Sécurité des systèmes répartis
1. Principes et techniques de base

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

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

Introduction
- les problèmes de la sécurité
- un outil de base : la cryptographie
  - principes, clé secrète, clé publique

Sécurité des systèmes client-serveur : techniques de base
- confidentialité
- authentification (avec clé privée ou clé publique)
- intégrité : fonctions de hachage

Sécurité des systèmes client-serveur : applications
- un service d'authentification : Kerberos
- signature électronique
- distribution des clés et certificats
- protocoles sécurisés : SSL, SET

Contrôle d'accès
- politiques et mécanismes

Sécurité informatique : objectifs (1)

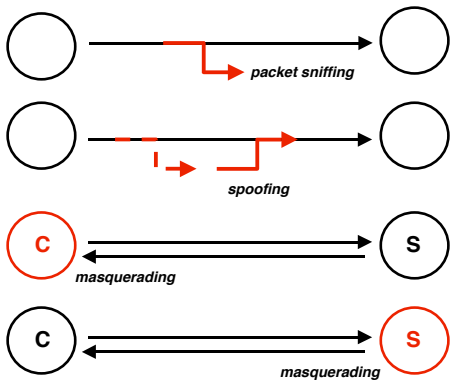
- **Préserver la confidentialité et l'intégrité des informations**
  - **Confidentialité** : empêcher la divulgation d'informations à des entités (sites, organisations, personnes, etc.) non habilitées à les connaître
  - **Intégrité** : empêcher toute modification d'informations (intentionnelle ou accidentelle) 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

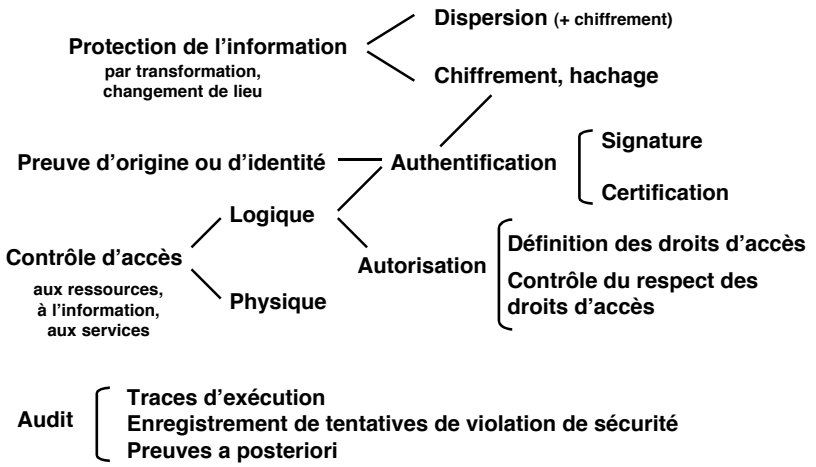
- **Définitions**
  - Droit (d'accès) : autorisation attachée
    - à l'obtention d'une information (lire un fichier, recevoir un message, etc.)
    - à la création ou à la modification d'information
    - à l'accès à un service (*login* sur une machine, accès à un SGBD, etc.)
    - à la création d'autorités 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

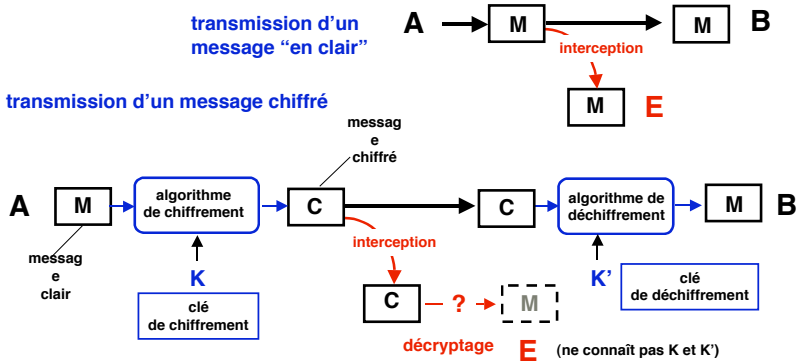


Un outil universel pour la sécurité : la cryptographie

- **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

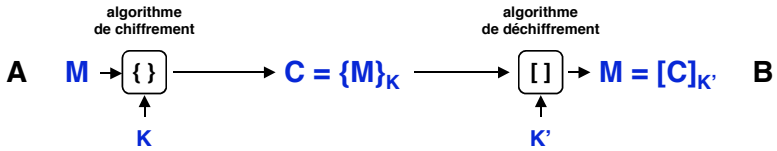
Introduction à la cryptographie

- 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'

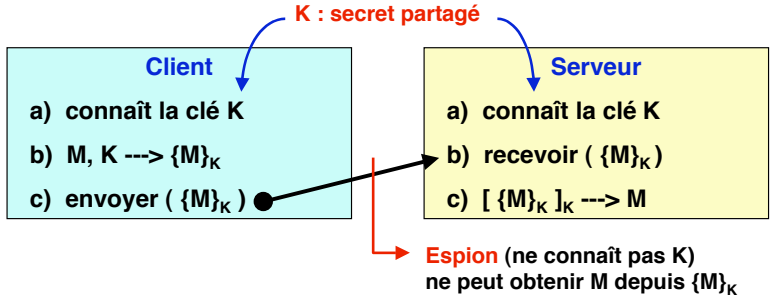
$$\begin{cases} M = [C]_K = [\{M\}_K]_K \\ C = \{M\}_K = \{[C]_K\}_K \end{cases}$$

