16	302
32	77169

Paradoxe d'anniversaire et sécurité wifi (WEP)



- la clé de 104 bits est tapé manuellement (13 caractères). La clé est fixe
- ce qui varie est le vecteur d'initialisation (24 bits) échangé entre le point d'accès et le PC.
- il ya 2^{24} possibilités du IV. presque 16 millions possibilités
- Par le paradoxe des anniversaires, il suffit d'à peu près 4824 échanges pour qu'il y ait plus d'une chance sur deux pour que le même IV soit utilisé.
- Ainsi, la sécurité d'une clé de 104 bits n'était que virtuelle dans WEP. La vraie sécurité était sur 24 bits seulement !
- WEP est remplacé par WPA (utilise RC4 mais change IV à chaque paquet) et WPA2 (utilise AES).

Structure générale d'une fonction de hachage



IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm

L = number of input blocks
n = length of hash code
b = length of input block

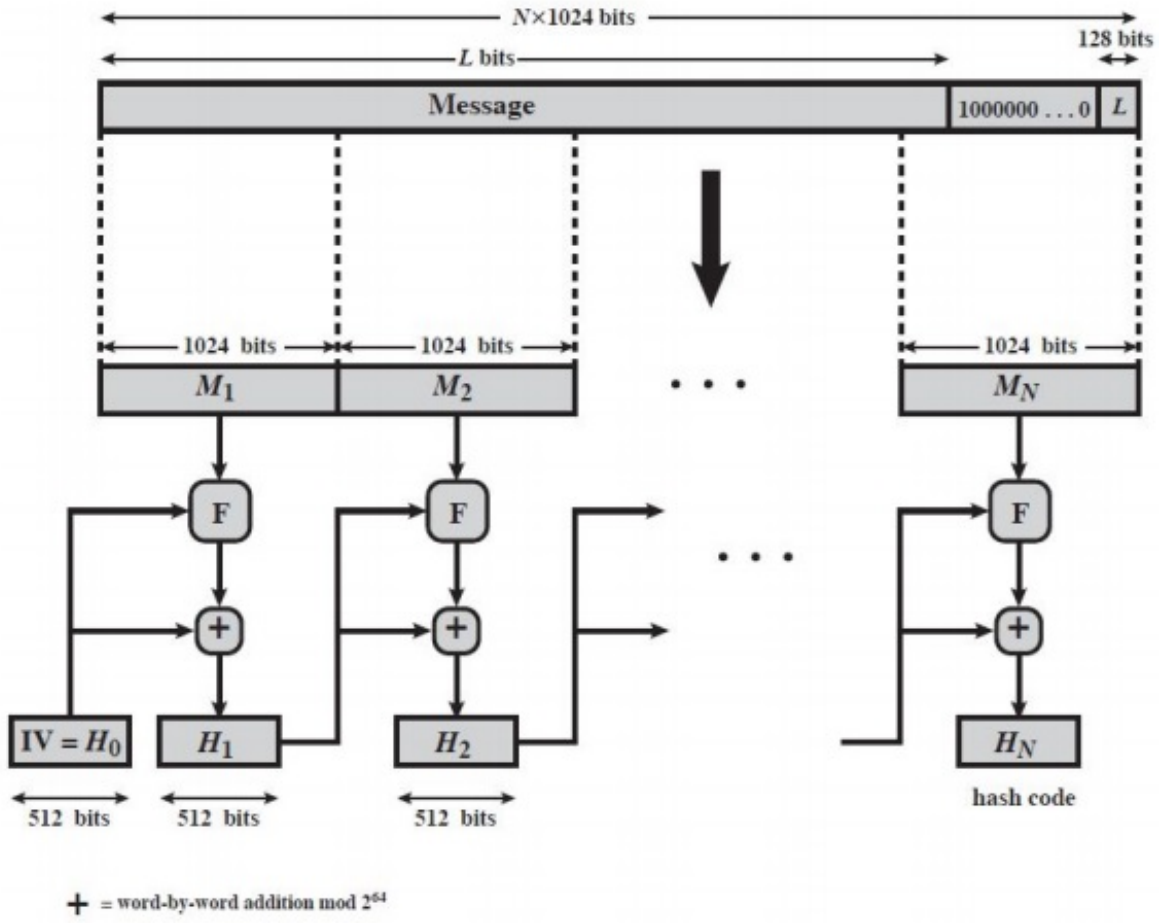
Secure Hash Algorithm (SHA)

