

Sécurité des systèmes répartis - 2

Sacha Krakowiak
Université Joseph Fourier
Projet Sardes (INRIA et IMAG-LSR)
<http://sardes.inrialpes.fr/~krakowia>

Sécurité des systèmes client-serveur : 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

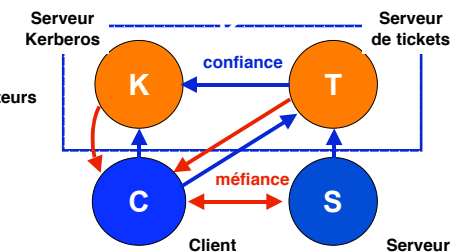
Relations de confiance

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

$$T_{CS} = \{S, C, \text{adr}, Td, \text{life}, K_{CS}\}_{K_S}$$

- ❖ **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

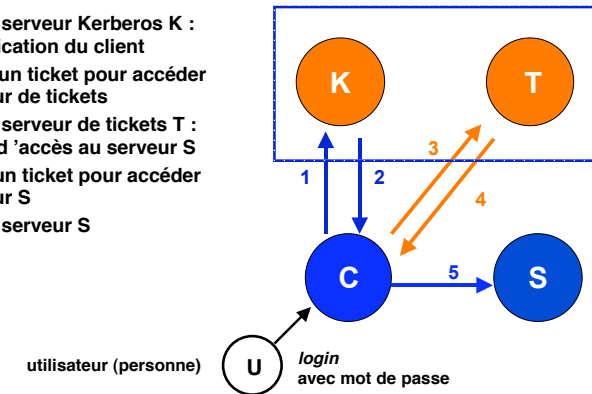
$$A_{CS}(t) = \{C, \text{adr}, t\}_{K_{CS}}$$

- ❖ engendré par le client
- ❖ permet une authentification « permanente » par le serveur

Kerberos (3)

■ Architecture

- 1 accès au serveur Kerberos K : authentification du client
- 2 retour d'un ticket pour accéder au serveur de tickets
- 3 accès au serveur de tickets T : contrôle d'accès au serveur S
- 4 retour d'un ticket pour accéder au serveur S
- 5 accès au serveur S



Kerberos (4)

■ Interaction entre client et serveur Kerberos

- 1 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

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

- 2 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 : $A_{C,T}(t1) = \{C, \text{adr}, t1\}_{K_{C,T}}$

- 3 C → T : message M3 = ($A_{C,T}$; $T_{C,T}$; S)
rappel : $T_{C,T} = \{T, C, \text{adr}, \text{td}, \text{life}, K_{C,T}\}_{K_T}$

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

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

- 4 T → C : message M4 = { $K_{C,S}$, $T_{C,S}$ } _{$K_{C,T}$}

Kerberos (6)

■ Interaction entre le client et le serveur S

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

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

(rappel : $T_{c,s} = \{S, C, \text{adr}, \text{td}, \text{lfe}, K_{c,s}\}_{K_s}$)

C construit un authentifieur : $A_{c,s}(t2) = \{C, \text{adr}, t2\}_{K_{c,s}}$

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 (déchiffirable avec la clé publique correspondante) ; soit K_{SA}

❖ A envoie à B : $(M, \{M\}_{K_{SA}})$

❖ B déchiffre $\{M\}_{K_{SA}}$ avec K_{PA} (publique)

❖ B compare $[\{M\}_{K_{SA}}]_{K_{PA}}$ avec M

❖ Si différents, le message (ou la signature) a été altéré

❖ Propriétés : vérification d'intégrité, authentification, non dénégarion

◆ 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

$$\begin{array}{ccc} \downarrow [\]_{KS} & & \downarrow [\]_{KPA} \\ M \rightarrow H(M) & \stackrel{=?}{=} & DM \end{array}$$

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 associe 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