Deux classes d'algorithmes sont utilisées en pratique

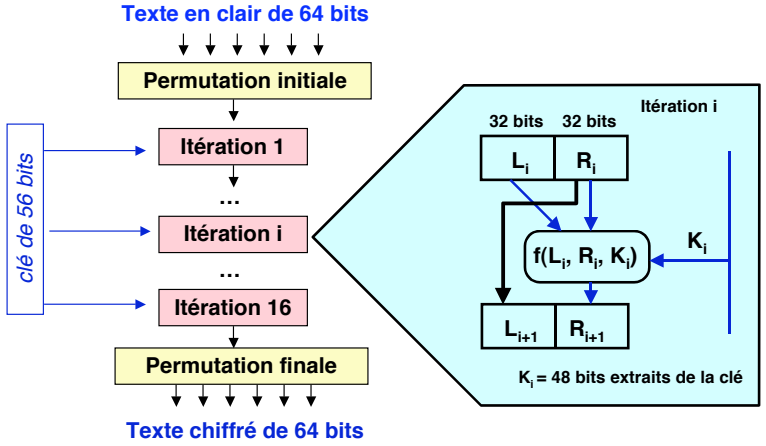
- clé secrète (en général symétrique), exemple DES
- clé publique (asymétrique, exemple RSA)

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*



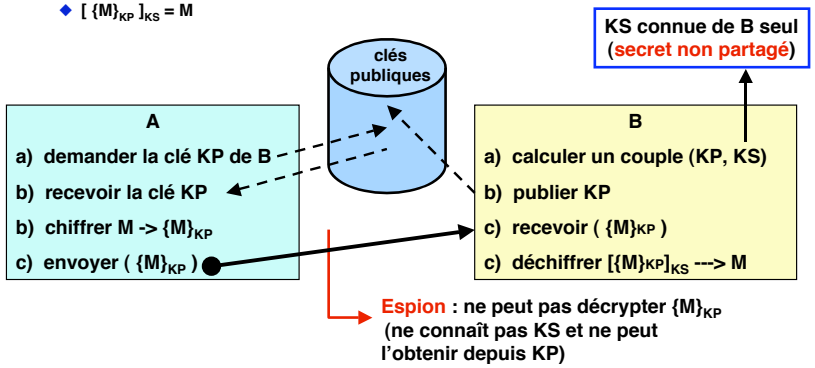
la difficulté de décryptage (non prouvée) repose sur la complexité du brouillage

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 déchiffrement 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	–	+

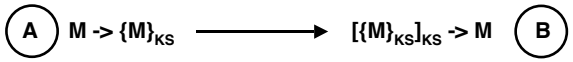
rapport de l'ordre de 100 à 1000

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

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

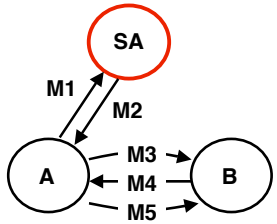
A déchiffre M2 et obtient K_{AB}, T_{AB}

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

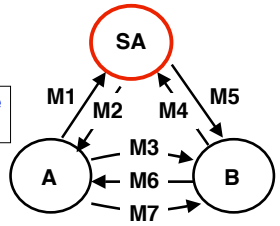
1) A->SA : M1 = (A, B)
2) SA -> A : $M2 = \{KPB, B\}_{KSSA}$
   A déchiffre M2 avec KPSA et obtient KPB
3) A -> B : $M3 = \{I_A, A\}_{KPB}$
   B déchiffre M3 avec KSB
4) B -> SA : M4 = (B, A)
5) SA -> B : $M5 = \{KPA, A\}_{KSSA}$
6) B -> A : $M6 = \{I_A, I_B, B\}_{KPA}$
7) A -> B : $M7 = \{I_B\}_{KPB}$

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**

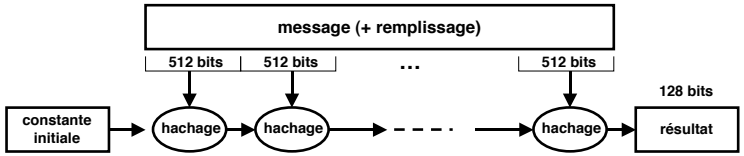
1) Il est "difficile "de reconstituer M à partir de H(M)
2) Étant donné M, il est "difficile" de trouver M' tel que H(M') = H(M)
3) Il est "difficile" de trouver un couple (M, M') tel que H(M') = H(M)

"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

Sécurité des systèmes client-serveur : Partie 2 : Applications

- Un service d'authentification : Kerberos
- Signature électronique
- Distribution des clés publiques et certification
- Contrôle d'accès
- Protocoles sécurisés
  - SSL
  - SET
- Sécurisation des réseaux (pare-feux)

Un service d'authentification : Kerberos

- Historique
  - développé au MIT pour le projet Athena
  - aujourd'hui : produit standard
- Objectif
  - protéger les serveurs partagés des accès non autorisés depuis les stations de travail (plusieurs milliers)
- Conditions de fonctionnement
  - les serveurs ne font aucune confiance aux clients (les stations clients sont ouvertes, le client peut réinstaller un système)
  - les clients accordent une confiance limitée aux serveurs
  - l'authentification est contrôlée par des serveurs spécialisés physiquement protégés