- SHA a été concu par "National Institute of Standards and Technology (NIST)" et publié comme "federal information processing standard" (FIPS 180) en 1993
- a été revisé en 1995 comme SHA-1
- Basé sur la fonction de hachage MD4
- Produit un condensé de taille 160-bit
- En 2002 NIST produit une version révisée du standard pour définir 3 autres de SHA avec des longueurs 256, 384, and 512 Connus comme SHA-2

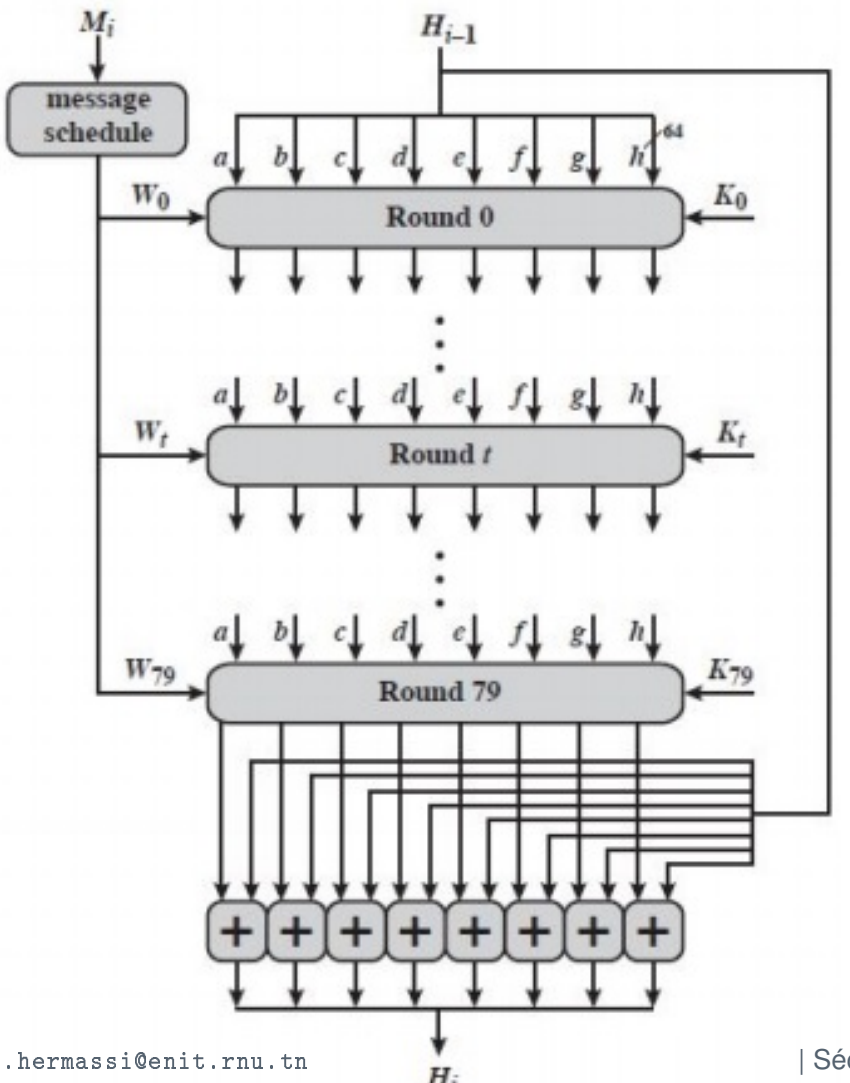
Comparaison des versions de SHA

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

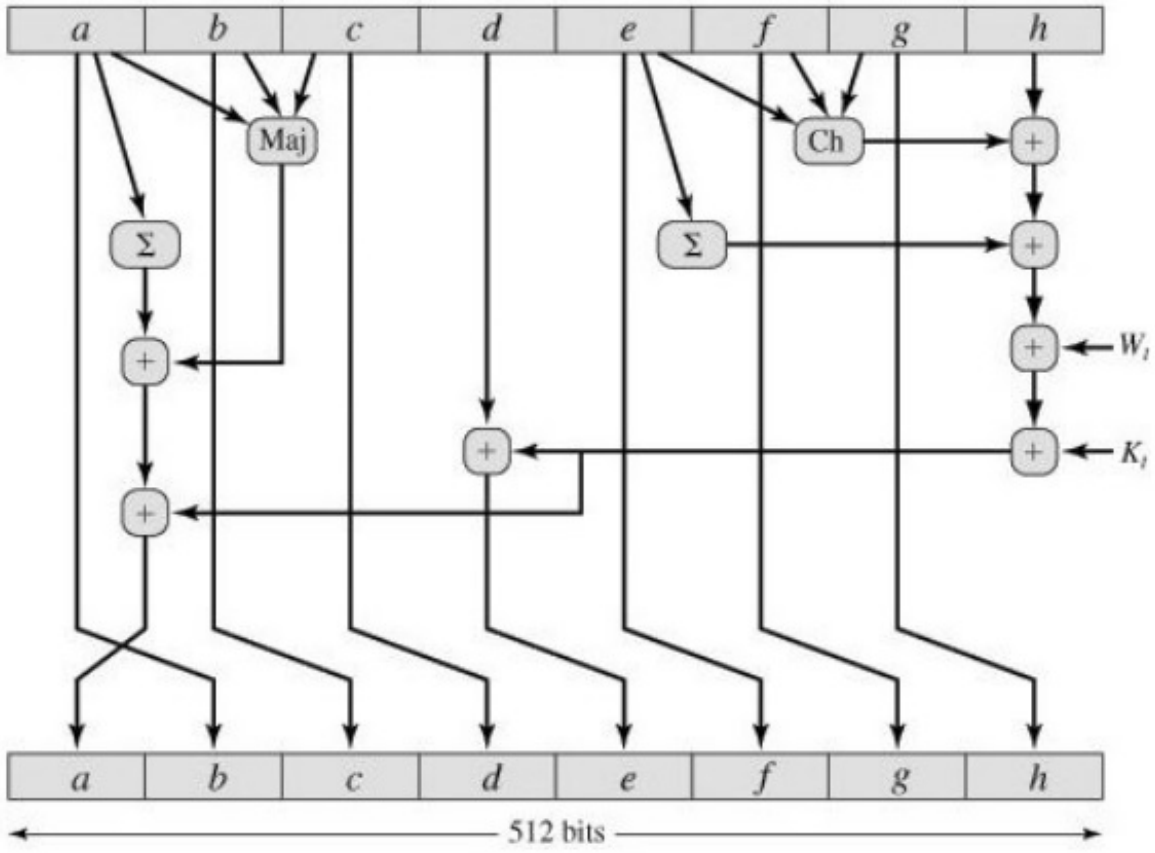
SHA-512


N×1024 bits
128 bits
L bits
Message
1000000 . . . 0
L
1024 bits
1024 bits
1024 bits
M1
M2
MN
F
F
F
+
+
+
IV = H0
H1
H2
HN
512 bits
512 bits
512 bits
hash code
+ = word-by-word addition mod 2^64

