Sécurité des systèmes répartis - 1

Sacha Krakowiak
Université Joseph Fourier
Projet Sardes (INRIA et IMAG-LSR)

http://sardes.inrialpes.fr/~krakowia

Plan de présentation

Sécurité informatique : objectifs (1)

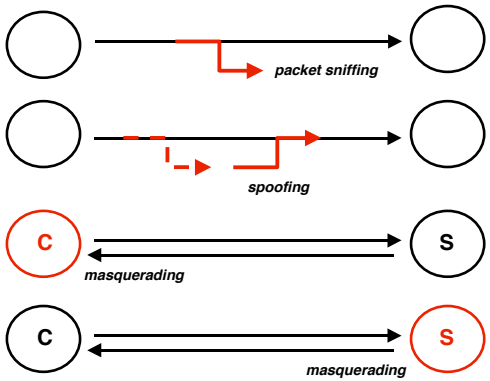
- **Préserver la confidentialité et l'intégrité des informations**
  - **Confidentialité** : empêcher la divulgation d'informations à des entités (sites, organisations, personnes, etc.) non habilitées à les connaître
  - **Intégrité** : empêcher toute modification d'informations (intentionnelle ou accidentelle) non explicitement requise par une entité habilitée
- **Garantir l'origine d'une information, l'identité d'une personne ou organisation**
  - **Authentification**
    - d'une information : prouver qu'une information provient de la source annoncée (auteur, émetteur)
    - d'une personne (ou groupe ou organisation) : prouver que l'identité est bien celle annoncée

Sécurité informatique : objectifs (2)

- **Protéger l'accès aux services**
  - Interdire l'accès d'un service à toute entité non explicitement autorisée à y accéder (**accès indu**)
  - Assurer l'accès effectif au service pour toute entité autorisée (empêcher le **déni de service**)
- **Fournir des éléments de preuve (en temps réel ou a posteriori)**
  - Sur la réalité de certaines actions
  - Sur les tentatives d'actions non autorisées

Remarque importante (cf tolérance aux fautes). Il n'existe pas de méthodes pour assurer la sécurité dans l'absolu, seulement des méthodes adaptées à des risques particuliers de violation de la sécurité. Les hypothèses d'attaque doivent donc être explicitement formulées, après **analyse soigneuse des risques**

Sécurité informatique : quelques attaques



espionnage de messages

injection, altération ou rejeu de messages

usurpation d'identité de client

usurpation d'identité de serveur

déni de service (empêcher l'accès d'usagers autorisés)

dénégation (non-reconnaissance d'actions effectuées, d'informations reçues)

Analyse des risques liés aux réseaux

Interception de messages

Risques potentiels :
violation de la confidentialité
usurpation d'identité (vol de mots de passe)

Injection de messages (ou réinjection de messages modifiés)

Risques potentiels :
usurpation d'identité (usager, site)
accès à des services indus
violation d'intégrité
violation de confidentialité
fausse attribution d'origine

déni de service (saturation du réseau ou des serveurs)

Sécurité informatique : notions de base

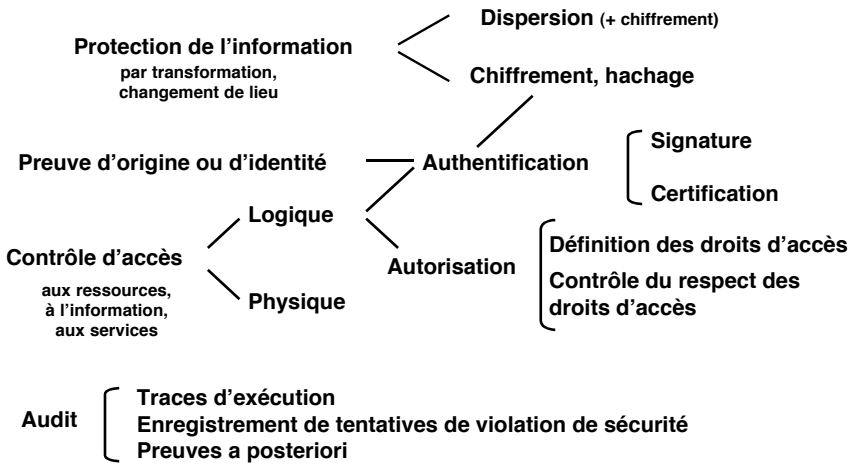
- **Définitions**
  - Droit (d'accès) : autorisation attachée
    - à l'obtention d'une information (lire un fichier, recevoir un message, etc.)
    - à la création ou à la modification d'information
    - à l'accès à un service (*login* sur une machine, accès à un SGBD, etc.)
    - à la création d'autorités et à l'attribution de droits
  - Autorité (en anglais : *principal*) : entité détentrice de droits (par exemple : usager, organisation, site, etc.)
    - une autorité doit pouvoir être authentifiée (reconnue comme telle)
  - Ressource (ou objet) : entité accessible à une autorité, et sujette à des droits d'accès (toute entité peut être considérée comme une ressource)