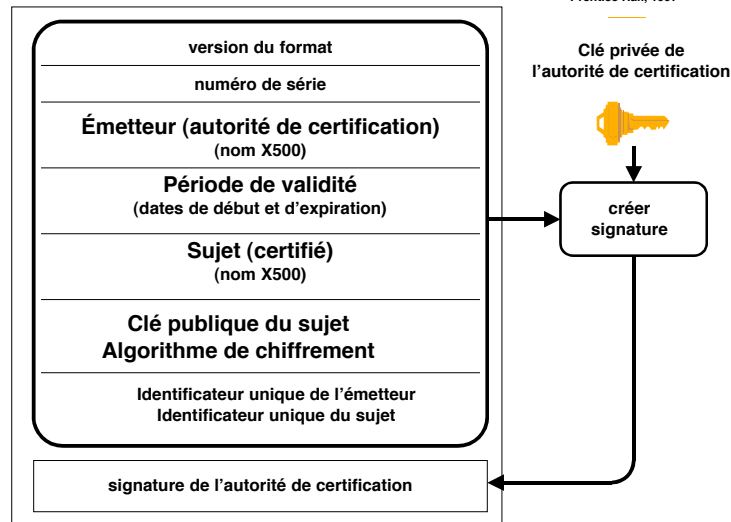
- ◆ certificat : protocole normalisé X509 (ITU-T X.509 international standard V3 - 1996) (*RFC2459*)
- ◆ liste des certificats révoqués (CRL *Certificate Revocation List*)
 - ❖ structures de données signées
 - ❖ format défini par le protocole X509 V2 CRL (*RFC2459*)
- ◆ support logique de publication : LDAP « *Lightweight Directory Access Protocol* » (*RFC2251*)

■ Garantie

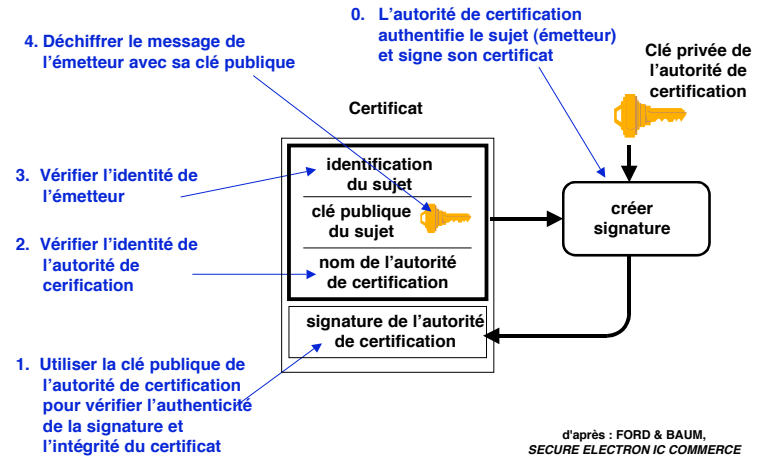
- ◆ Un certificat est garanti par la signature de l'autorité qui l'a délivré (à condition)
 - ❖ que la date d'expiration ne soit pas passée
 - ❖ qu'il n'ait pas été révoqué

Certificats (1)

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



Certificats (2)

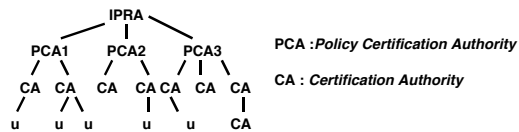


d'après : FORD & BAUM,
SECURE ELECTRON IC COMMERCE
Prentice Hall, 1997

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)

Fonctionnement usuel de la certification

■ Le contrôle des certificats est intégré au navigateur

- ◆ le navigateur maintient la liste des autorités "reconnues" par l'utilisateur
- ◆ le navigateur maintient les certificats obtenus par l'utilisateur

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 géant 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 aussi 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	--
...				bloquer, tuer		

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 services 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)

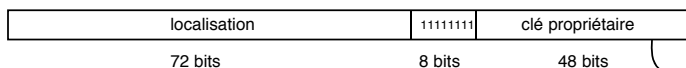
■ Usage

- ◆ 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.