Un service d'authentification : Kerberos

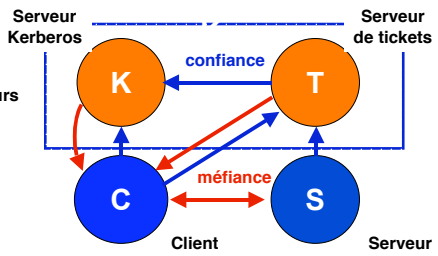
Relations de confiance

Le serveur Kerberos détient

- les mots de passe de tous les utilisateurs pour l'accès aux stations clientes
- la clé privée du serveur de tickets

Le serveur de tickets détient

- les clés privées de tous les serveurs



- Principes directeurs
  - mode d'exécution client-serveur
  - vérification de l'identité d'un «client» (utilisateur sur une station)
  - contrôle du droit d'accès à un serveur pour le client
  - fourniture au client d'une clé d'accès (*ticket*) pour le serveur
    - clé différente pour chaque serveur
    - clé valide pour une période de temps finie

Kerberos (2)

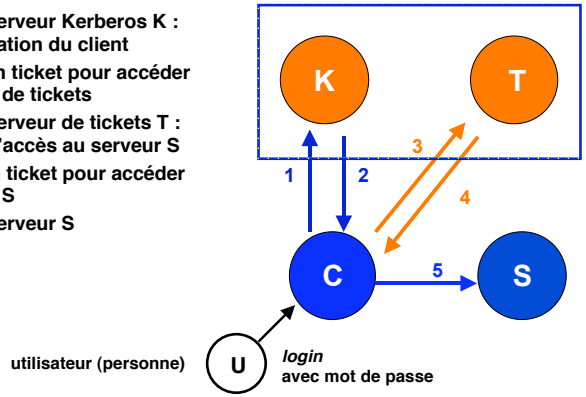
- Principe de fonctionnement : certificats « infalsifiables »
  - Ticket : caractérise une session entre un client C et un serveur S

    $T_{cs} = \{S, C, adr, Td, life, K_{cs}\}_{Ks}$
    - adr : adresse IP du client
    - Td : heure de début de session
    - life : durée maximale de session
    - K_{cs} : clé de session partagée par C et S
    - K_s : clé permanente (secrète) du serveur S
  - Le ticket ne peut être déchiffré que par le serveur, non par le client
  - Authentifieur : caractérise le client à un instant *t* vis-à-vis d'un serveur

    $A_{cs}(t) = \{C, adr, t\}_{Kcs}$
    - engendré par le client
    - permet une authentification « permanente » par le serveur

Kerberos (3)

- **Architecture**
  - ❶ accès au serveur Kerberos K : authentification du client
  - ❷ retour d 'un ticket pour accéder au serveur de tickets
  - ❸ accès au serveur de tickets T : contrôle d 'accès au serveur S
  - ❹ retour d'un ticket pour accéder au serveur S
  - ❻ accès au serveur S



Kerberos (4)

- **Interaction entre client et serveur Kerberos**

❶ C ---> K : message M1 = $\{C, T\}_{mdp(u)}$ chiffré par le mot de passe de U (que connaît K)

K déchiffre M1 et crée une clé de session $K_{c,T}$ pour chiffrer le dialogue entre C et T

K crée un ticket $T_{C,T}$ pour autoriser l'accès du client C au serveur T

$T_{C,T} = \{T, C, adr, td, life, K_{C,T}\}_{KT}$ (K connaît la clé K_T)

❷ K ---> C : message M2 = $\{K_{C,T}, T_{C,T}\}_{mdp(c)}$

C déchiffre M2 à l 'aide de mdp(c) et mémorise la clé $K_{C,T}$

C mémorise le ticket $T_{C,T}$ (sans pouvoir le déchiffrer)

Kerberos (5)

- **Interaction entre le client et le serveur de tickets**

C construit un authentifieur : $A_{C,T}(t1) = \{C, adr, t1\}_{Kc,T}$

❸ C ---> T : message M3 = $(A_{C,T} ; T_{C,T} ; S)$

rappel : $T_{C,T} = \{T, C, adr, td, life, K_{C,T}\}_{KT}$

T déchiffre le ticket $T_{C,T}$ à l'aide de sa clé K_T, vérifie sa validité, et récupère ainsi la clé de session $K_{C,T}$

T déchiffre l'authentifieur $A_{C,T}$ à l 'aide de la clé de session $K_{C,T}$, et récupère l 'identification du client

T contrôle le droit d 'accès du client C au serveur S

T crée une clé de session $K_{C,S}$ pour chiffrer le dialogue entre C et S

T crée un ticket $T_{C,S}$ pour autoriser l'accès du client C au serveur S

$T_{C,S} = \{S, C, adr, td, life, K_{C,S}\}_{Ks}$ (T connaît la clé K_s)

❹ T ---> C : message M4 = $\{K_{C,S}, T_{C,S}\}_{Kc,t}$

Kerberos (6)

- **Interaction entre le client et le serveur S**

C déchiffre le message M4 à l 'aide de la clé Kc,t et mémorise $K_{C,S}$

C mémorise le ticket $T_{C,S}$ (sans pouvoir le déchiffrer)

(rappel : $T_{C,S} = \{S, C, adr, td, life, K_{C,S}\}_{Ks}$)

C construit un authentifieur : $A_{C,S}(t2) = \{C, adr, t2\}_{Kc,s}$

C ---> S : message M5 = (requête, $T_{C,S}$, $A_{C,S}$)