Sécurité informatique : politiques et mécanismes

La distinction entre politiques et mécanismes est un principe universel ; elle est particulièrement importante pour ce qui concerne la sécurité

- **Politiques**
  - Définition des autorités et des ressources
  - Organisation, règles d'usage
  - Spécification des droits
  - Exemples : classification des documents, accès aux moyens informatiques
  - Considérations sociales, juridiques, réglementaires plutôt que techniques
- **Mécanismes**
  - Moyens pour la mise en œuvre d'une politique
  - Exemple : protection physique, authentification par mot de passe, chiffrement, listes d'accès, capacités

Sécurité informatique : quelques moyens

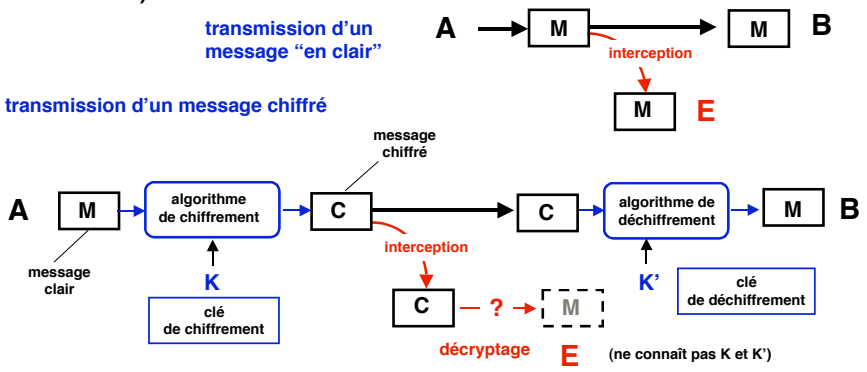


Un outil universel pour la sécurité : la cryptographie

- **Définition**
  - Initialement (et depuis 2000 ans), méthodes de dissimulation du contenu des messages (cryptographie = "écriture cachée")
- **Champ d'application**
  - Confidentialité des messages (et plus généralement, des informations)
  - Mais aussi (avancées récentes) :
    - intégrité des informations
    - authentification des personnes (ou des sites, des organisations, etc.)
      - preuve d'identité
    - authentification des documents (signature électronique)
      - preuve d'origine
      - non dénégation
    - authentification des programmes
      - code mobile

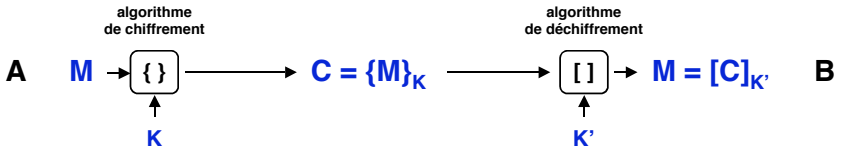
Introduction à la cryptographie

- Objectif : À l'origine, préserver la confidentialité d'une communication même si les messages sont interceptés par un tiers malveillant
- Moyen : Transformation des messages à l'aide de clés (informations gardées secrètes)



Défi : rendre le décryptage "très difficile" (i.e. irréalisable en pratique)