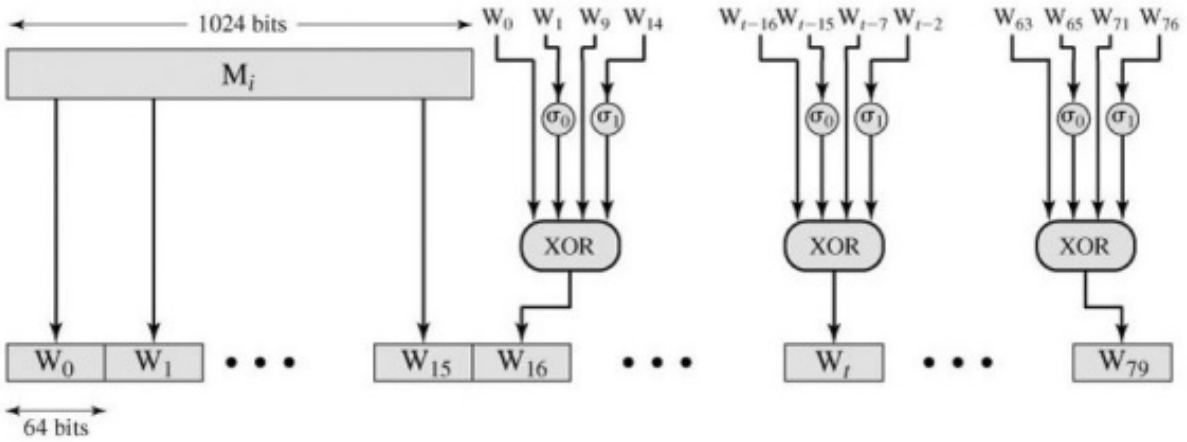
SHA-512 : traitement d'un bloc de 1024-Bit



SHA-512 : Mise à jour des buffers



SHA-512 : traitement du message Mi



Problèmes et contre-mesure des problèmes de sécurité

Différents types d'attaques/problèmes dans un réseau

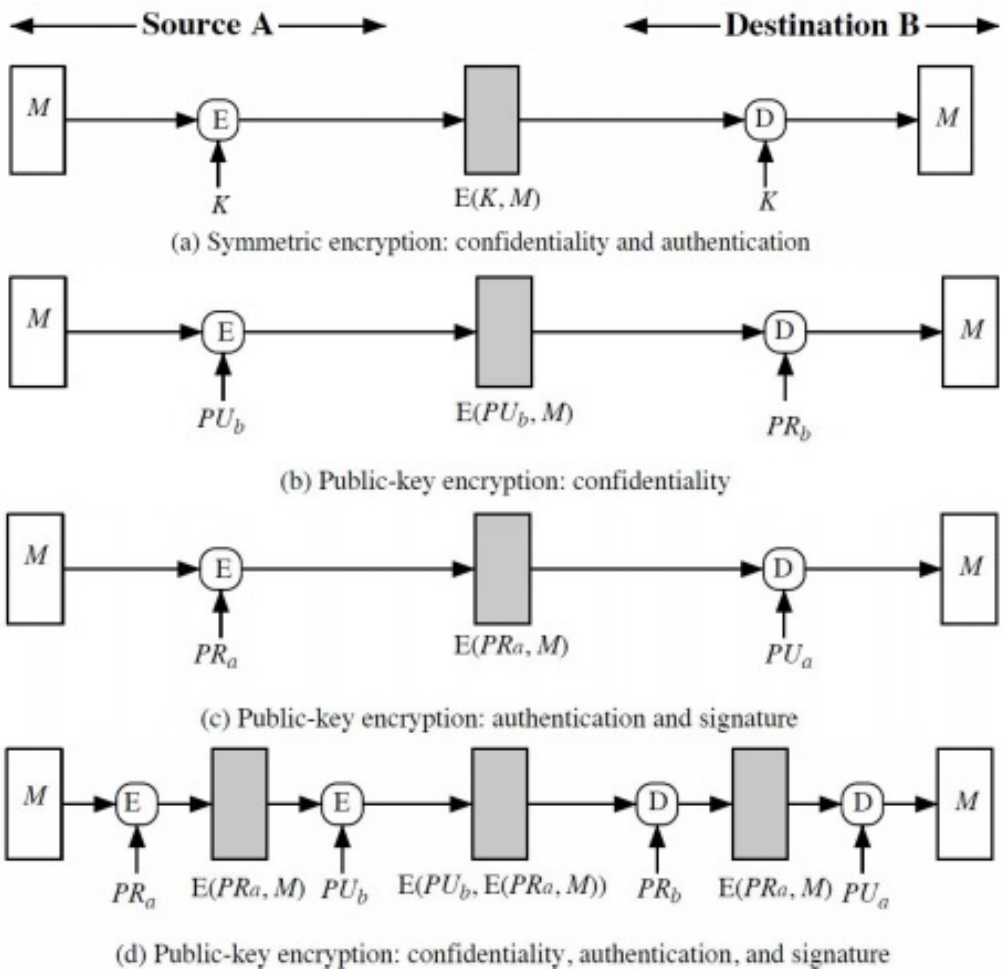
Différents types d'attaques/problèmes dans un réseau