S déchiffre le ticket $T_{C,S}$ à l 'aide de sa clé K_S, vérifie sa validité, et récupère ainsi la clé de session $K_{C,S}$

S déchiffre l'authentifieur $A_{C,S}$ à l 'aide de la clé de session $K_{C,S}$

La session peut commencer entre C et S

Signature électronique : besoins

- Fonctions d'une signature "ordinaire"
  - Authentifier le signataire (garantir son identité)
  - Engager le signataire (elle peut lui être opposée en cas de litige)
  - Conséquences
    - une signature doit être difficile à contrefaire
    - la signature et le document sont indissociables
    - une signature ne doit pas pouvoir être reniée
    - le document signé ne doit pas être changé après signature
- La signature électronique doit avoir les mêmes propriétés
  - authentification
  - difficulté de contrefaçon
  - non-dénégation
  - indissociabilité d'avec le document signé (pas de copier-coller)
  - intégrité du document

Signature électronique : principes

- Plusieurs degrés possibles de garanties pour un message
  - Intégrité seule
  - Intégrité + confidentialité
  - Intégrité + confidentialité + authentification (garantie d'origine)
- Exemple simple : intégrité seule
  - Utilisation d'une fonction de hachage H(M), par exemple MD5
  - Envoyer M+H(M) ne suffit pas
    - un intrus peut intercepter M, changer M en M', calculer H(M'), et renvoyer M'+H(M')
  - Une solution avec clé secrète K partagée par A et B
    - A concatène M, L (longueur de M) et K et calcule D = H(M, L, K)
    - A envoie à B : M+D
    - B recalcule H(M, L, K) à partir du message M reçu (il connaît K)
    - B vérifie que H(M, L, K) = D

Signature électronique : réalisation

- Signature avec intégrité et authentification
  - Idée initiale :
    - A chiffre le message avec sa clé privée (déchiffrable avec la clé publique correspondante) ; soit KSA
    - A envoie à B : (M, $\{M\}_{KSA}$)
    - B déchiffre $\{M\}_{KSA}$ avec KPA (publique)
    - B compare $[\{M\}_{KSA}]_{KPA}$ avec M
    - Si différents, le message (ou la signature) a été altéré
    - Propriétés : vérification d'intégrité, authentification, non dénégation
  - Problème : très coûteux si le message M est long
  - Amélioration : ne chiffrer (avec clé privée) qu'un condensé du message (et chiffrer en outre le corps du message avec une clé secrète - chiffrement peu coûteux - si la confidentialité est requise)

Signature électronique : réalisation

- Signature avec intégrité, confidentialité et authentification

A et B s'accordent sur une clé secrète KS
A calcule H(M), par exemple avec MD5
A envoie à B

$$\{M\}_{KS}, \{H(M)\}_{KSA}$$

B déchiffre $\{M\}_{KS}$ avec KS, $\{H(M)\}_{KSA}$ avec KPA

$[\ ]_{KS}$ $\qquad$ $[\ ]_{KPA}$

M -> H(M) =? DM

B calcule H(M) (avec le même H() que A) et le compare avec DM.
test d'intégrité (OK si H(M) = DM, violation sinon)
confidentialité grâce à KS
authentification et non dénégation (car seul A a pu chiffrer H(M))
interception et modification de M "difficiles" car il faut à la fois KS et KSA

Certification (1)

Problème des signatures électroniques : comment être sûr de l'identité du signataire ?

On sait que

Le signataire possède la clé privée associée à la clé publique indiquée pour lui

Mais on n'a pas la garantie que la clé publique listée pour XX appartient réellement à XX.

Exemple : un escroc fabrique un couple (clé publique Kp, clé privée Ks) et affiche sur une page Web (ou diffuse par *mail*) : "la clé publique de XX est Kp"

Réponse : la certification

Repose sur des "tiers de confiance", ou autorités habilitées à délivrer des certificats

Un certificat asssocie une clé publique à une personne (ou organisation), + autres informations, et garantit l'authenticité de ces informations

Un certificat a une date d'expiration (doit être renouvelé)

Certification (2)

- **Infrastructure de Gestion des Clefs (IGC)**

  en anglais : *Public Key Infrastructure* (*PKI*)

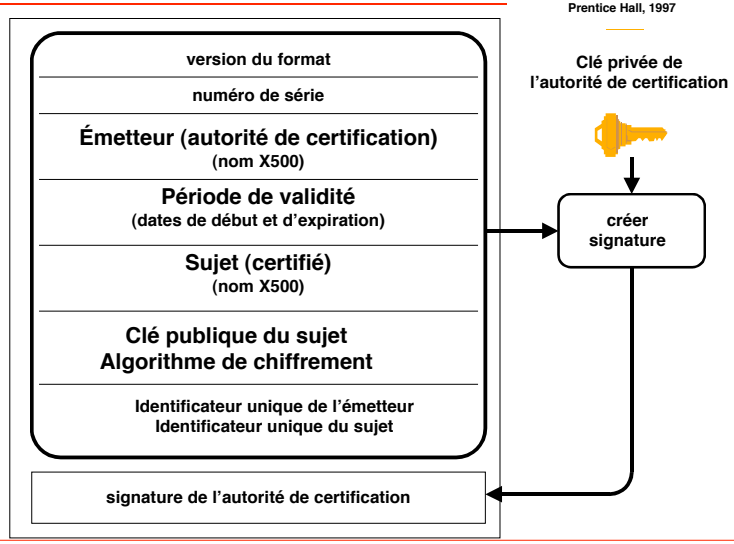
  - **Certification = garantie que**
    - **la clef est bien celle appartenant à la personne (ou l'organisation) avec qui les échanges sont envisagés**
    - **le possesseur de cette clef est « digne de confiance »**
    - **la clef est toujours valide**
  - **La certification repose sur la confiance accordée à l'autorité certificatrice. L'IGC est la structure permettant d'établir cette confiance**
  - **Diverses solutions**
    - **hiérarchie (arbre) : repose sur la confiance dans la racine**
    - **réseau de confiance (communauté, exemple PGP)**