Cryptographie : notations de base



$M = [C]_{K'} = [\{M\}_K]_{K'}$

Chiffrement symétrique : K = K'

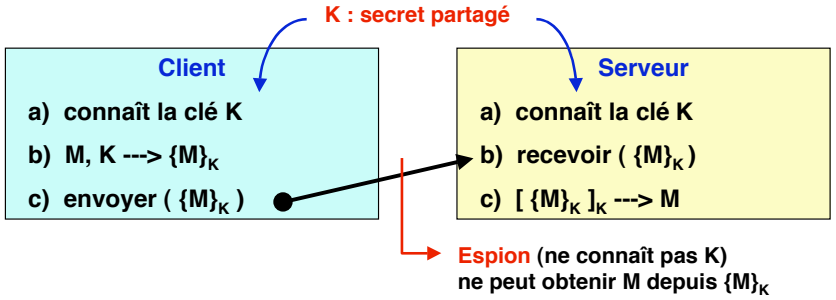
$$\begin{cases} M = [C]_K = [\{M\}_K]_K \\ C = \{M\}_K = \{[C]_K\}_K \end{cases}$$

Deux classes d'algorithmes sont utilisées en pratique

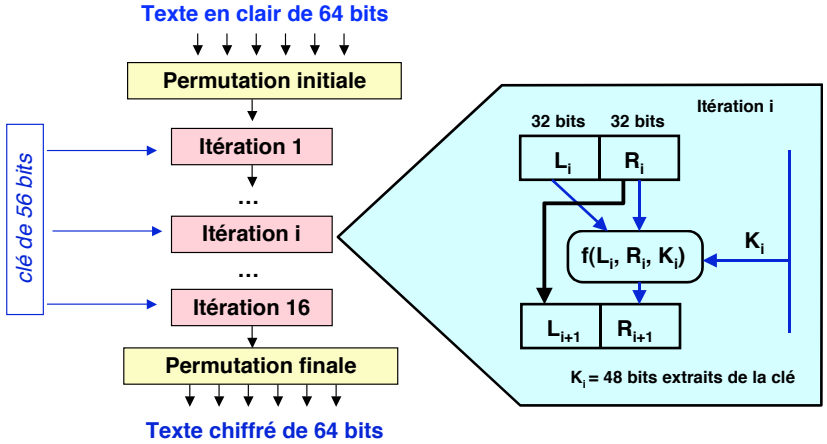
- clé secrète (en général symétrique), exemple DES
- clé publique (asymétrique, exemple RSA)

Système à clé secrète

- **Principes directeurs (chiffrement symétrique)**
  - clé secrète K partagée par les deux parties
  - fonctions de chiffrement { } et de déchiffrement [ ] (non secrètes)
    - $[\{M\}_K]_K$ ---> M



DES : *Data Encryption Standard*



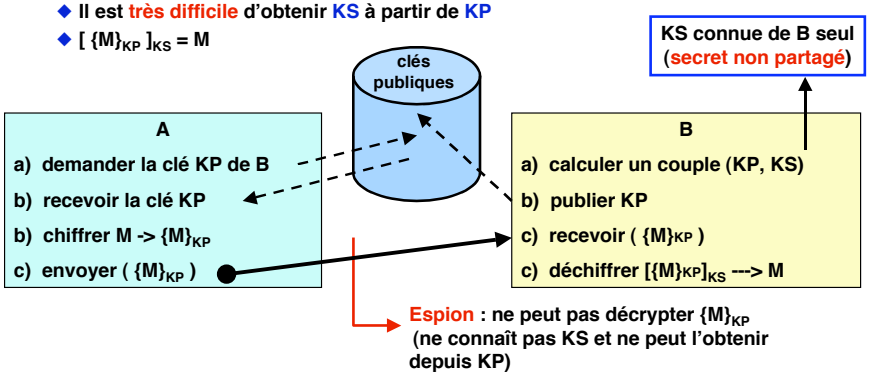
la difficulté de décryptage (non prouvée) repose sur la complexité du brouillage