- Divulgation des messages ⇛ Sol : cryptage
- Analyse de trafic ⇛ Sol : cryptage
- Mascarade : insertion de message depuis une source frauduleuse ⇛ Sol : Authentification du message
- modification de contenu : insertion, suppression, transposition et modification ⇛ Sol : Authentification du message
- modification en temps : retard ou rediffusion (replay) de message ⇛ Sol : Authentification du message
- Répudiation de la source : Déni de transmission du message par la source ⇛ Sol : Signature numérique
- Répudiation de la destination : Déni de réception du message par le destinataire ⇛ Sol : Signature numérique

Les techniques d'authentification de message

1. Les fonction de hachages : une fonction qui accepte comme entrée n message de longueur variable et fait sortir un condensé de longueur fixe. Le condensé est l'authentificateur du message (déjà vu)
2. Le cryptage du message : le ciphertext du message constitue son authentificateur
3. Le MAC (Message authentication code) : une fonction du message et d'une clé secrète qui produisent une sortie de longueur fixe MAC ce qui constitue l'authentificateur du message

plusieurs scénarios d'utilisation du cryptage



(a) Symmetric encryption: confidentiality and authentication

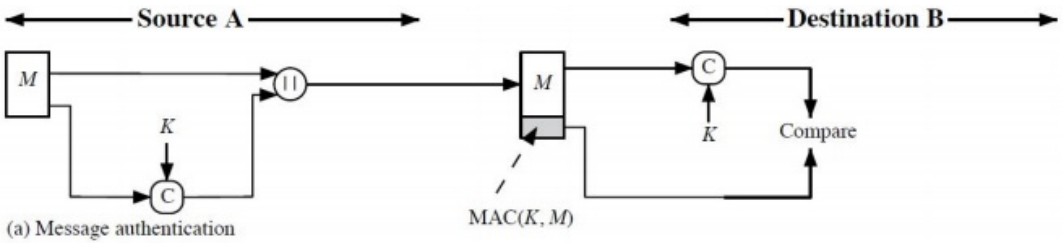
(b) Public-key encryption: confidentiality

(c) Public-key encryption: authentication and signature

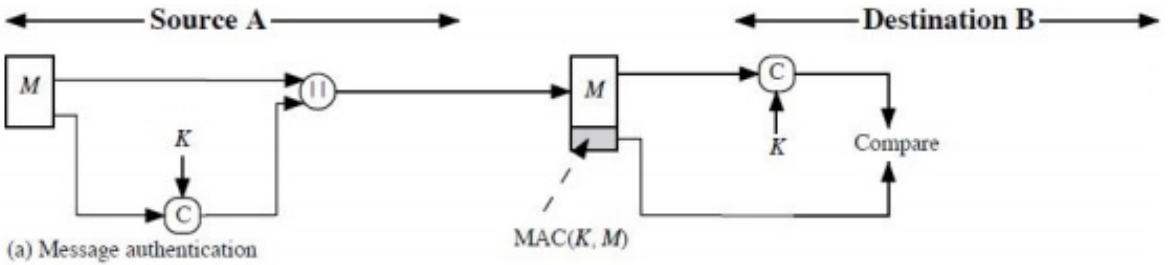
(d) Public-key encryption: confidentiality, authentication, and signature