Certification (3)

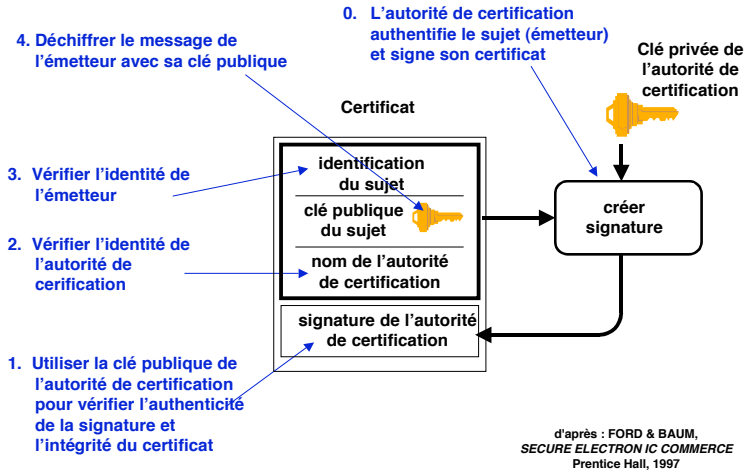
- **Standards**
  - **certificat : protocole normalisé X509 (ITU-T X.509 international standard V3 - 1996) *(RFC2459)***
  - **liste des certificats invalidés (CRL *Certificate Revocation List*)**
    - **structures de données signées**
    - **format défini par le protocole X509 V2 CRL *(RFC2459)***
  - **support logique de publication : LDAP « *Lightweight Directory Access Protocol* » *(RFC2251)***
- **Garantie**
  - **Un certificat est garanti par la signature de l'autorité qui l'a délivré (à condition**
    - **que la date d'expiration ne soit pas passée**
    - **qu'il n'ait pas été invalidé**
- **Fonctionnement**
  - **Le contrôle des certificats est intégré au navigateur**
    - **le navigateur maintient la liste des autorités "reconnues" par l'usager**
    - **le navigateur maintient les certificats obtenus par l'usager**

Certificats (1)

d'après : FORD & BAUM,
SECURE ELECTRON IC COMMERCE
Prentice Hall, 1997



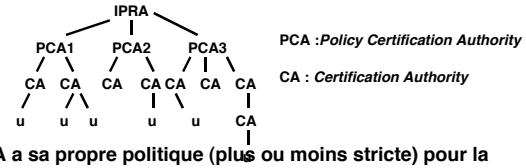
Certificats (2)



d'après : FORD & BAUM,
SECURE ELECTRON IC COMMERCE
Prentice Hall, 1997

Certification : exemple du courrier électronique

- **Solution de l'IETF (Internet) : *Privacy Enhanced Mail* (PEM)**
  - **système hiérarchique, racine = *Internet Policy Registration Authority***



  - **Chaque PCA a sa propre politique (plus ou moins stricte) pour la délégation de sa propre autorité**
- **Solution non officielle : *Pretty Good Privacy* (PGP)**
  - **repose sur un réseau de connaissances**
  - **tout usager peut certifier une clé (la confiance repose sur la connaissance ou la réputation des signataires et sur le nombre des certificats**

Législation française sur la cryptographie (1)

- **Décrets du 17 mars 1999 (N°99-199 et N°99-200) et arrêté relatifs aux moyens et prestations de cryptologie**
  - **libre d'*utilisation* pour** *authentification*, *intégrité* **et** *non-dénégation* (signature électronique)
  - **liberté, déclaration ou autorisation pour *utilisation* dans le cadre de la** *confidentialité* ⇨ chiffrement
    - **< 40 bits : aucune formalité**
    - **40 à 128 bits : dispense de formalité si à usage exclusivement privé ou si déclaration préalable du fournisseur**
    - **> 128 bits : libre si autorisation accordée au fournisseur**
    - **pour utilisation générale ; soumis à autorisation sinon**
  - **Dans le cas de la *fourniture* :**
    - **≤ 128 bits : déclaration**
    - **> 128 bits : autorisation**

http://www.telecom.gouv.fr/francais/activ/techno/technweb1g.htm
http://www.ssi.gouv.fr

Législation française sur la cryptographie (2)

- **Preuves légales**
  - **Projet de loi au 1er septembre 1999 «** *portant adaptation du droit de la preuve aux technologies de l'information et relatif à la signature électronique* **» adopté définitivement le 29 février 2000 .**
  - **Décret d'application le 30 mars 2001**
- **Organismes**
  - **Décrets du 24 février 1998 (N°98-102) définissant les conditions dans lesquelles sont agréés les organismes gérant pour le compte d'autrui des conventions secrètes de cryptologie**
  - **Au futur :**
    - **libéralisation complète de l'utilisation de la cryptologie**
    - **suppression de l'obligation de recourir à des organismes agréés de séquestre de clefs**
  - **mais obligation, sur demande des autorités judiciaires, à la remise des textes en clair.**