DES : propriétés

- **Performances**
  - rapidité de chiffrement
    - circuit le plus rapide : 1 Gigabits/sec.
  - encombrement
    - l'implémentation la plus compacte occupe 700 octets
- **Sûreté**
  - une attaque exhaustive nécessite en moyenne 2^{55} tentatives …
  - … mais on sait faire beaucoup mieux
    - décryptage en 4 mois en 1997
    - décryptage en 3 jours en 1998 (avec une machine spécialisée)
    - décryptage en 22 heures en 1999 (avec 100 000 PC en réseau)
      - voir http://www.eff.org/pub/Privacy/Crypto/Crypto_misc/DESCracker
  - ☛ remèdes :
    - augmentation de la taille de la clé
    - triple chiffrement avec 3 clés (standard pour PPP)
      - Chiffrer avec clé A, déchiffrer avec clé B, chiffrer avec clé C (on peut prendre C = A)
      - Coût triplé, mais impossible à décrypter dans la pratique

Systèmes à clé publique (1)

- **Principes directeurs** (Diffie, Hellman 1976)
  - Chaque usager a une clé publique KP accessible à tous, utilisée par ses correspondants pour chiffrer les messages qui lui sont adressés
  - Chaque usager a une clé privée KS, (secrète) connue de lui seul, pour déchiffrer les messages qu'il reçoit
  - Il est très difficile d'obtenir KS à partir de KP
  - $[\{M\}_{KP}]_{KS} = M$



Systèmes à clé publique (2)

- **Propriétés**
  1. le calcul de $\{M\}_{KP}$ doit être facile et rapide
  2. connaissant $\{M\}_{KP}$ et KP, il doit être difficile de calculer M
  3. connaissant KS et $\{M\}_{KP}$, il doit être facile de calculer M
  4. il doit être facile, pour une autorité, de créer des couples (KP, KS)
  5. il doit être difficile de calculer KS à partir de KP
  - 1) et 2) définissent $\{\ \}_{KP}$ comme une fonction à sens unique
  - 3) implique l'existence d'une clé inverse KS dont la possession rend facile le décodage du message chiffré avec KP
  - 2) et 5) assurent l'inviolabilité du système
  - propriété essentielle : KS, la clé privée, n'est pas partagée
- **Une réalisation : RSA** (Rivest, Shamir, Adleman : 1977)
  - utilise la théorie des nombres (décomposition en facteurs premiers)

Algorithme de chiffrement à clé publique RSA (Rivest, Shamir, Adleman : 1978)

Choisir p, q nombres premiers grands ($> 10^{100}$)
soit $n = p \times q$ et soit $z = (p - 1)(q - 1)$
Choisir d tel que d et z soient premiers entre eux
Trouver e tel que $e \times d = 1 \bmod z$
(trouver le plus petit de $z + 1$, $2z + 1$, $3z + 1$, ... divisible par d)

Découper le message clair en blocs de k bits, avec $2^k < n$

La clé publique est (e, n). Chiffrement : $C = M^e \bmod n$
La clé privée est (d, n). Déchiffrement : $M = C^d \bmod n$
} coûteux

Le décryptage nécessite de trouver p et q, donc de factoriser n (difficile !)

L'algorithme est réversible : $M = [\{M\}_{KP}]_{KS} = [\{M\}_{KS}]_{KP}$ (utile pour l'authentification)

Si a et n premiers entre eux, alors $a^z \bmod n = 1$. Donc :
$M^{e.d} \bmod n = M^{e.d\ -\ 1} \times M \bmod n = M \bmod n$ (propriété démontrée)

Systèmes de chiffrement : comparaison

	Efficacité	Sécurité du canal de communication de la clé
Système à clé secrète	+	—
Système à clé publique	—	+

rapport de l'ordre de 100 à 1000

Combinaison des deux méthodes

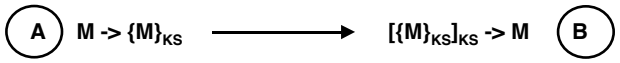
- Exemple : communication entre A et B, à l'initiative de A
- A engendre de façon aléatoire une clé secrète K
- A chiffre la clé à l'aide de la clé publique de B et l'envoie à B
- B déchiffre le message à l'aide de sa clé privée, et récupère la clé secrète K
- la suite des échanges est chiffrée à l'aide de la clé secrète K