Principe

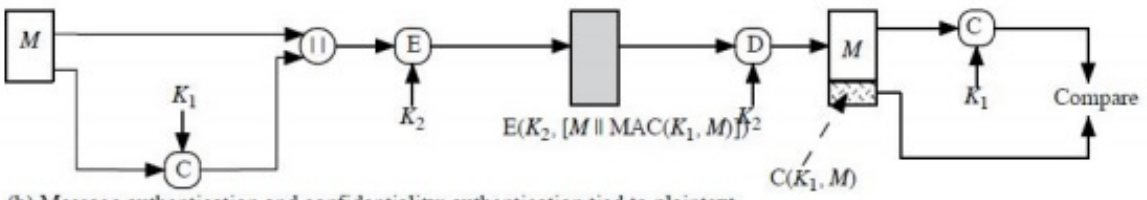
- Connu aussi par fonction de hachage à clé
- utilisé lorsque deux entités partageant la même clé pour authentifier l'information échangée entre eux
- Prend comme entrée une clé secrète K et un bloc de donnée M et produit un MAC=C(K,M)
- le MAC est associé au message lors de son envoi
- Si l'intégrité du message doit être vérifié, la fonction MAC est appliqué au message et le résultat est comparé au MAC associé (recu)
- un hacker qui veut modifier le message sera incapable de modifier le MAC sans la connaissance de la clé secrète
- Le MAC n'est pas une signature numérique



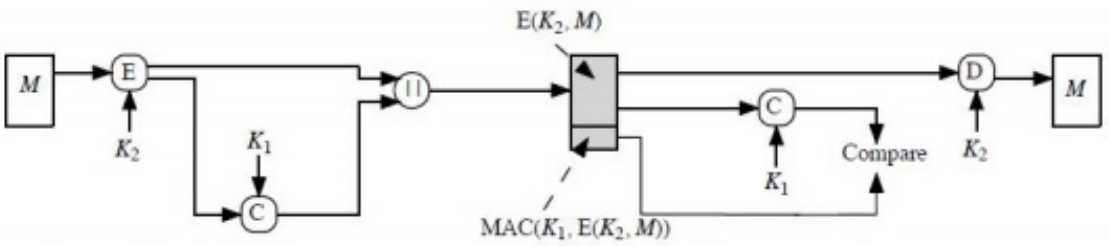
Utilisations basiques du MAC



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Cryptage authentifié

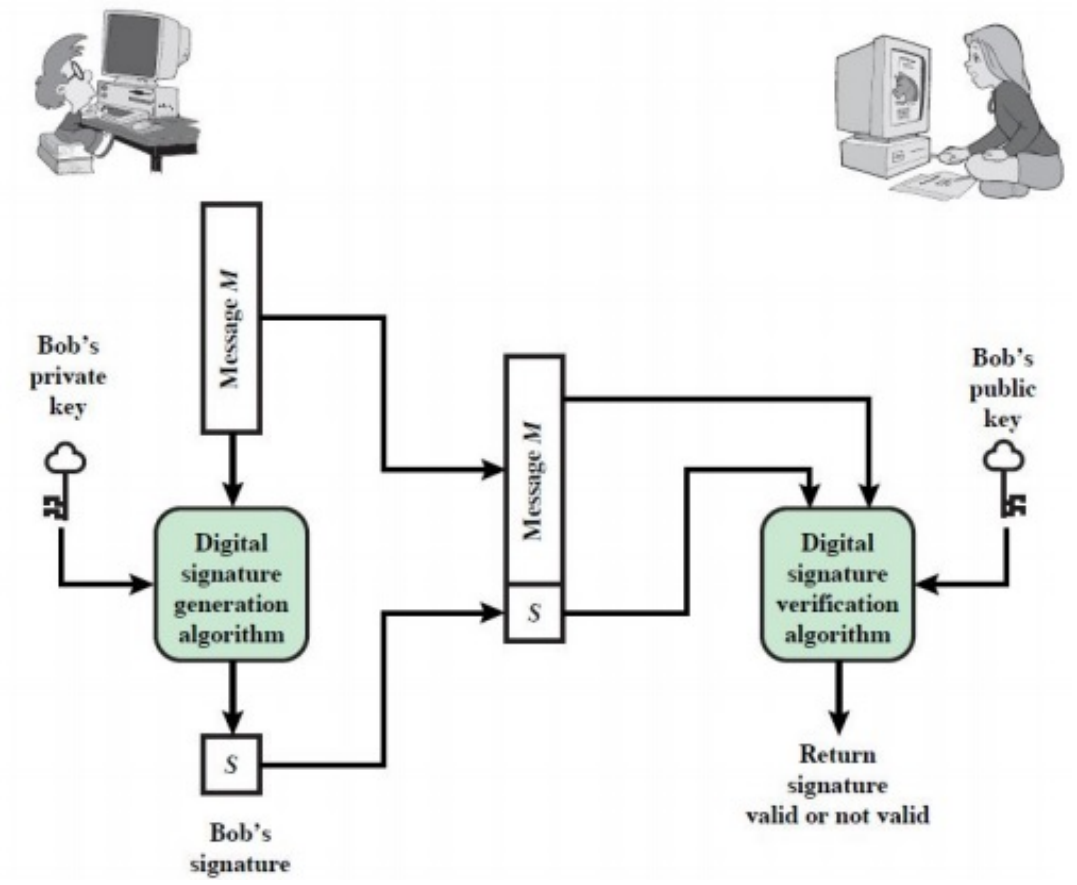
- Protèger la confidentialité et fournir l'authentification en meme temps
- Différent approches :
  - Hash-then-encrypt : $E(K, (M \parallel H(M))$
  - MAC-then-encrypt : $E(K_2, (M \parallel MAC(K_1, M))$
  - Encrypt-then-MAC : $C = E(K_2, M), T = MAC(K_1, C)$
  - Encrypt-and-MAC : $C = E(K_2, M), T = MAC(K_1, M)$
- decryptage et la vérification est facile