Contrôle d'accès

- **Objectif**
  - assurer le "bon usage" d'un ensemble de ressources (ou services) : qui a le droit de faire quoi ?
- **Politiques**
  - contenu
    - définition des autorités (agents, sujets)
    - définition des ressources (objets)
    - définition des droits
  - mode d'élaboration
- **Mécanismes de protection**
  - représentation des autorités, ressources, droits
  - mise en œuvre des autorisations

Un modèle général de protection (1)

- **Éléments du modèle**
  - autorités (agents, sujets) : entités capables d'exécuter des actions
  - ressources (objets) : entités sur lesquelles s'exercent les actions
    - dans ce contexte, un agent peut ausi être objet. Exemple : processus
      - peut lire ou écrire dans un fichier (comme sujet)
      - peut être bloqué ou détruit par un autre processus (comme objet)
  - actions : exécutables par un agent sur un objet
  - droit (d'accès) : autorisation pour un agent d'exécuter une action sur un objet

Un modèle général de protection (2)

- **Matrice de droits**

agents / objets	proc. p1	proc. p2	usager u1	usager u2	groupe g1	groupe g2 …
fichier f1	r, w	r, x	r	r, w, x	r	- -
fichier f2	- -	r, w	- -	r	r, w	r
service s1	appel	- -	appel	- -	appel, changer droits	- -
processus p1	bloquer tuer	bloquer tuer	- -	changer droits bloquer, tuer	bloquer	- -
…						

case (i, j) de la matrice : ensemble des actions autorisées à l'agent i sur l'objet j

Un modèle général de protection (3)

- **Limitations du schéma général**
  - statique, mal adapté aux changements de droits
  - la matrice d'accès est très creuse et ne peut être représentée telle quelle
- **Un schéma amélioré**
  - Notion de domaine de protection
    - inclut un ensemble d'objets et de droits sur ces objets
    - tout processus s'exécutant dans le domaine accède aux objets du domaine avec les droits spécifiés
    - un processus peut changer de domaine (ce qui est une opération protégée, donc associée à des droits)
    - les domaines sont représentés comme des agents
  - Représentations compactes de la matrice de droits
    - par lignes : liste d'accès (*Access Control List*, ou ACL) associée à un objet- liste d'agents et droits associés
    - par colonnes : liste de capacités (*capability lists*) associée à un agent - liste d'objets et droits associés

Listes d'accès

- Pour un objet : {(agent 1, droit 1), … (agent *i*, droit *i*), … }
- Exemple : fichiers Unix
  - 3 agents : l'usager propriétaire, le groupe propriétaire, tous les autres
  - 3 droits de base possibles : lire (r), écrire (w), exécuter (x)
  - le droit "changer les droits" est associé au droit w
  - un droit particulier : *setuid*
    - si le fichier exécutable a le droit *setuid* (bit *setuid* à 1), tout processus peut l'exécuter avec les droits de l'usager propriétaire
    - on réalise ainsi une notion très rudimentaire de domaine : un "domaine" est associé à chaque usager, et les processus créés par cet usager s'exécutent dans ce domaine
    - pour un processus *p*, exécuter un fichier *f* (appartenant à l'usager *u*) revient à "entrer" dans le domaine correspondant à cet usager propriétaire *u*, mais uniquement pour exécuter le fichier *f* spécifié. À la fin de l'exécution, *p* revient dans son domaine d'origine
    - utilisation courante : les fichiers exécutables qui réalisent des servicess du système d'exploitation (exécutés avec les droits de *root*)

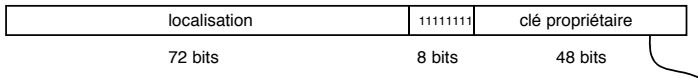
Capacités

- Définition
  - une capacité contient
    - la désignation d'un objet (une information permettant d'atteindre l'objet : pointeur, référence, etc)
    - un ensemble de droits associés à cet objet
  - le détenteur d'une capacité peut accéder à l'objet désigné, avec les droits spécifiés (c'est un "ticket d'accès")
- Problème
  - les capacités doivent être protégées
    - contre la contrefaçon (fabriquer une fausse capacité)
    - contre la modification (augmenter les droits d'une capacité existante)
  - solutions matérielles (machines à capacités, maintenant abandonnées), ou logicielles (chiffrement, coûteux)
- Usage
  - les capacités sont une bonne représentation des domaines (un domaine = une liste de capacités, dont celles permettant d'appeler d'autres domaines)
  - En pratique, utilisation à "gros grain"(service), cf "tickets d'accès" dans Kerberos, ou protection dans les cartes à puce

Exemple d'utilisation de capacités

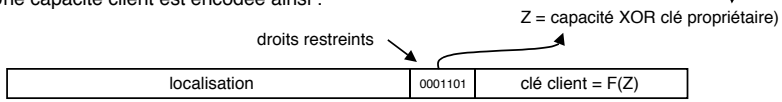
Utilisé dans le système expérimental Amoeba, qui gère des objets répartis.