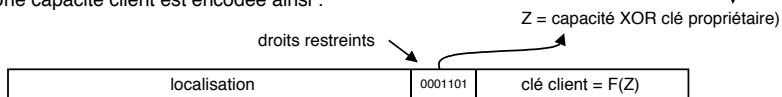
Capacité = informations de localisation de l'objet + droits + clé (aléatoire)



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>

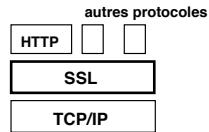
Secure Socket Layer (SSL)

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)

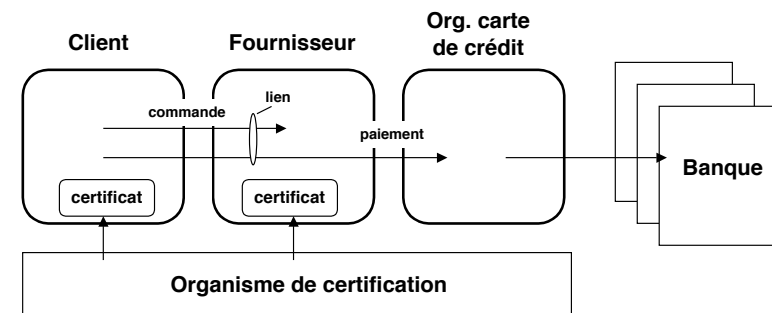
Origine

- ◆ 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



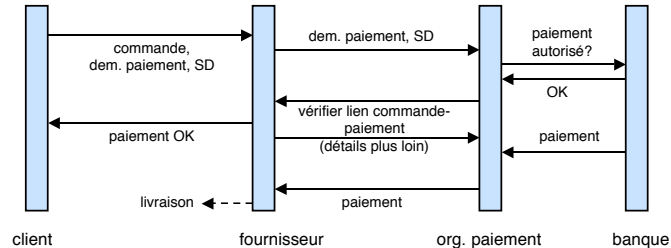
- ◆ Le client et le fournisseur doivent avoir un certificat (délivré par une "autorité habilitée")
- ◆ Le fournisseur ne voit aucune information de paiement
- ◆ L'organisme de cartes de crédit ne voit aucune information relative à la commande
- ◆ La banque ne communique qu'avec l'organisme de cartes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

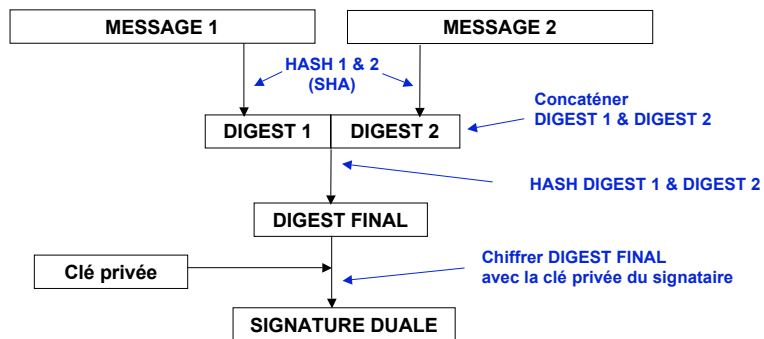
SET : principe de fonctionnement -2 (très simplifié)

- 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.