Principe

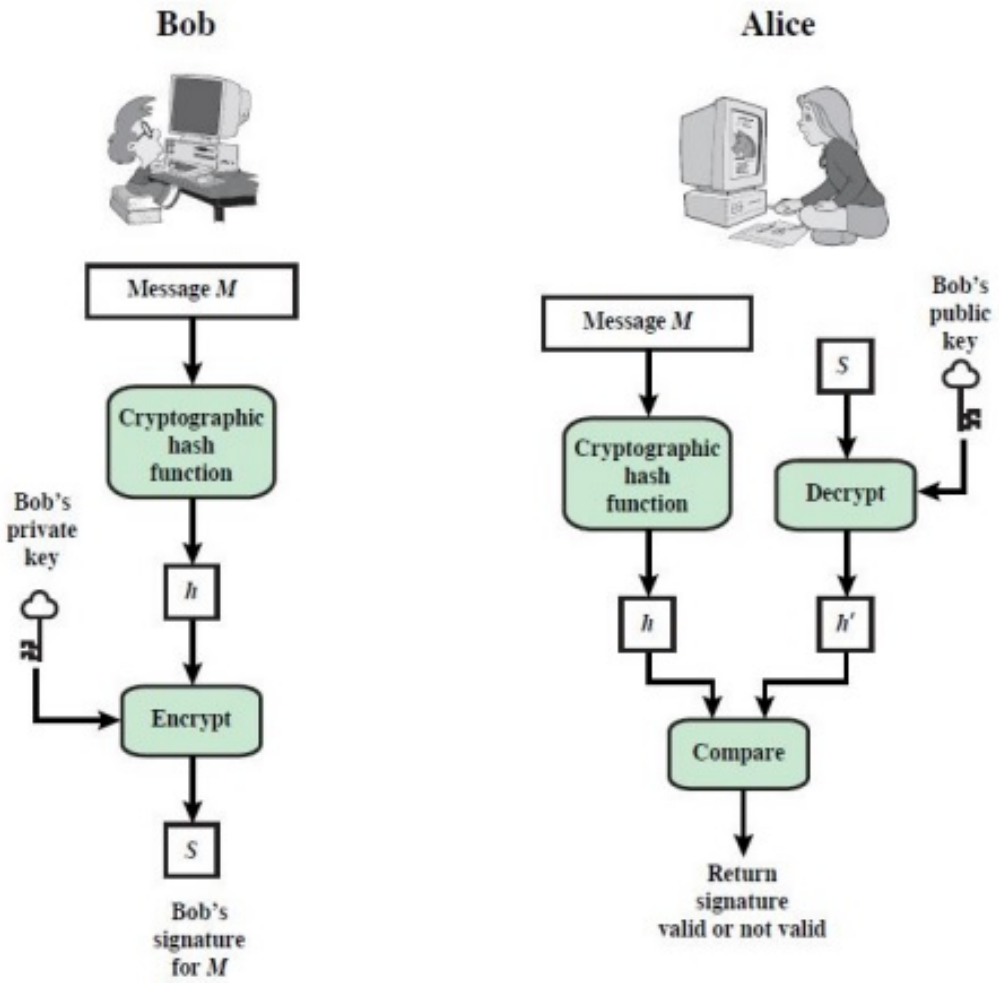
- Similaire qu'au MAC
- le condensé du message est chiffré par la clé privée de l'émetteur du message
- N'importe quelle personne connaissant la clé publique de l'émetteur peut vérifier l'intégrité du message
- un hacker qui veut modifier le message a besoin de connaître la clé privée de l'émetteur
- 3 propriétés :
  - elle doit vérifier l'auteur, le temps et la date du document signé
  - elle doit authentifier le contenu au temps de la signature
  - elle doit être vérifiée par une tierce partie pour résoudre les disputes