Sécurité des systèmes client-serveur : outils de base

- **Confidentialité**
- **Authentification**
  - clé secrète
  - clé publique
- **Intégrité**
  - fonctions de hachage

Confidentialité

- **Utilisation directe de la cryptographie**
  - A envoie un message chiffré à M (par clé secrète pour des raisons d'efficacité)
  - B déchiffre le message avec la clé



  - Problème : distribution des clés (comment A et B obtiennent KS)
    - Accord préalable
    - Génération par un tiers de confiance
    - Création par l'un des partenaires et communication directe
    - Création conjointe par les deux partenaires

vu plus loin, combiné avec authentification

Authentification

- **Définition**
  - Prouver qu'une autorité est bien celle qu'elle prétend être
    - exemple : *login* d'un usager sur un système
- **Moyens** (pour authentifier A)
  - A fournit une information connue (en principe) de lui seul
    - exemple : mot de passe
  - A exhibe un objet qu'il est seul à détenir
    - exemple : carte à puce
  - A exhibe une caractéristique qui lui est propre
    - exemple : tests biométriques
  - A exécute une action dont lui seul (en principe) est reconnu capable
  - A fait appel à une autorité B qui certifie l'identité de A
    - mais le problème se repose (récursivement) pour B : besoin d'une autorité a priori

Utilisation de la cryptographie pour l'authentification

- **Principes**
  - Idée de base : R. Needham - M. Schroeder, 1978
  - Algorithme à clé secrète
    - nécessite une autorité de référence
  - Algorithme à clé publique
    - fonctionne sans autorité de référence (sinon pour valider la clé publique)
- **Applications**
  - Clé secrète
    - Kerberos : authentification pour système client-serveur
    - il y a aussi une version à clé publique
  - Clé publique
    - signature électronique

Authentification avec clé secrète (1)

- **Protocole de Needham - Schroeder (1978)**
  - A et B veulent s'authentifier mutuellement
  - A et B font confiance à un serveur d'authentification SA auquel ils confient leurs clés secrètes K_A et K_B
  - Le serveur d'authentification doit
    - fournir à A et B une clé de session K_{AB} pour leurs échanges futurs
    - authentifier A pour B (prouver à B que A est bien A)
    - authentifier B pour A (prouver à A que B est bien B)
  - Preuve de l'identité : détention de la clé secrète
  - Problèmes
    - les clés secrètes doivent rester secrètes
    - il faut prévoir la possibilité d'interception et de rejeu des messages

Authentification avec clé secrète (2)

I_A : identificateur unique

A->SA : M1 = (A, B, I_A)

S crée une clé K_{AB} et $T_{AB} = \{K_{AB}, A\}_{KB}$

SA -> A : M2 = $\{I_A, B, K_{AB}, T_{AB}\}_{KA}$ ← S est authentifié pour A par K_A

A déchiffre M2 et obtient K_{AB}, T_{AB}

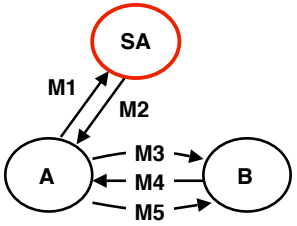
A -> B : M3 = T_{AB}

B déchiffre T_{AB} et crée I_B

B -> A : M4 = $\{I_B\}_{KAB}$ ← B est authentifié pour A car il a déchiffré T_{AB}

A -> B : M4 = $\{I_B - 1\}_{KAB}$

A est authentifié pour B car il possède K_{AB} délivré par SA



Authentification avec clé secrète (3)

- L'algorithme précédent présente un point faible
  - si un espion collecte les "vieux" tickets et réussit à obtenir une clé ancienne d'échange K_{AB}, il peut renvoyer un ticket ancien à B et se faire passer pour A
- Remède (utilisé dans la pratique)
  - rajouter une estampille (heure courante) dans le ticket
  - $T_{AB} = \{K_{AB}, A, t\}_{KB}$
  - à l'ouverture du ticket, on vérifie l'heure courante et on la compare avec la date du message