Sécurisation des réseaux

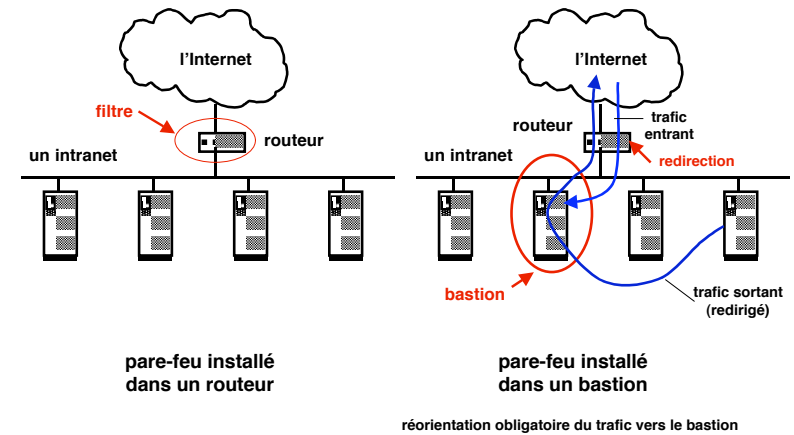
■ Problème

- ◆ protéger le réseau interne d'une organisation (intranet), tout en permettant certains types d'accès vers ou depuis l'extérieur (Internet)
- ◆ les types d'accès permis doivent être soigneusement et exhaustivement spécifiés
 - ❖ de manière tolérante (tout ce qui n'est pas interdit est permis)
 - ❖ de manière restrictive (tout ce qui n'est pas permis est interdit)

■ Techniques

- ◆ implanter des fonctions de **filtrage** (laisser passer, rediriger ou arrêter un message) dans les organes de liaison entre réseaux (routeurs) et/ou dans certaines stations hôtes (bastions)
- ◆ le filtrage peut être fait à différents niveaux
 - ❖ bas niveau (paquets IP)
 - ❖ haut niveau (par exemple application)

Sécurisation d'un Intranet : pare-feu (firewall)



Filtrage de paquets

■ Principe

- ◆ Analyser chaque paquet IP entrant ou sortant
- ◆ Décider d'arrêter ou laisser passer, en fonction de la politique choisie et des informations suivantes
 - ❖ adresses IP origine (supposée) et destination
 - ❖ adresses de porte UDP ou TCP origine et destination
 - ❖ type de message (protocole)
 - ❖ message d'acquittement ou synchronisation (TCP)

■ Mise en œuvre

- ◆ Dans un routeur

■ Caractéristiques

- ◆ Réalisation simple, mais sélectivité limitée
- ◆ Filtrage sur une classe de services, non sur des utilisateurs
- ◆ Ne prend pas en compte le contenu des données

Programmation d'un pare-feu filtre de paquets (1)

- ◆ **Politique** : autoriser uniquement l'accès aux services Telnet externes
- ◆ **Caractéristiques du service Telnet**
 - ❖ porte réservée : 23
 - ❖ protocole : TCP
- ◆ **Propriétés des paquets utilisés dans Telnet**

	Direction service	Direction paquet	Adresse source	Adresse destinat.	Protocole	Porte source	Porte destinat.	ACK
E	sortant	out	interne	externe	TCP	X	23	oui sauf premier
	sortant	in	externe	interne	TCP	23	Y	oui
I	entrant	in	externe	interne	TCP	Z	23	oui sauf premier
	entrant	out	interne	externe	TCP	23	T	oui

E : serveur externe

X, Y, Z, T > 1023

I : serveur interne

Programmation d'un pare-feu filtre de paquets (2)

Programme du pare-feu

Règle	Direction paquet	Adresse source	Adresse destinat.	Protocole	Porte source	Porte destinat.	ACK	Décision
A	out	interne	*	TCP	>1023	23	*	Accepter
B	in	*	interne	TCP	23	>1023	oui	Accepter
C	*	*	*	*	*	*	*	Rejeter

Règle A : appel d'un serveur Telnet à partir d'un client interne, paquets entrants (y compris 1er)

Règle B : paquet de réponse

Règle C : interdit tout le reste

Programmation d'un pare-feu filtre de paquets (3)

Autre exemple

Politique :

- autoriser connexions vers les adresses IP internes 123.4.*.*
- autoriser connexions Telnet (p. 23) vers 123.4.5.6 (le relais Telnet)
- autoriser connexions SMTP (mail, p. 25) vers 123.4.5.7 et 123.4.5.8 (relais mail)
- autoriser connexions NNTP (news, p. 119) depuis 129.6.48.254 (serveur news) et seulement vers 123.4.5.9 (relais news)
- autoriser connexions NTP (Network Time Protocol, p. 123) vers tous les sites