Capacité = informations de localisation de l'objet + droits + clé (aléatoire)



La capacité du propriétaire a tous les bits de droits à 1.
Problème : distribuer des capacités à d'autres utilisateurs (clients), avec des droits restreints, sans qu'ils puissent augmenter les droits

Solution : le serveur conserve la capacités propriétaire avec l'objet.
Une capacité client est encodée ainsi :



F fonction de hachage non inversible

Quand le serveur reçoit une capacité d'un client, il calcule
Z' = capacité du client XOR clé propriétaire (qu'il possède), puis F(Z'), et compare F(Z') avec la clé client de la capacité. La capacité n'est acceptée que s'il y a identité.
Si les droits ont été manipulés, il n'y a pas identité. Principe de la signature électronique

Politiques de protection

- Politiques discrétionnaires
  - en fait, pas de "politique" au sens d'ensemble de règles cohérentes
  - les droits d'accès sont définis ad hoc, au cas par cas
- Politiques "régulées" (*mandatory*)
  - l'attribution des droits d'accès est guidée par un ensemble de règles définissant la politique
  - Exemple (le plus largement répandu) : modèle de Bell et LaPadula, pour la confidentialité

Exemple de politique de protection régulée : le modèle hiérarchique (Bell et LaPadula)

- **Contexte**
  - on définit un ensemble ordonné de niveaux de confidentialité (ex : libre, confidentiel, secret, très secret)
  - tout objet est classifié à un niveau
  - tout agent est habilité pour un niveau (et les niveaux inférieurs)
- **Règles**
  - un agent habilité au niveau n peut
    - lire uniquement les objets aux niveaux $i \leq n$
    - écrire uniquement dans les niveaux $j \geq n$
      - explication par l'exemple : s'il est habilité au niveau "secret", il risquerait, si la règle d'écriture n'était pas suivie, de lire un objet classé "secret" et de le recopier dans un niveau "confidentiel"
  - les droits d'accès ne peuvent être attribués que s'ils suivent ces règles (politique régulée)

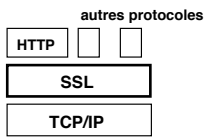
Deux exemples de protocoles sécurisés sur les réseaux

- **Un protocole général : SSL**
  - destiné à sécuriser un échange client-serveur (confidentialité, authentification)
- **Un protocole spécialisé pour les paiements : SET**
  - destiné à sécuriser les paiements par carte bancaire (confidentialité, authentification, intégrité, non-dénégation)

SSL http://home.netscape.com/security/techbriefs/ssl.html
SET http://www.setco.org

Secure Socket Layer (SSL)

- **Fonctions**
  - Protocole d'échanges de données (au-dessus de TCP/IP), qui assure
    - la confidentialité des échanges entre 2 applications
    - l'authentification des serveurs
  - Origine : Netscape - utilisé en particulier pour HTTP
- **Principe**
  - utilise RSA (clé publique) pour échanger des clés DES (secrètes)
  - protocole de négociation (choix clés)
  - protocole d'échange (chiffré par DES)
  - ne fournit pas de garantie contre le déni
  - authentifie un navigateur, non une personne



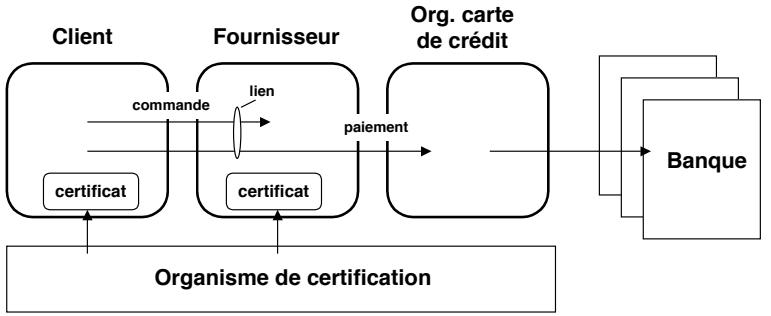
Fonctionnement de SSL

- **Phase de négociation**
  - Authentification
    - utilise des certificats émis par une autorité de certification
    - authentifier le serveur vis-à-vis du client (navigateur)
    - authentifier le navigateur vis-à-vis du serveur (facultatif)
  - Génération des clés de session
    - génération et échange préalable (par RSA) d'une donnée commune de création de clés (*master secret*)
    - création des clés de session (secrètes) par le client et le serveur, en utilisant le *master secret*
  - À la fin de la négociation
    - le client et le serveur sont authentifiés mutuellement
    - le client et le serveur disposent de clés secrètes pour la phase d'échange

Secure Electronic Transactions (SET)

- **Origine**
  - Protocole développé par Visa et MasterCard, avec d'autres partenaires (IBM, Microsoft, Netscape, etc.)
  - Spécification publique, gérée par un consortium
- **Objectifs**
  - Assurer le paiement par carte bancaire sur l'Internet, avec garanties de :
    - confidentialité
    - intégrité de données
    - authentification du client et du fournisseur
    - non-déni pour le client et le fournisseur

SET : les entités participantes



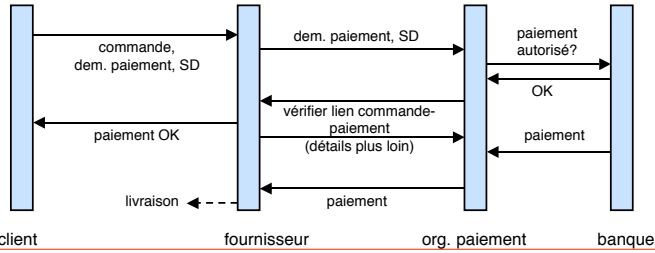
- Le client et le fournisseur doivent avoir un certificat (délivré par une "autorité habilitée")
- Le fournisseur ne voit aucune information de paiement
- L'organisme de cartes de crédit ne voit aucune information relative à la commande
- La banque ne communique qu'avec l' organisme de cartes de crédit