Signature numérique
Signature numérique

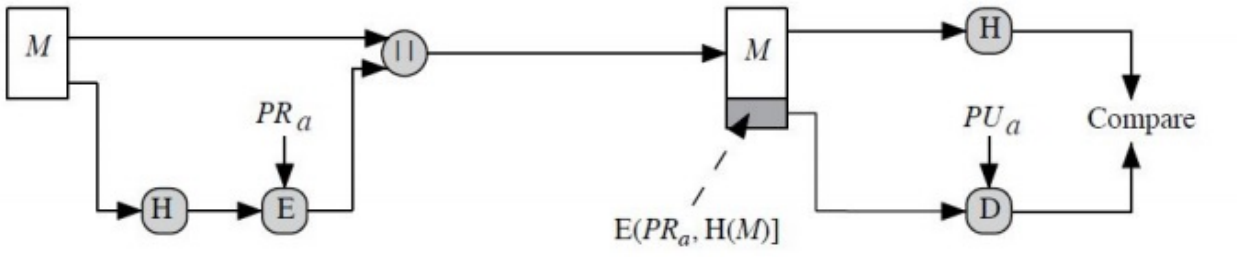
Modèle général de la signature numérique



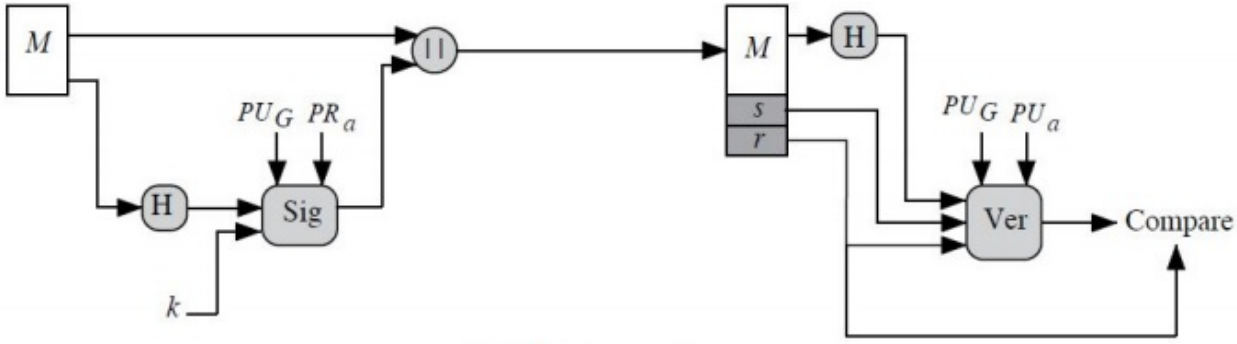
Modèle détaillé de la signature numérique



2 approches de la signature numérique



(a) RSA Approach



(b) DSS Approach

Merci pour votre attention!