Règle	Direction paquet	Adresse source	Adresse destinat.	Protocole	Porte source	Porte destinat.	Décision
A	in	*	123.4.5.6	TCP	>1023	23	Accepter
B	in	*	123.4.5.7	TCP	>1023	25	Accepter
C	in	*	123.4.5.8	TCP	>1023	25	Accepter
D	in	129.6.48.254	123.4.5.9	TCP	>1023	119	Accepter
E	in	*	123.4.*.*	UDP	>1023	123	Accepter
F	in	*	*	*	*	*	Rejeter
...	out
....	out

Filtrage au niveau de l'application

■ Principe

- ◆ Analyse spécifique au niveau d'une application (porte sur le contenu des paquets, non seulement sur l'enveloppe)

■ Mise en œuvre

- ◆ Réorientation des paquets (par le routeur) vers une station dédiée au filtrage
- ◆ Mise en œuvre de la politique par un serveur spécifique sur la station de filtrage
- ◆ Plusieurs serveurs différents peuvent coexister

■ Exemple

- ◆ Serveur de *login* sur un intranet (filtre les connexions externes et impose un contrôle strict, ex. mot de passe à usage unique)
- ◆ Restriction du service *telnet* vers l'extérieur
- ◆ Restriction sur le courrier entrant ou sortant

Exemple de filtrage

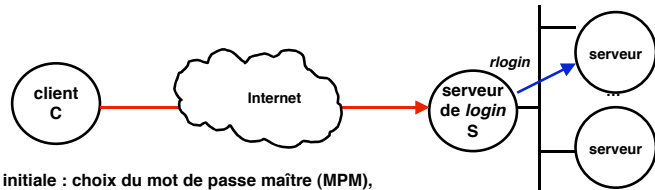
■ Passage obligé par un serveur de *login*

- ◆ *login* impossible sauf sur un serveur dédié
- ◆ puis *rlogin* (interne) depuis ce serveur vers la machine choisie

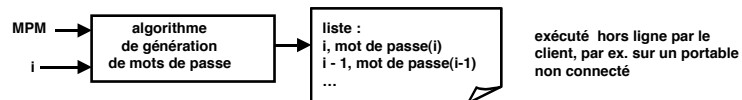
■ *login* sécurisé avec OTP (*One Time Password*)

- ◆ idée
 - ❖ le mot de passe change à chaque *login*
 - ❖ les mots de passe sont engendrés à partir d'un mot de passe maître, secret partagé entre le serveur de *login* et le client, mais qui ne circule jamais sur l'Internet
 - ❖ un mot de passe intercepté ne peut pas resservir

Fonctionnement de l'OTP



Phase initiale : choix du mot de passe maître (MPM), partagé avec le serveur de *login* (communication hors Internet)



exécuté hors ligne par le client, par ex. sur un portable non connecté

Protocole de *login* (sur l'Internet)

C -> S : login C

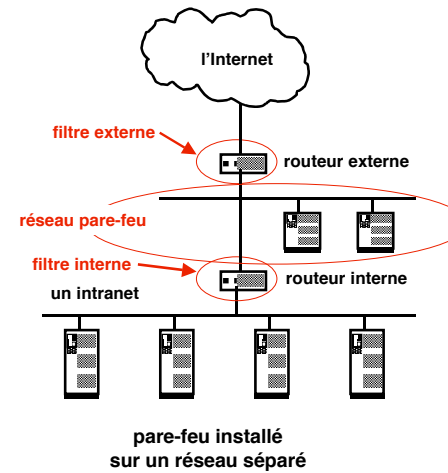
S -> C : i

C -> S : mot de passe(i)

S calcule mot de passe(i) à partir de MPM et i, par le même algorithme que le client, et vérifie que c'est bien celui fourni par le client

S décrémente i de 1 à chaque nouveau *login* de C

Réseau pare-feu



Avantages

- ◆ double protection (si le routeur externe est franchi, l'intranet reste protégé)
- ◆ possibilité de mettre en œuvre une politique élaborée (serveurs de filtrage sur le réseau interne)

Inconvénients

- ◆ coût élevé

Précautions

- ◆ un seul routeur interne (plusieurs routeurs externes possibles)

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