Authentification avec clé publique

- **Principe**
  - On utilise la réversibilité du chiffrement RSA
    $M = [\{M\}_{KP}]_{KS} = [\{M\}_{KS}]_{KP}$
  - Le chiffrement par KP procure la confidentialité (car il faut KS, privée, pour déchiffrer)
  - Le chiffrement par KS ne donne pas la confidentialité (car la clé de déchiffrement KP est publique), mais procure l'authentification (seul le détenteur de KS peut chiffrer avec KS)
  - Connaissance des clés publiques : 2 hypothèses
    - les clés publiques ont connues (communication directe)
    - un serveur d'authentification sert d'annuaire pour les clés publiques (certifiées) ; sa propre clé publique est connue de tous

Application : distribution de clés secrètes

- principe : utiliser clé publique pour la distribution de clés secrètes, avec authentification
- avantages
  - efficacité (on n'utilise le chiffrement à clé publique, coûteux, que pour un message court, la clé secrète)
  - pas de secret initialement partagé

Initialement : A connaît la clé publique de B, KPB
B connaît la clé publique de A, KPA

B crée une clé secrète KSS (aléatoire) et la chiffre avec sa clé privée $\{KSS\}_{KSB}$ (pour s'authentifier comme B)

B -> A : $\{\{KSS\}_{KSB}\}_{KPA}$

A déchiffre le message ci-dessus en deux étapes :

confidentialité	$[\{\{KSS\}_{KSB}\}_{KPA}]_{KSA}$ -> $\{KSS\}_{KSB}$	s'assure que le message vient de B
authentification	$[\{KSS\}_{KSB}]_{KPB}$ -> KSS	obtient la clé secrète KSS

A et B peuvent communiquer en utilisant KSS pour chiffrer leurs échanges
(si nécessaire, A peut d'abord s'authentifier vis-à-vis de B)

Authentification avec clé publique

A et B connaissent KPSA, clé publique de SA, serveur d'authentification et annuaire

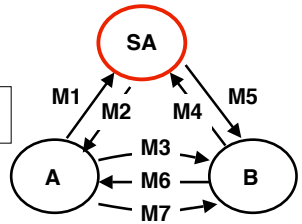
1) A->SA : M1 = (A, B)
2) SA -> A : M2 = $\{KPB, B\}_{KSSA}$
A déchiffre M2 avec KPSA et obtient KPB
3) A -> B : M3 = $\{I_A, A\}_{KPB}$
B déchiffre M3 avec KSB
4) B -> SA : M4 = (B, A)
5) SA -> B : M5 = $\{KPA, A\}_{KSSA}$
6) B -> A : M6 = $\{I_A, I_B, B\}_{KPA}$
7) A -> B : M7 = $\{I_B\}_{KPB}$

SA est authentifié pour A par K_{SSA}

SA est authentifié pour B par K_{SSA}

B est authentifié pour A car il a pu lire I_A (il a donc KSB)

A est authentifié pour B car il a pu lire I_B (il a donc KSA)



NB : on peut éviter 1, 2, 4, 5 si les clés publiques sont en cache

A et B peuvent maintenant créer une clé secrète à partir de I_A et I_B.

Pièges de la sécurité

Protocole (simplifié), sans SA (clés publiques connues)

1) A -> B : M3 = $\{I_A, A\}_{KPB}$
B déchiffre M3 avec KSB
2) B -> A : M6 = $\{I_A, I_B, B\}_{KPA}$
3) A -> B : M7 = $\{I_B\}_{KPB}$