SET : principe de fonctionnement -1
(très simplifié)

- **À l'origine**
  - Le client obtient une carte de crédit (banque + org. carte)
  - Le client obtient un certificat (org. de certifcation)
  - Le fournisseur obtient un certificat (id.)
  - Les certificats contiennent les clés publiques, avec garantie d'appartenance
- **Phase initiale (côté client)**
  - Le client (navigateur) contacte le fournisseur et obtient une offre de prix
  - Le client (navigateur) vérifie la validité du certificat du fournisseur
- **Commande (côté client)**
  - Le client envoie :
    - sa commande, chiffrée avec la clé publique du fournisseur
    - un ordre de paiement chiffré avec la clé publique de l'organisme de crédit
    - une information (signature duale) faisant le lien avec les deux précédentes
  - Ces messages sont envoyés au fournisseur (le client ne communique pas directement avec l'organisme de crédit)

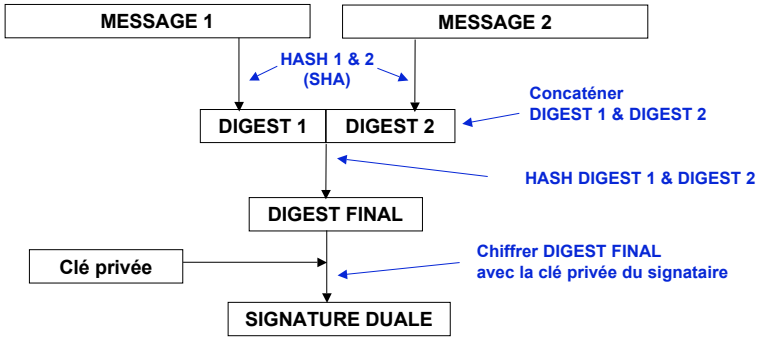
SET : principe de fonctionnement -2
(très simplifié)

- Le fournisseur vérifie le certificat du client
- Le fournisseur envoie un message de paiement à l'org. de crédit, chiffré avec la clé de celui-ci, contenant
  - l'ordre de paiement du client (non lisible par le fournisseur)
  - la signature duale
  - le certificat du fournisseur
- L'org. de crédit vérifie
  - l'identité du fournisseur (certificat)
  - le message de paiement et l'information qu'il contient
  - la solvabilité du client (banque)
- L'org. de crédit envoie une autorisation (authentifiée par signature) au fournisseur
- Le fournisseur peut alors se faire payer et délivrer la commande



Une technique utilisée dans SET : la signature duale

- Associe deux messages, envoie le tout à deux destinataires, mais seul un destinataire peut lire chacun des messages.



Usage de la signature duale

A veut envoyer Message 1 à B et Message 2 à C
B ne doit pas voir Message 2, C ne doit pas voir Message 1 ...

Exemple dans SET
Message 1 = commande
Message 2 = ordre de paiement
A = client, B = fournisseur, C = org. carte crédit

... mais B et C doivent être sûrs que les 2 messages sont liés et non modifiés

A -> B : { Message 1, Digest 2, Signature Duale}
A -> C (en fait via B) : { Message 2, Digest 1, Signature Duale}

B construit SHA(Message 1) = Digest 1, le concatène avec Digest 2 et fait SHA du total (obtient Digest Final)
B déchiffre la signature duale avec la clé publique de A.
Si le résultat est bien Digest Final, tout est OK (message intact et authentifié)

B->C : Digest 2 ; B demande à C s'il a bien reçu le message correspondant à Digest 2 (soit Message 2). Si oui, le lien est établi.

Actions symétriques pour C vis à vis de B.

Conclusions sur la sécurité

- **Autres aspects non traités**
  - Sécurité liée au code mobile
    - Protection du code
    - Protection des récepteurs (*sandbox*, vérificateurs, etc.)
  - Techniques de protection des réseaux
    - Pare-feux (*firewalls*)
  - Protection contre les intrusions
  - Déni de service
- **Prospective**
  - Domaine important
    - Théorie
    - Impact pratique
  - Besoins
    - Fondements théoriques pour la spécification et la preuve
    - Construction automatique de code pour la sécurité
    - Liens avec aspects sociaux et juridiques

Références

Cryptographie
B. Schneier, *Cryptographie Appliquée*, Thomson, 1998
Kerberos
RFC1510 : http://www.rfc-editor.org/rfc/rfc1510.txt
RSA
http://rsasecurity.com/rsalabs/
MD5
RFC1321 : http://www.rfc-editor.org/rfc/rfc1321.txt
SSL
http://home.netscape.com/security/techbriefs/ssl.html
SET
http://www.setco.org
Certification
http://sitesearch.netscape.com/certificate/v1.0/faq/
W. Ford, M. S. Baum, Secure Electronic Commerce, Prentice Hall, 1997

Pare-feux
E. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security*, Addison-Wesley, 1994

Législation française sur la cryptographie :
http://www.telecom.gouv.fr/francais/activ/techno/technweb1g.htm
http://www.ssi.gouv.fr

Référence générale : G. Coulouris, J. Dollimore, T. Kindberg, *Distributed Systems*, Addison-Wesley, 2000, chapitre 7