X : intrus (connu de A), mène 2 échanges en parallèle a et b

1a) A -> X : $\{I_A, A\}_{KPX}$
1b) X(A) -> B : $\{I_A, A\}_{KPB}$
2b) B -> X(A) : $\{I_A, I_B\}_{KPA}$
2a) X -> A : $\{I_A, I_B\}_{KPA}$
3a) A -> X : $\{I_B\}_{KPX}$
3b) X(A) -> B : $\{I_B\}_{KPB}$

recopie

1a) A -> X : $\{I_A, A\}_{KPX}$
1b) X(A) -> B : $\{I_A, A\}_{KPB}$
2b) B -> X(A) : $\{I_A, I_B, B\}_{KPA}$
2a) X -> A : $\{I_A, I_B, B\}_{KPA}$
3a) A -> X : $\{I_B\}_{KPX}$
3b) X(A) -> B : $\{I_B\}_{KPB}$

recopie

ne marche pas car A attend $\{I_A, I_B, X\}_{KPA}$

Intégrité

- **Définition**
  - Une information possède la propriété d'intégrité si les seules modifications qu'elle subit sont celles explicitement voulues ; sont donc exclues les modifications par malveillance ou par corruption accidentelle
  - L'intégrité peut être garantie ou simplement vérifiée
- **Garantie de l'intégrité**
  - contre malveillance : contrôle des droits d'accès
  - contre modification accidentelle : redondance
    - duplication (cf tolérance aux fautes)
    - codes correcteurs
- **Vérification de l'intégrité**
  - repose sur la redondance
    - codes détecteurs
    - fonction de hachage

Fonctions de hachage

- **Notions de base**
  - Hachage = compression d'une information (avec perte)

M -> H(M), H fonction de hachage (*hashing function*)
H(M) a une taille fixe ; en général H(M) << M

  - Usages
    - clé de recherche (non nécessairement univoque)
    - vérification d'intégrité (cas particulier : CRC)
- **Propriétés (souhaitables) d'une fonction de hachage**

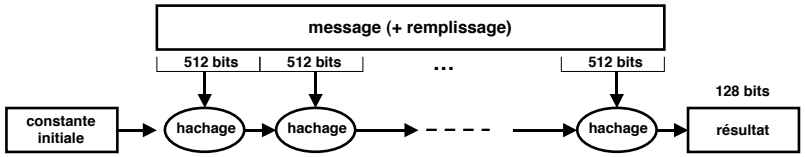
1) Il est "difficile "de reconstituer M à partir de H(M)
2) Étant donné M, il est "difficile" de trouver M' tel que H(M') = H(M)
3) Il est "difficile" de trouver un couple (M, M') tel que H(M') = H(M)

"difficile" = comme d'habitude, irréalisable en pratique (aujourd'hui)

Exemple de fonction de hachage

- **MD5 (*Message Digest*) [Rivest, 1992]**

RFC1321 : http://www.rfc-editor.org/rfc/rfc1321.txt



Efficace (logiciel : 85 Mbit/s sur un Alpha ; matériel : x100 Mbit/s
Semble posséder les propriétés 1, 2, 3 (pas de preuve : conjectures, tests empiriques)

Autre exemple : SHA (*Secure Hash Algorithm*)
produit 160 bits ; standard aux USA : http://www.itl.nist.gov/fipspubs/fip180-1.htm

Usage : signature électronique, cf plus loin

Problèmes des fonctions de hachage

- **Collisions...**
  - L'absence de collision dans les fonctions de hachage actuelles MD5 et SHA n'a pas été démontrée (elle est seulement improbable)
  - En fait, une collision dans SHA-0 a été trouvée en 2004, c'est-à-dire
    2 messages m1, m2, m1≠m2 et SHA-0(m1) = SHA-0(m2)
  - Des collisions ont également été trouvées en 2004 dans MD-5
- **Conséquences**
  - Cela veut-il dire que ces fonctions ne sont pas sûres en pratique?
    - L'attaque sur SHA-0 (trouver m1 et m2) a demandé 80 000 heures de calcul sur un processeur puissant
    - Le problème de trouver m2 (≠m1) pour m1 donné, tel que hash(m1)=hash(m2) n'est pas résolu pour le moment
  - Mais...
    - Il faut envisager dès maintenant une migration vers de nouvelles fonctions plus sûres en pratique
    - Le problème de la preuve de non-collision reste entier